Further Component Oriented Systems

Deepak Dhungana

dhungana@ase.jku.at

Institute for System Engineering and Automation

Thomas Wuerthinger

wuerthinger@ssw.jku.at

Institute for System Software

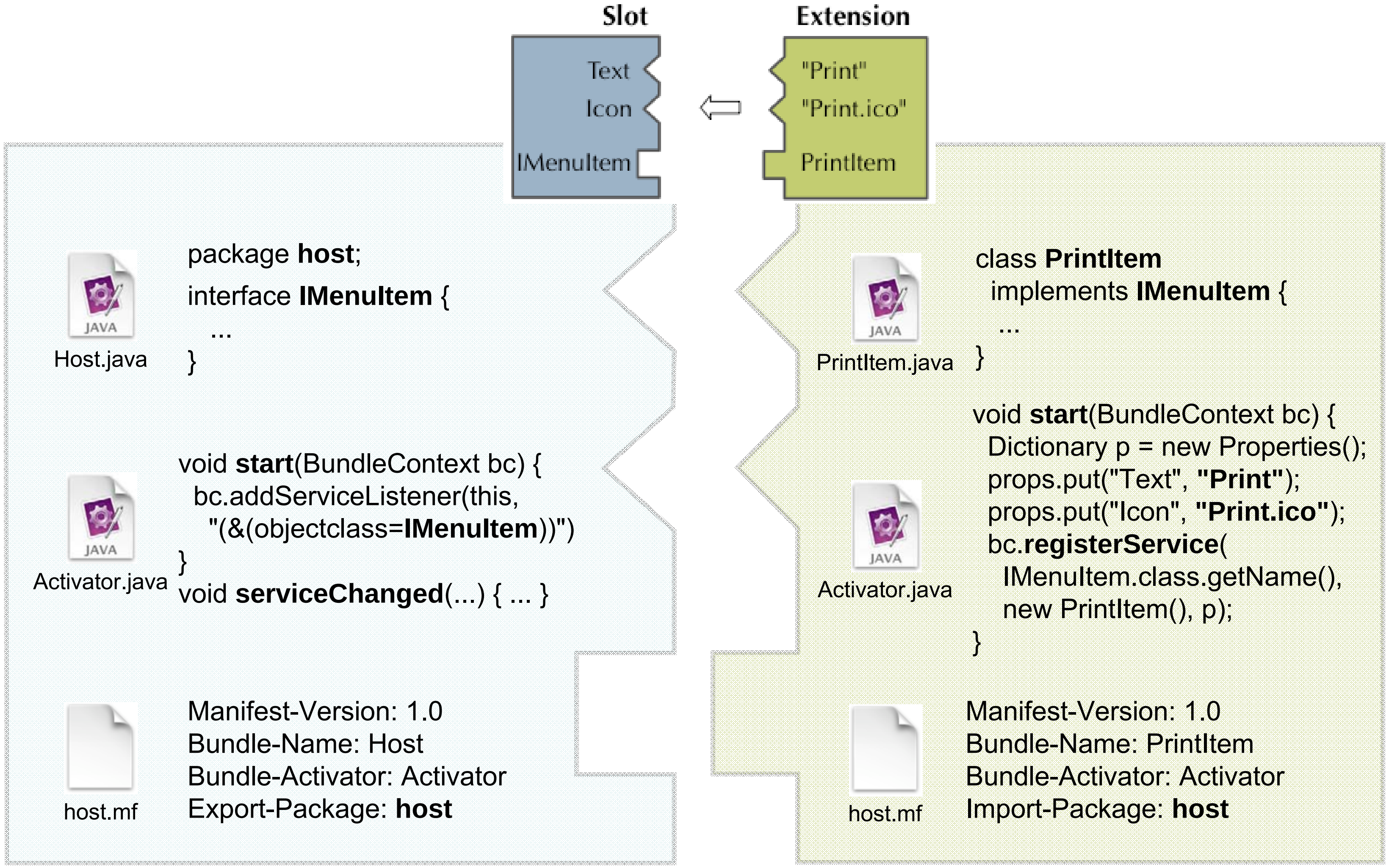
Johannes Kepler University Linz, Austria
http://www.jku.at

On the agenda today

- Plux.NET
- Mozilla
- Microsoft COM
- Visual Studio

Plux.NET

OSGi



Slot

Text
Icon
IMenuItem

⇦

Extension

"Print"
"Print.ico"
PrintItem

Host.java

```
package host;
interface IMenuItem {
    ...
}
```

Activator.java

```
void start(BundleContext bc) {
  bc.addServiceListener(this,
    "(&(objectclass=IMenuItem))")
}
void serviceChanged(...) { ... }
```

host.mf

```
Manifest-Version: 1.0
Bundle-Name: Host
Bundle-Activator: Activator
Export-Package: host
```

PrintItem.java

```
class PrintItem
  implements IMenuItem {
    ...
}
```

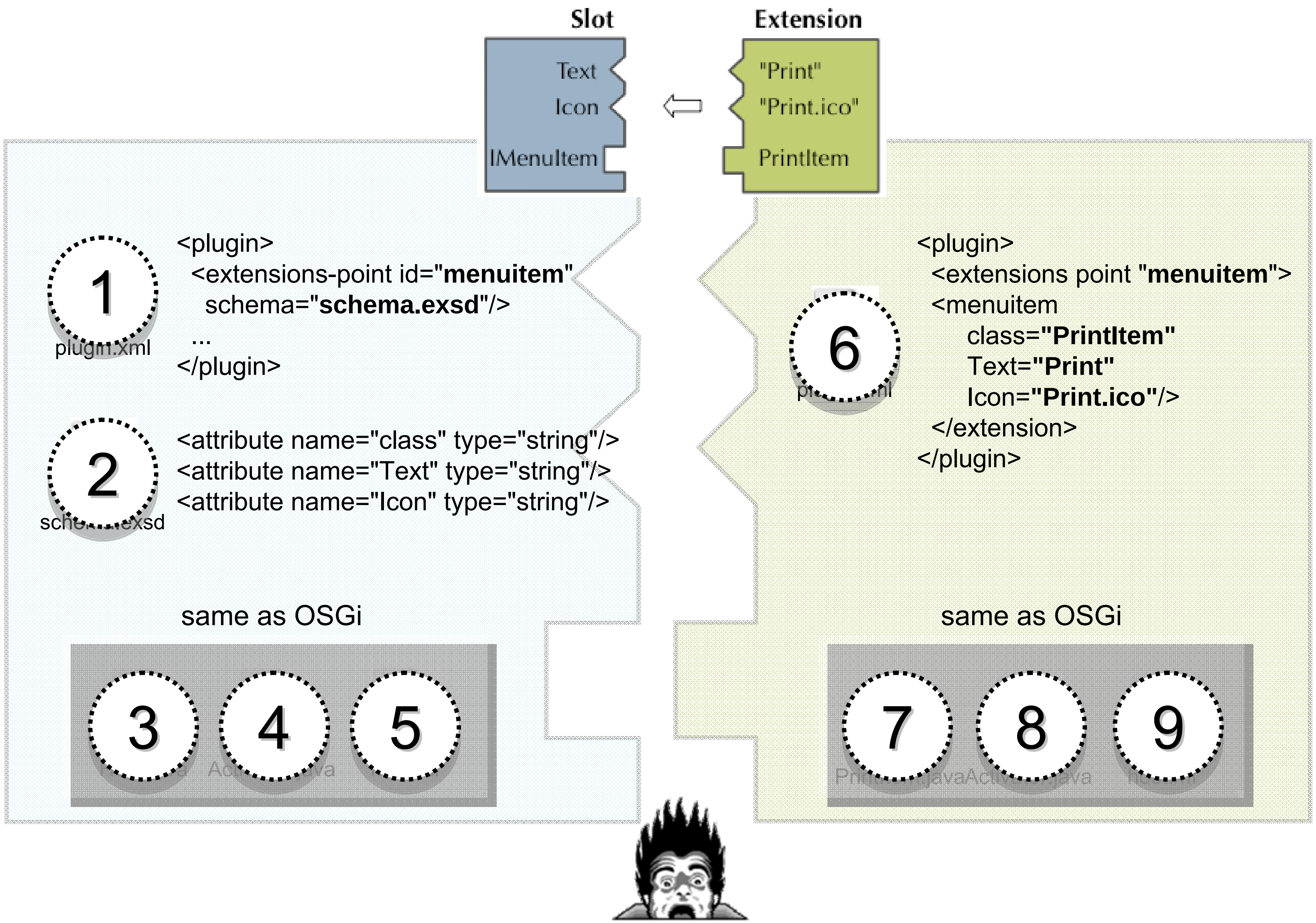
Activator.java

```
void start(BundleContext bc) {
  Dictionary p = new Properties();
  props.put("Text", "Print");
  props.put("Icon", "Print.ico");
  bc.registerService(
    IMenuItem.class.getName(),
    new PrintItem(), p);
}
```

host.mf

```
Manifest-Version: 1.0
Bundle-Name: PrintItem
Bundle-Activator: Activator
Import-Package: host
```

Eclipse



```
<plugin>
  <extensions-point id="menuitem"
    schema="schema.exsd"/>
  ...
</plugin>
```

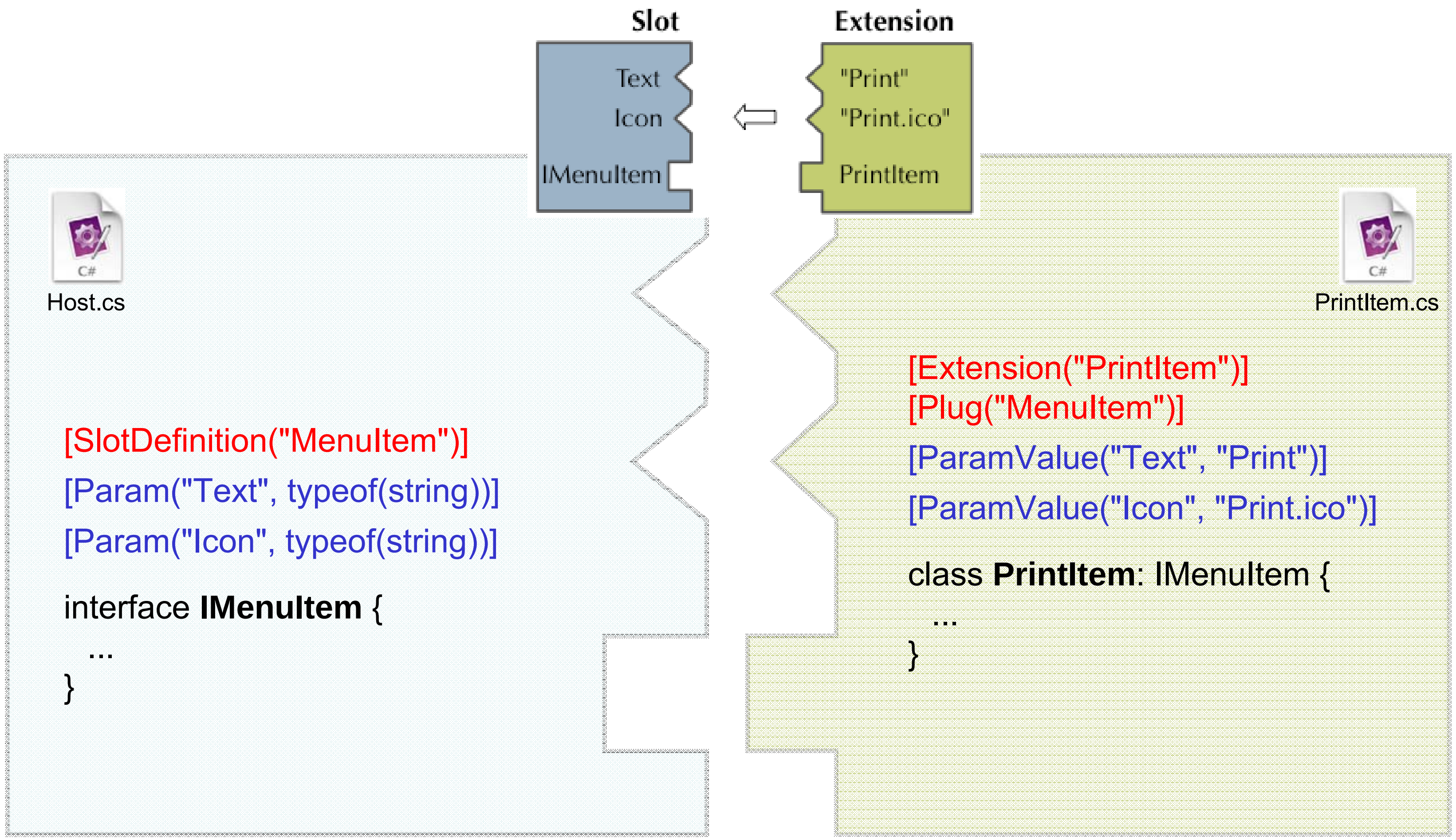
```
<attribute name="class" type="string"/>
<attribute name="Text" type="string"/>
<attribute name="Icon" type="string"/>
```

same as OSGi

```
<plugin>
  <extensions point "menuitem">
  <menuitem
       class="PrintItem"
       Text="Print"
       Icon="Print.ico"/>
  </extension>
</plugin>
```

same as OSGi

Plux.NET



```
[SlotDefinition("MenuItem")]
[Param("Text", typeof(string))]
[Param("Icon", typeof(string))]
interface IMenuItem {
   ...
}
```

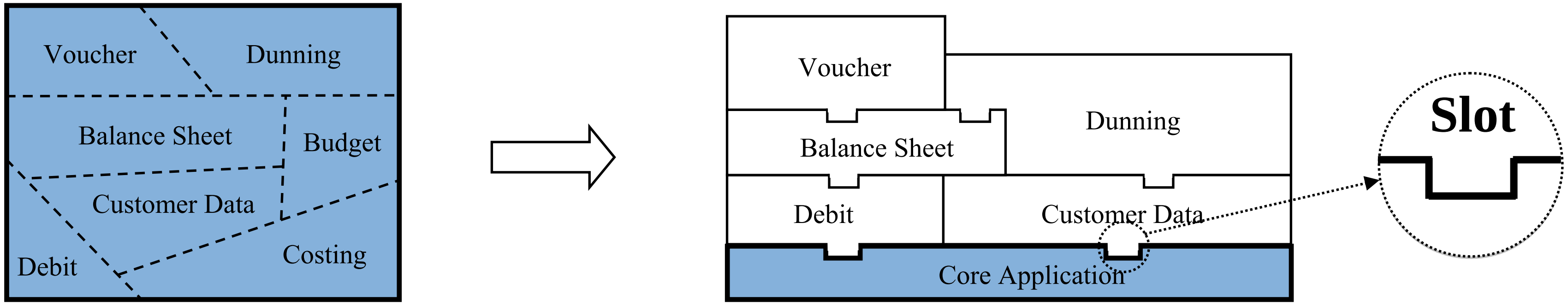
```
[Extension("PrintItem")]
[Plug("MenuItem")]
[ParamValue("Text", "Print")]
[ParamValue("Icon", "Print.ico")]
class PrintItem: IMenuItem {
   ...
}
```

Wolfinger, R., Dhungana, D., Prähofer, H., and Mössenböck, H.:
A Component Plug-in Architecture for the .NET Platform.
7th Joint Modular Languages Conference (JMLC), 2006.

Plux.NET

- Adopt and adapt plug-in components to domain of enterprise applications.

Today's enterprise application *New architecture*



- coarse-grained components
- monolithic piece of software

- slim core application
- extend with features
- plug into core application
- integrate seamlessly
- as simple as dropping an executable into the application folder
- dynamic addition/removal/replacement
- unlimited but controlled extensibility

Define a Slot

A *slot definition* specifies how a host wants to be extended
•*contract* specifies expected behaviour

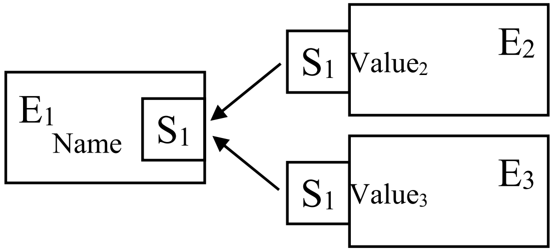
```
[SlotDefinition(“S1”)]
interface IContract {
  void Foo();
}
```

- *parameters* specify metadata
- can be determined w/o loading code
- choose from different contributors
- integrate statically (lazy load)

```
[Param(“Name”, typeof(string)]
[SlotDefinition(“S1”)]
interface IContract {
  void Foo();
}
```

A *host extension* opens a slot and ...

```
[Extension(“E1”)]
[Slot(“S1”)]
class E1 { ... }
```



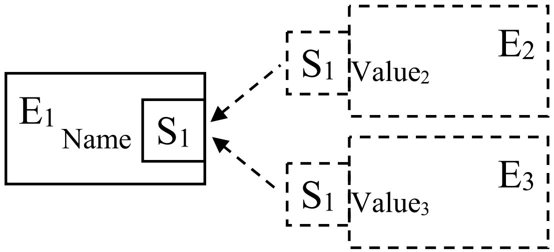
contributor extensions fill a slot.

```
[Extension(“E2”)]
[Plug(“S1”)]
[ParamValue(“Name”, “Value”)]
class E2 : IContract { ... }
```

Attach to a Slot

To *attach* means to notify a host that contributor has been added

•parameter values can be read



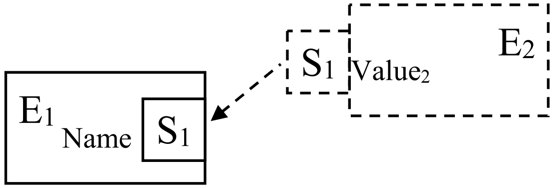
```
[Slot("S1", OnAttached="S1_Attached")]
class E1 {
  void S1_Attached(object s, AttachEventArgs args) {
    string v = args.GetParamValue("Name");
    // integrate extension
  }
}
```

- for instance create UI widgets
- based on metadata
- defer loading until widget is clicked
- extensions not qualified for slot definition are not attached

To *detach* means to notify a host that a contributor has been removed

•plug-in removed

•configuration changed

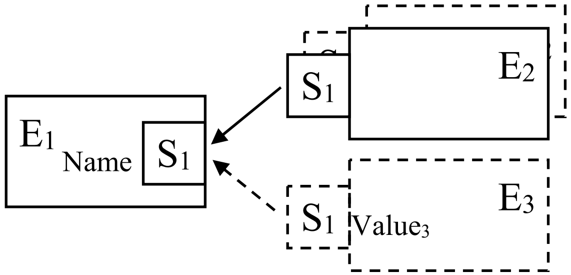


```
[Slot("S1", OnDetached="S1_Detached")]
class E1 {
  void S1_Detached(object s, AttachEventArgs args) {
    // disintegrate extension
  }
}
```

- remove UI widgets

Plug into a Slot

To *plug* means to instantiate contributor and start communication with slot



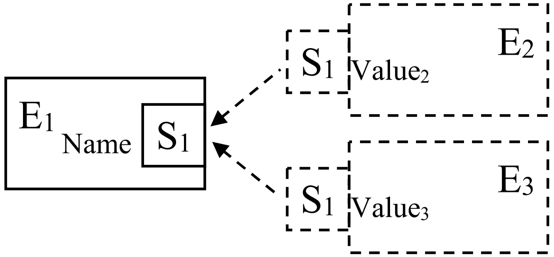
```
[Slot("S1", OnPlugged="S1_Plugged")]
class E1 {
  void S1_Plugged(object s, PlugEventArgs args) {
    IContract obj = (IContract) args.Extension;
    obj.Foo();
  }
}
```

- host uses contributor

To *unplug* means to stop communication and release contributor

•plug-in removed

•configuration changed



```
[Slot("S1", OnUnplugged="S1_Unplugged")]
class E1 {
  void S1_Unplugged(object s, PlugEventArgs args) {
    // release references to contributor
  }
}
```

Secure Extensibility

Constrain what extensions can do

•based on digitally signed plug-ins

-creator always trusted

•compositional constraints

-who can use a slot definition

→ *Protected Slot Definition*

-who can attach to slot

→ *Public Slot, Private Slot*

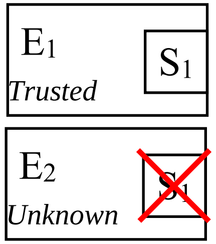
-who can open slots

→ *Bounded Slot*

•code access constraints

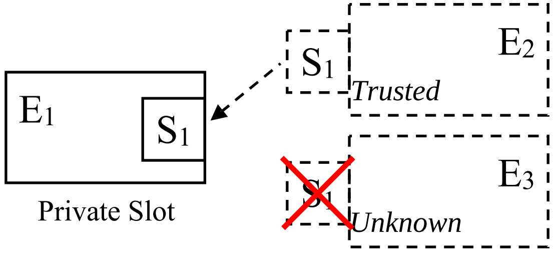
-what can plug-in do

→ *Secure Slot*



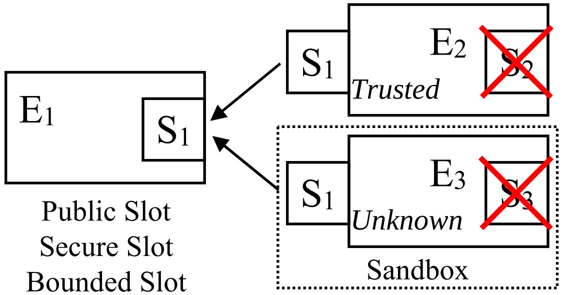
Protected Slot Definition

```
[SlotDefinition("S1")]
[ProtectedSlot]
[AllowOpen(
  "Trusted.certificate"]
interface IContract { ... }
```



Private Slot

```
[Extension("E1")]
[PrivateSlot("S1")]
[AllowAttach(
  "Trusted.certificate"]
class E1 { ... }
```

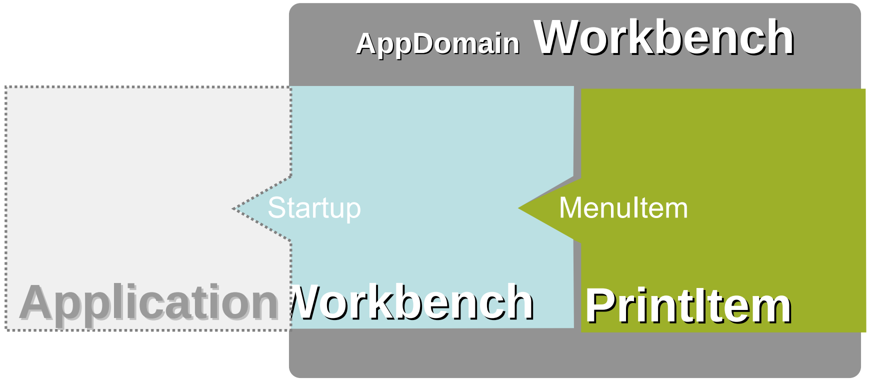


Public Slot
Secure Slot
Bounded Slot

```
[PublicSlot("S1", Bounded=true)]
[AllowPlug("Trusted.certificate",
  PermissionSet="Full"]
[AllowPlug(
  PermissionSet="Sandbox")]
class E1 { ... }
```

Integrationsmodelle (1)

- Tightly Coupled - no Isolation
  - Host und Plug-in teilen eine AppDomain
    - für Komponenten der Kernapplikation

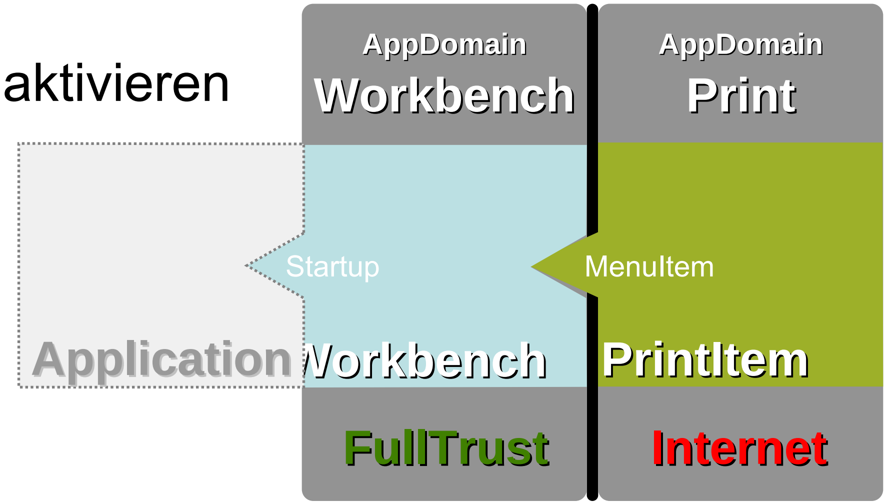


```
[Extension(Name="Workbench", Slot="Startup")]
[Isolation(None)][Domain("Workbench")]
class Workbench : IStartup {
  ...
}
```

```
[Extension(Name="PrintItem", Slot="MenuItem")]
[Domain("Workbench")]
class PrintItem : IMenuItem {
  ...
}
```

Integrationsmodelle (2)

- AppDomain Isolation
  - Plug-in in separater AppDomain aktivieren
    - Plug-in entladen
    - beschränken was Plug-in darf



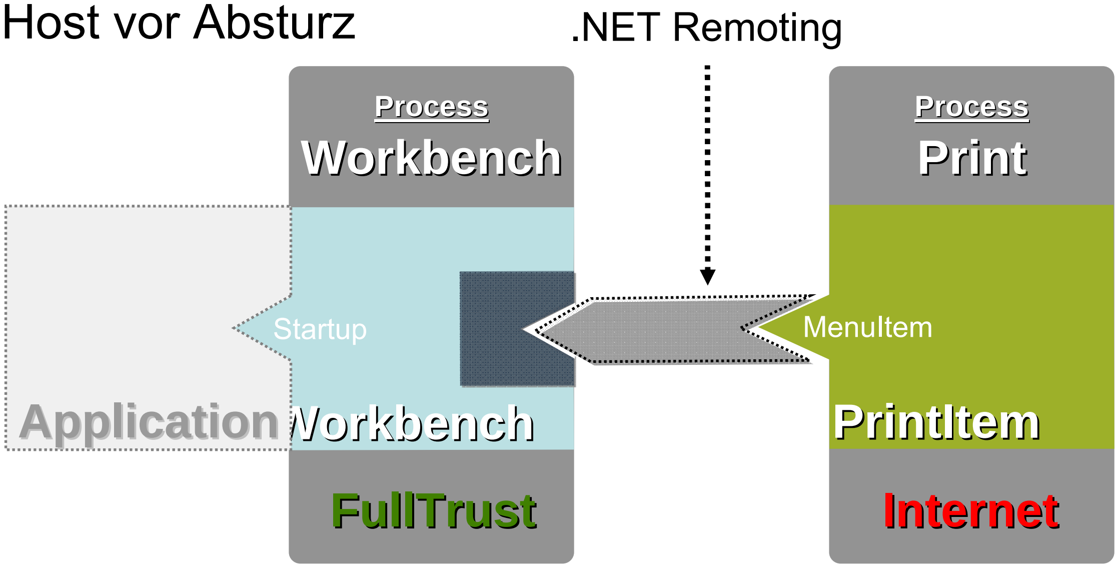
```
[Slot("MenuItem")][Security(PermissionSet=Internet)]
interface IMenuItem {  ...
}
```

```
[Extension(Name="Workbench", Slot="Startup")]
[Isolation(AppDomain)]
[Domain("Workbench")]
class Workbench : IStartup {
  ...
}
```

```
[Extension(Name="PrintItem", Slot="MenuItem")]
[Domain("Print")]
class PrintItem : IMenuItem {
  ...
}
```

Integrationsmodelle (3)

- Process Isolation
  - Plug-in in separatem Prozess aktivieren
    - schützt Host vor Absturz



```
[Extension(Name="Workbench", Slot="Startup")]
[Isolation(Process)][Domain("Workbench.Workbench")
]
class Workbench : IStartup {
 ...
}
```

```
[Extension(Name="PrintItem", Slot="MenuItem")]
[Domain("Print.Print")]
class PrintItem : IMenuItem {
  ...
}
```

Motivating Scenarios

–Developed together with our industry partner BMD Systemhaus

- Szenario 1: On-the-fly product customization

–Live-Preview of application, “I know it when I see it”

- Szenario 2: Guided system upgrades

–Which additional features are possible with current configuration?

- Szenario 3: Renting features

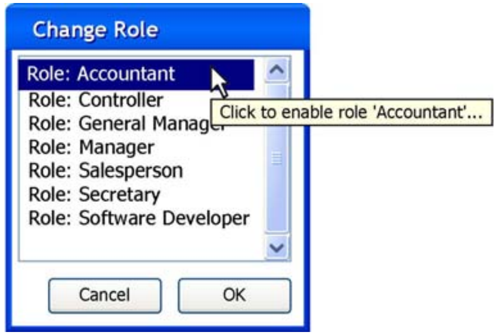
–Use features for limited time period

- Szenario 4: Instant help desk support

–Transmit and replicate user configuration at help desk

Scenario 5: Role Change

In the morning Maria works as an accountant...

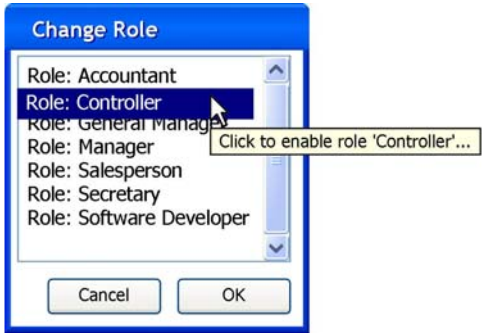


Change Role

Role: Accountant
Role: Controller
Role: General Manag
Role: Manager
Role: Salesperson
Role: Secretary
Role: Software Developer

Click to enable role 'Accountant'...

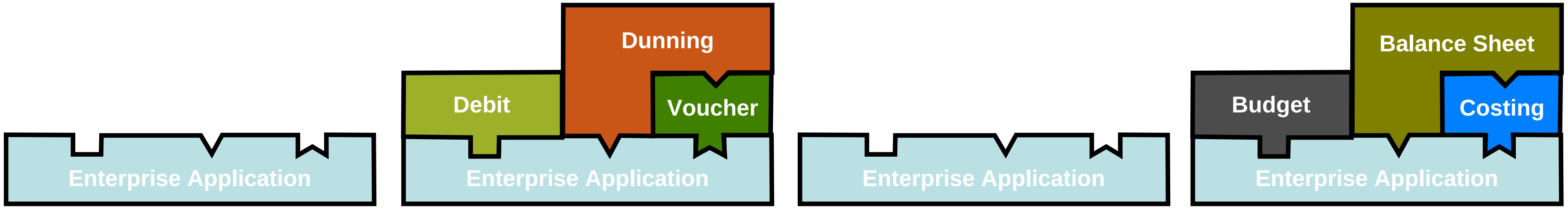
...and in the afternoon she changes her role to controller.



Change Role

Role: Accountant
Role: Controller
Role: General Manag
Role: Manager
Role: Salesperson
Role: Secretary
Role: Software Developer

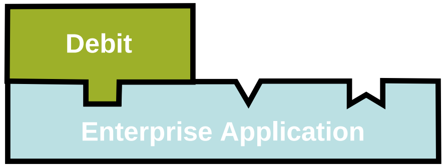
Click to enable role 'Controller'...



Adapt your user interface on the fly matching the current role.
Tailor your application at run time!

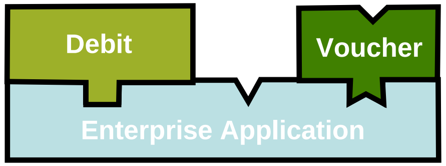
Scenario 6: Optimized Training

Lesson 1: How to book an outgoing invoice



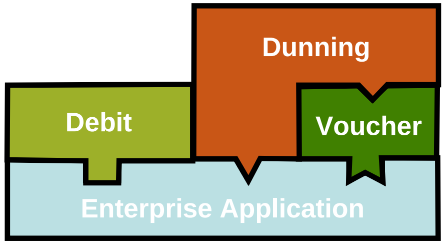
Trainee can focus on content to trained in lesson 1.

Lesson 2: How to determines bills due



Trainee can focus on content to be trained in lesson II and integrate with knowledge from previous lessons.

Lesson 3: How to generate dunning letters



Incrementally build up system alongside with trainee's learning progress.

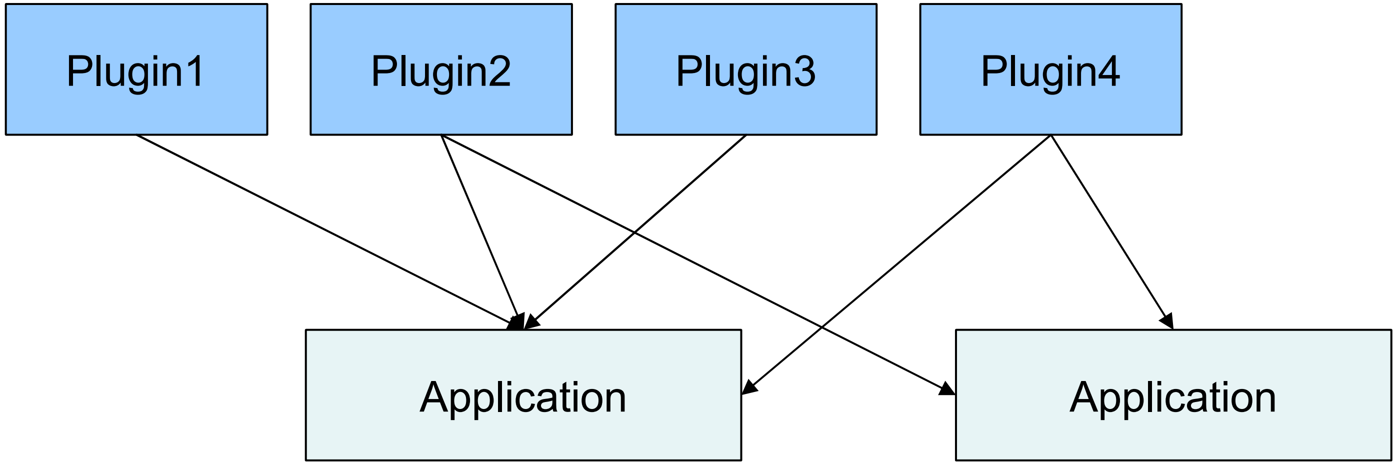
Mozilla Firefox

Mozilla Extensions

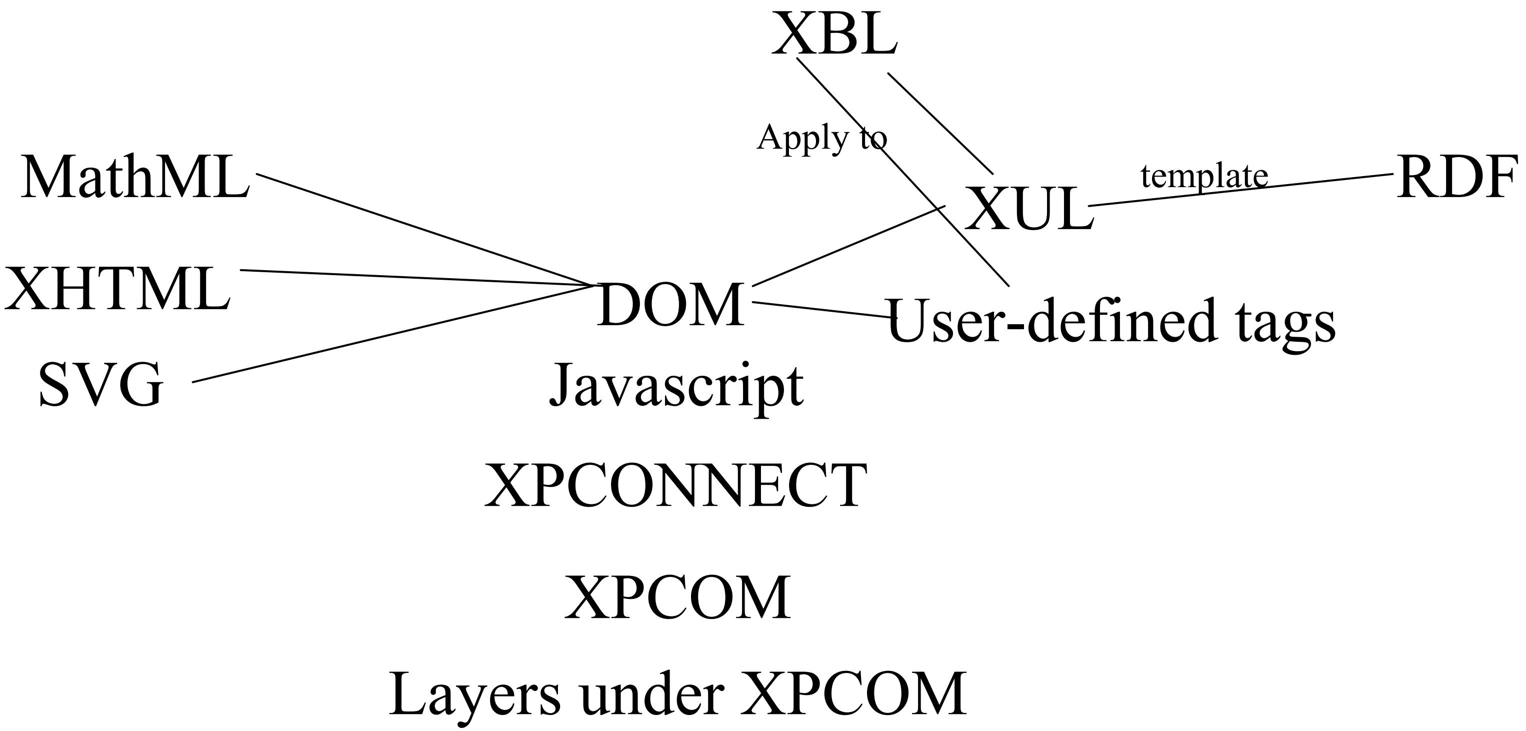
https://developer.mozilla.org/en/Extensions

Flat structure – no dependencies between modules
One module can be plugged into different applications
(e.g. Thunderbird, Firefox, ...)

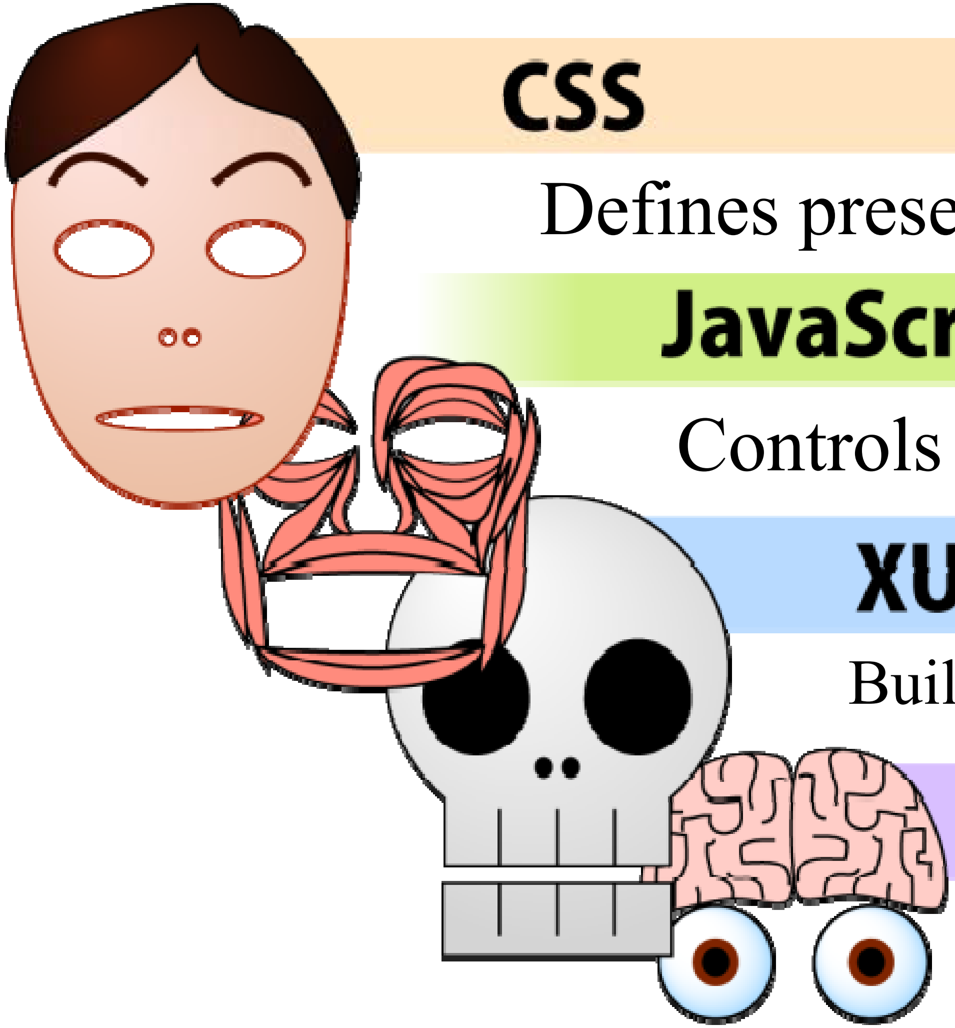
Versioning:
- Plugins have a version number
- Plugins can specify the supported application version numbers



Architecture of Mozilla



Technologies in Use



CSS

Defines presentation

JavaScript

Controls all parts

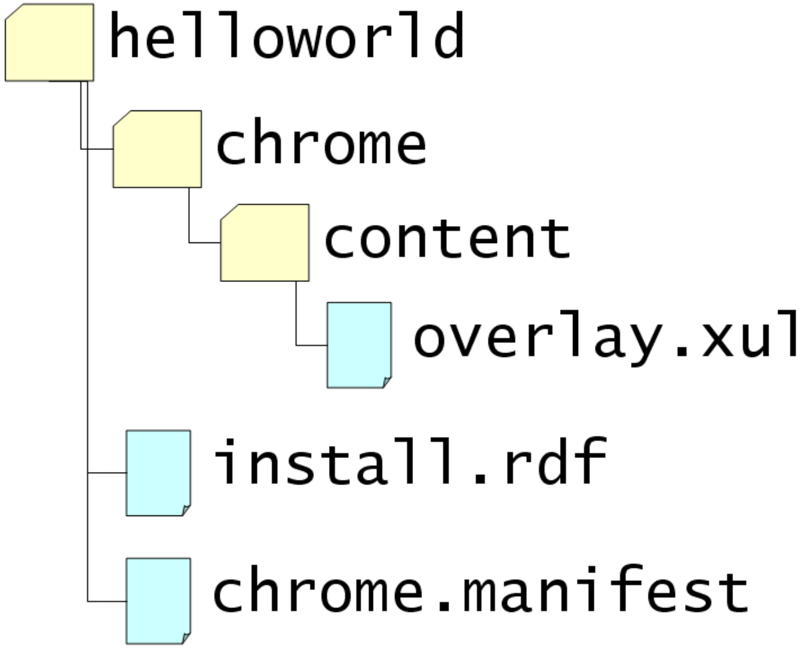
XUL

Builds architectural framework

XPCOM

Black box for specialized tasks

What is required?



overlay.xul

```
<?xml version="1.0" encoding="UTF-8"?>
<overlay id="helloworldOverlay"
xmlns="http://www.mozilla.org/
keymaster/gatekeeper/there.is.only.xul">

  <menupopup id="menu_ToolsPopup">

  <menuitem id="helloworldMenuitem"
    label="Hello, world!"
    insertbefore="sanitizeSeparator"/>

  </menupopup>

</overlay>
```

install.rdf

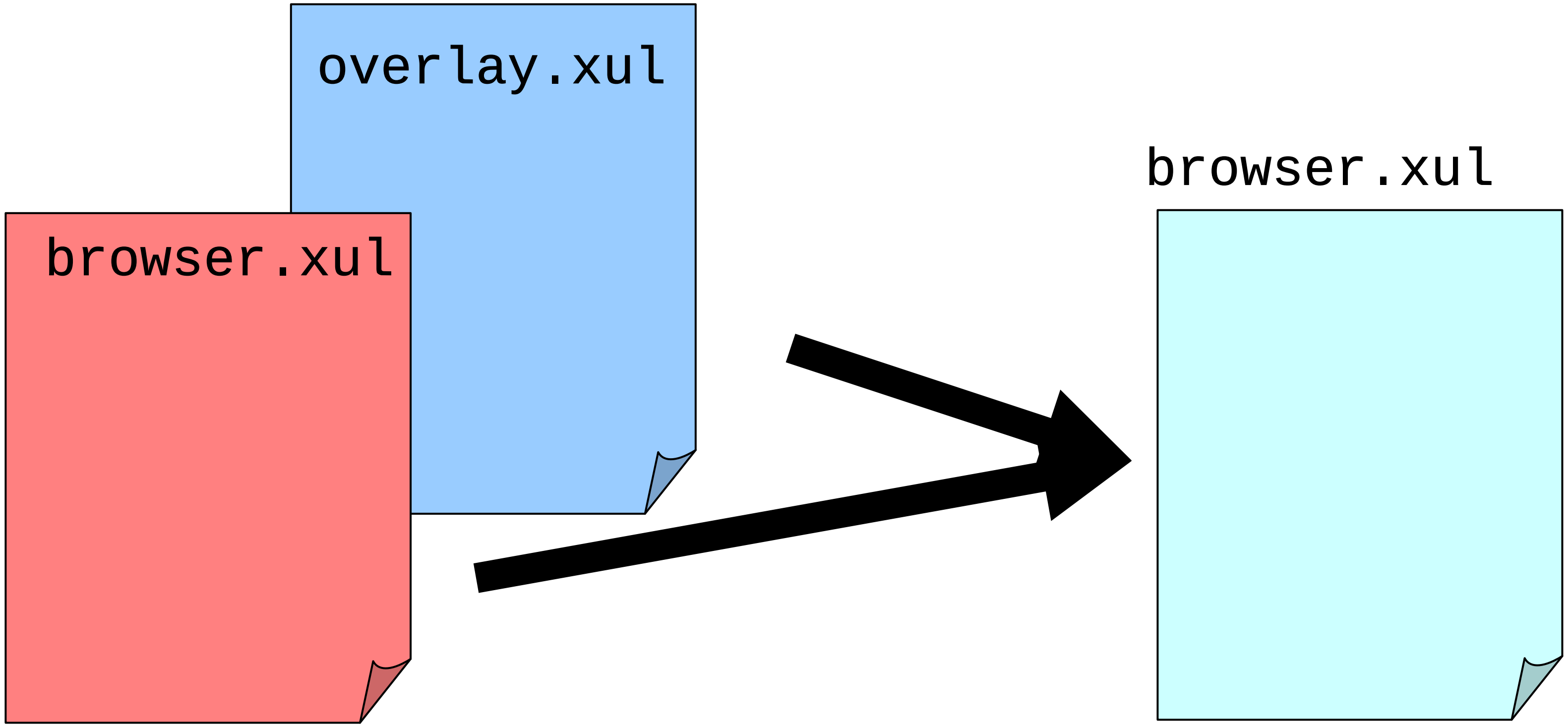
```
<?xml version="1.0" encoding="UTF-8"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">
  <Description about="urn:mozilla:install-manifest">
    <em:id>helloworld@dhungana.at</em:id>
    <em:type>2</em:type>
    <em:name>Hello, world!</em:name>
    <em:version>0.1</em:version>
    <em:description>My first extension.</em:description>
    <em:creator>eDeepak</em:creator>
    <em:homepageURL>http://dhungana.at</em:homepageURL>
    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>2.0</em:minVersion>
        <em:maxVersion>2.0.0.*</em:maxVersion>
    </em:targetApplication>
  </Description>
</RDF>
```

chrome.manifest

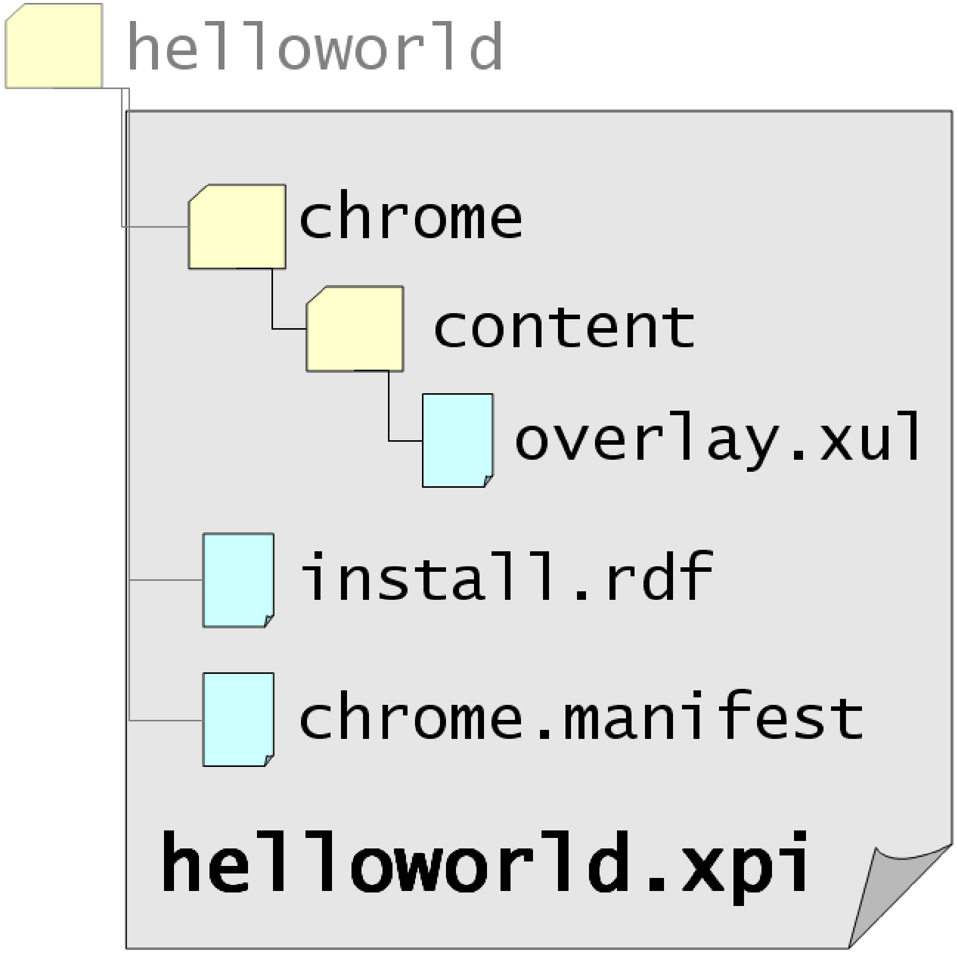
```
content  helloworld  chrome/content/
overlay  chrome://browser/content/browser.xul
      chrome://helloworld/content/overlay.xul
```

chrome.manifest(2)

```
2.overlay  chrome://browser/content/browser.xul
         chrome://helloworld/content/overlay.xul
```



Deployment



helloworld
chrome
content
overlay.xul
install.rdf
chrome.manifest
helloworld.xpi

XUL

- XUL stands for XML User-Interface Language which is a cross-platform language for describing user interfaces of applications.
- Most of the Mozilla applications such as the browser, addressbook, DOM inspector are written in XUL.
- A very good tutorial (go through some examples)

  http://www.xulplanet.com/tutorials/xultu/
- Examples: DOM inspector, a prototype of an equation editor, GeoSVg

```
<?xml version="1.0"?>
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
<window id="mywindow" xmlns=
      "http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
<script>
function test() {
    var myLabel = document.getElementById("myLabel");
    myLabel.value += "!!!";
}
</script>
<label id="myLabel" value="Hello world"/>
<button label="Add !!!" oncommand="test()"/>
</window>
```

XUL (continued)

- RDF is another stepping stone of the Mozilla. It can supply data sources to XUL elements, even SVG elements (an example in SVG from croczilla)
- Datasources of an element can be from a RDF file or internal datasource such as bookmarks, history, and mail messages.

XUL Widgets
http://www.hevanet.com/acorbin/xul/top.xul

Live Editor:
http://ted.mielczarek.org/code/mozilla/xuledit/xuledit.xul

XUL (continued)

- XUL files can be referenced with a regular HTTP URL
- XUL files can also be installed as an XUL package to be an standalone application of an extension of the browser.
- Installed packages are placed under the chrome directory and can be invoked by URL in a form like chrome://inspector/content/inspector.xul
- A package usually has three folders content, skin, and locale

XPCOM

- XPCOM stands for Cross Platform Component Object Model. It's written in C/C++.
- It's an object broking systems like COBRA and COM.
  - An object broker is a piece of code that finds objects and makes them available. If all objects built provide a standard or common interface that the broker can use, then all members of a large set of objects can be handled the same way.
- XPCOM is like a miniature of an OS.
- Applications written in XUL and Javascripts only interact with XPCOM.

XPCOM (continued.)

- Javascripts access objects in XPCOM in a standard way
  - Get a component
  - Get the part of the component that implements the interface that we want to use.
  - Call the function we need
- Example: Get a service for handling local files

```
var aFile =
     Components.classes["@mozilla.org/file/local;1"].createInstance();
    if (aFile) aFile.QueryInterface(Components.interfaces.nsILocalFile);
```

XPCOM

Create a temporary folder by calling XPCOM from JavaScript

```
const nsILocalFile = Components.interfaces.nsILocalFile;
var file = Components.classes['@mozilla.org/file/local;1']
                     .createInstance(nsILocalFile);
file.initWithPath('C:\\');
file.append('temp');
if (!file.exists()) {
  file.create(nsILocalFile.DIRECTORY_TYPE, 0755);
}
```

Mozilla Chrome

https://developer.mozilla.org/en/Chrome_Manifest

Used in combination with XUL for declarative specification of the GUI (e.g. menu items, tool bars, ...)

Chrome List Add-In:

https://addons.mozilla.org/en-US/firefox/addon/4453

Two bugs:

- disable/enable when it gets stuck on startup
- use Properties → Resolved URL if a XUL document cannot be loaded

Chrome Providers

- Content
- Locale
- Skin

Module coordination similar to the NetBeans FileSystem!

- specified in “chome.manifest”
- overlay other XUL files (all overlaid files are merged)
- override other XUL files

DEMO: Extension Wizard

1. Goto Online Extension Skeleton Wizard
   http://ted.mielczarek.org/code/mozilla/extensionwiz/

2. Create an extension and save it to myex.zip

3. Extract the zip file

4. ZIP the contents of the myex directory and name it myex.xpi

5. Open firefox and open this file

6. Install the extension and restart firefox

7. See your newly created menu item in the Tools menu

7. Analyze the generated files in myex.zip

COM

OLE and ActiveX Controls

- Object Linking and Embedding Controls
  - Using Excel object within Word
  - VBX and OCX components
- ActiveX Controls
  - Running applications on the browser
  - HTML-Tag <OBJECT> and <PARAM> Tag

Anwendung der Controls

- benötigen Eintrag in Registry
- Erstellung von .cab Datei
  (z.B. durch Visual Basic 5.0 Setup Wizard)
- Einbindung in HTML:

```
<OBJECT classid="clsid:D75D5AEA-4B9A-11CF-8980-444553540000“
        id=„UserControl1“ codebase="Project1.CAB#version=1,0,0,0">
</OBJECT>
```

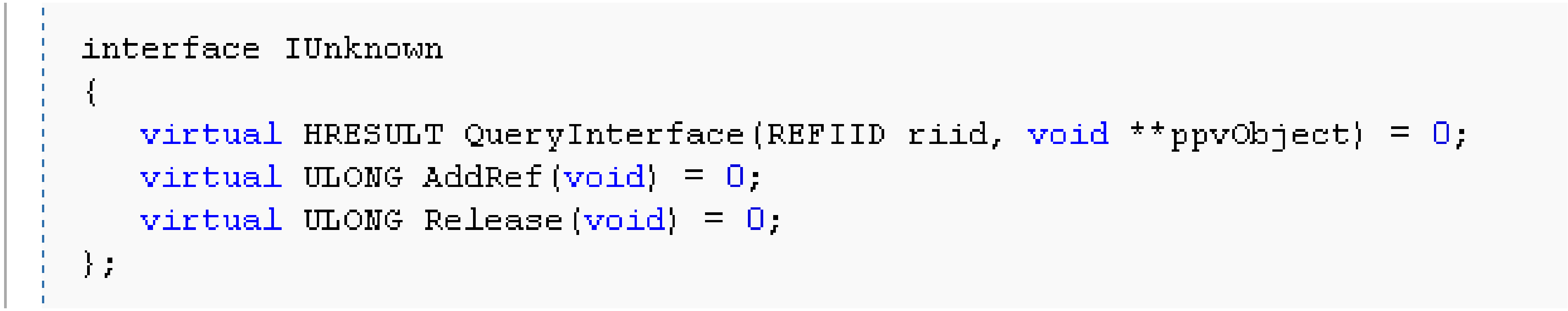
classid:eindeutige Identifizierung
codbase:Position, wo das Control zu finden ist

Component Object Model (COM)

- COM Evolved from OLE and ActiveX.
- COM defines a common way to access data and software services.
- Each COM object supports one or more interfaces.
- Different programs can share data and services using 1 standard.

Basics of COM

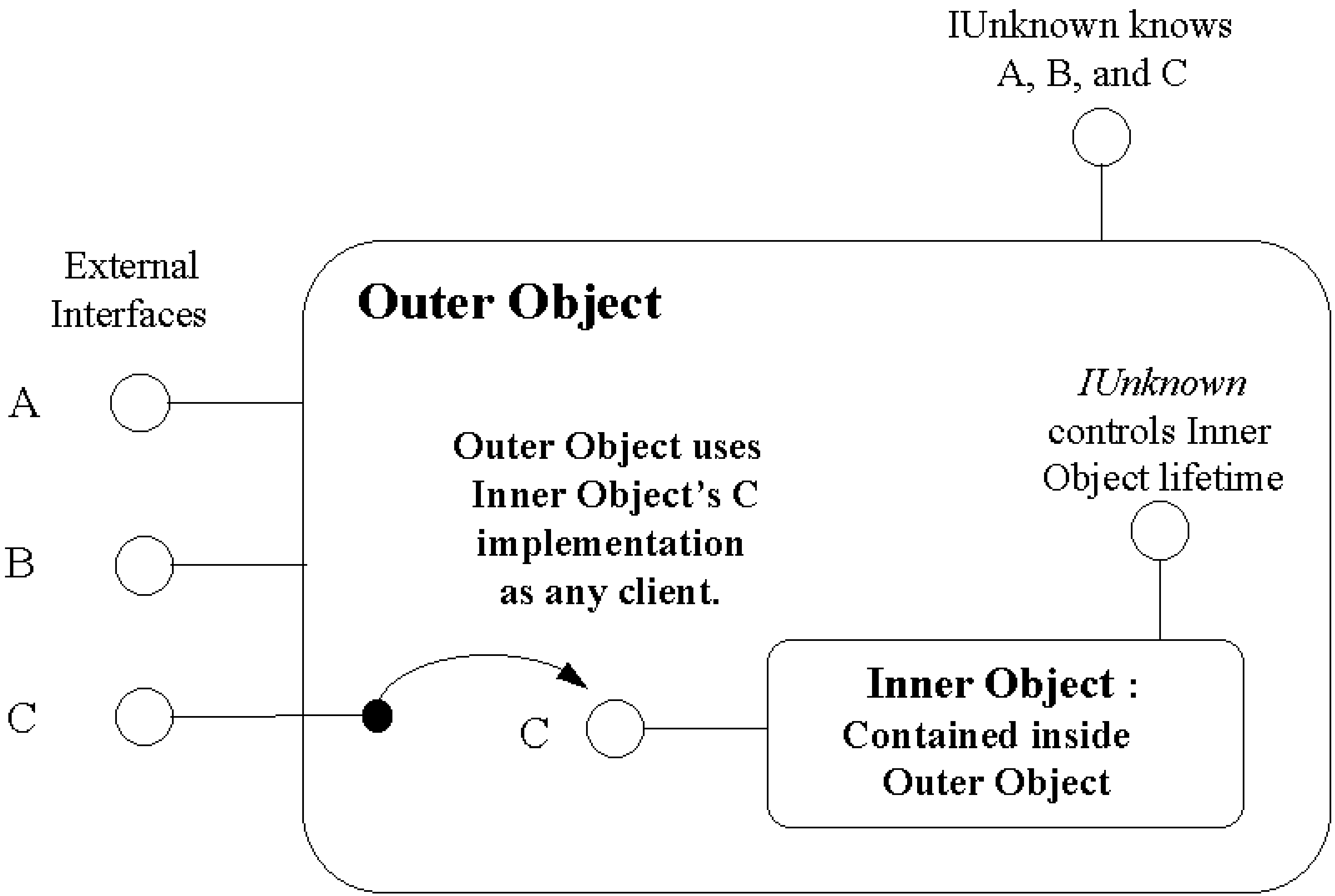
- All COM components must (at the very least) implement the standard IUnknown interface

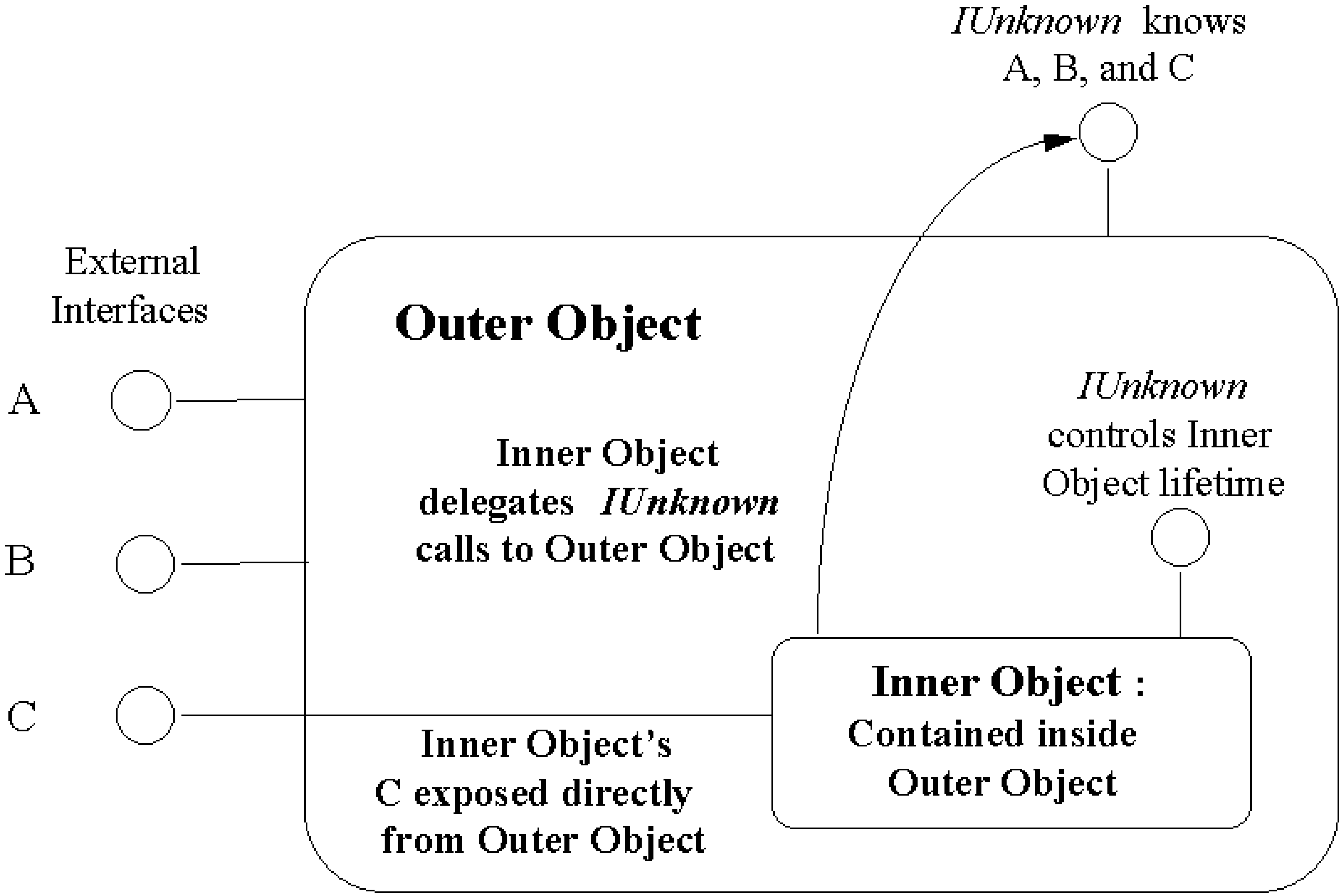


```
interface IUnknown
{
   virtual HRESULT QueryInterface(REFIID riid, void **ppvObject) = 0;
   virtual ULONG AddRef(void) = 0;
   virtual ULONG Release(void) = 0;
};
```

•QueryInterface is used to obtain a pointer to another interface
•AddRef is used by clients to indicate that a COM object is being referenced
•Release is used by clients to indicate that they have finished using the COM object.

QueryInterface() is similar to dynamic cast<> in C++ or casts in Java and C#.


IUnknown knows
A, B, and C
External
Interfaces
A
B
C
Outer Object
IUnknown
controls Inner
Object lifetime
Outer Object uses
Inner Object's C
implementation
as any client.
C
Inner Object :
Contained inside
Outer Object

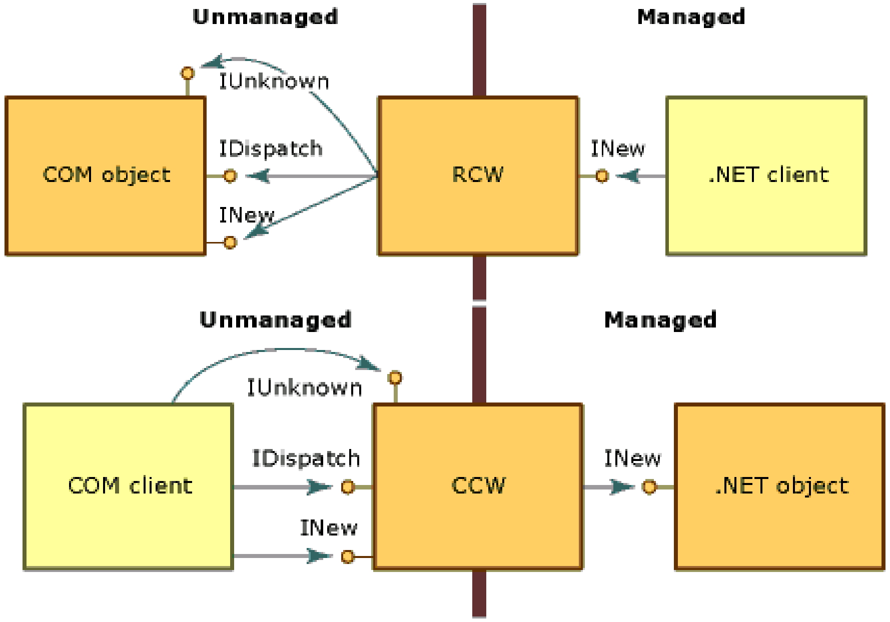

IUnknown knows
A, B, and C
External
Interfaces
A
B
C
Outer Object
Inner Object
delegates *IUnknown*
calls to Outer Object
IUnknown
controls Inner
Object lifetime
Inner Object :
Contained inside
Outer Object
Inner Object's
C exposed directly
from Outer Object

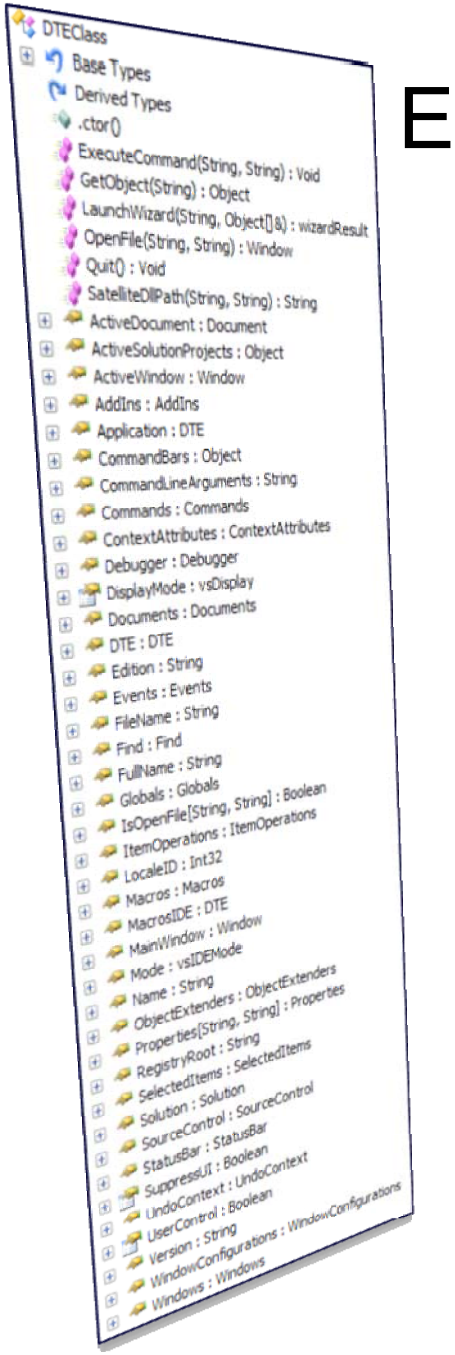
What Is A COM Add-in?

- ActiveX DLL or EXE with special registration
- Standard way of extending Office
  - Across all host applications
  - Available both in user interface and Visual Basic for Applications environment
  - Still need to deal with various object models
- Create with any COM development tool

COM & .NET

- COM components must be registered prior to use with .Net applications (regsvr32.exe).
- .Net Assemblies must be registered for use with COM components (regasm.exe)
- RCW: Runtime Callable Wrappers (INTEROP.*COMLib*)
- TlbImp.exe: Converts COM to .Net Assembly.





```
DTEClass
  Base Types
  Derived Types
  .ctor()
  ExecuteCommand(String, String) : Void
  GetObject(String) : Object
  LaunchWizard(String, Object[]&) : wizardResult
  OpenFile(String, String) : Window
  Quit() : Void
  SatelliteDllPath(String, String) : String
  ActiveDocument : Document
  ActiveSolutionProjects : Object
  ActiveWindow : Window
  AddIns : AddIns
  Application : DTE
  CommandBars : Object
  CommandLineArguments : String
  Commands : Commands
  ContextAttributes : ContextAttributes
  Debugger : Debugger
  DisplayMode : vsDisplay
  Documents : Documents
  DTE : DTE
  Edition : String
  Events : Events
  FileName : String
  Find : Find
  FullName : String
  Globals : Globals
  IsOpenFile[String, String] : Boolean
  ItemOperations : ItemOperations
  LocaleID : Int32
  Macros : Macros
  MacrosIDE : DTE
  MainWindow : Window
  Mode : vsIDEMode
  Name : String
  ObjectExtenders : ObjectExtenders
  Properties[String, String] : Properties
  RegistryRoot : String
  SelectedItems : SelectedItems
  Solution : Solution
  SourceControl : SourceControl
  StatusBar : StatusBar
  SuppressUI : Boolean
  UndoContext : UndoContext
  UserControl : Boolean
  Version : String
  WindowConfigurations : WindowConfigurations
  Windows : Windows
```

Extensibility of Visual Studio

- Macros
  - repeatable tasks and actions
  - developers can record programmatically for saving, replaying, and distributing
  - cannot be used to implement new commands or create tool windows
  - written using Visual Basic and are not compiled
- Templates
- Add-ins
  - plugged in to the IDE via COM
- Packages
  - created using the Visual Studio SDK

Guidance Automation Packages

- A Guidance Package consists of
  - Visual Studio Templates
    - Provide integration with Visual Studio
      - “Create New Project/Item” dialog box
      - Create Solutions, Projects, Project Items, ...
    - Defined in .vstemplate files
  - Guidance Automation Recipes
    - Automated activities that define a series of instructions
      - Abstract an action that the developer would need to do manually
      - E.g. create projects, add references, ...
    - Defined in an xml file
- Link between both: Templates refer to Recipes

Visual Studio Templates

```
<VSTemplate Version="2.0" Type="ProjectGroup"
  xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>Application Block</Name>
    <Description>Guidance Package that creates a new Application Block.</Description>
    <ProjectType>CSharp</ProjectType>
    <Icon>ApplicationBlock.ico</Icon>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <ProjectTemplateLink ProjectName="$ApplicationBlockNamespace$.$ApplicationBlockName$">
        Projects\Runtime\Runtime.vstemplate</ProjectTemplateLink>
    </ProjectCollection>
  </TemplateContent>
  <WizardExtension>
    <Assembly>Microsoft.Practices.RecipeFramework.VisualStudio, Version=1.0.51206.0,
      Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a</Assembly>
    <FullClassName>Microsoft.Practices.RecipeFramework.VisualStudio.Templates.UnfoldTemplate
      </FullClassName>
  </WizardExtension>
  <WizardData>
    <Template xmlns="http://schemas.microsoft.com/pag/gax-template" SchemaVersion="1.0"
      Recipe="CreateApplicationBlock">
    </Template>
  </WizardData>
</VSTemplate>
```

Guidance Automation Recipes

```
<GuidancePackage xmlns="http://schemas.microsoft.com/pag/gax-core"
    Name="JelleDruyts.EnterpriseLibraryGuidance" Caption="Enterprise Library Guidance"
    Description="Provides guidance around the creation of Application Blocks"
    Guid="2cac5b9c-a04f-4a49-8a56-3ee5d63bd83f" SchemaVersion="1.0">
  <Recipes>
    <Recipe Name="CreateApplicationBlock">
      <Caption>Create a new Enterprise Library Application Block</Caption>
      <Arguments>
        <Argument Name="ApplicationBlockName" Required="true">
          <Converter Type="Microsoft.Practices.RecipeFramework.Library.Converters.
            CodeIdentifierStringConverter, Microsoft.Practices.RecipeFramework.Library" />
        </Argument>
        <Argument Name="ApplicationBlockNamespace" Required="true">
          <Converter Type="Microsoft.Practices.RecipeFramework.Library.Converters.
            NamespaceStringConverter, Microsoft.Practices.RecipeFramework.Library" />
        </Argument>
      </Arguments>
      <GatheringServiceData>
        <Wizard xmlns="http://schemas.microsoft.com/pag/gax-wizards" SchemaVersion="1.0">
          <Pages>
            <Page>
              <Title>Application Block Information</Title>
              <Fields>
                <Field ValueName="ApplicationBlockName" Label="Application Block Name"
                  InvalidValueMessage="Must be a valid .NET identifier." />
                <Field ValueName="ApplicationBlockNamespace" Label="Namespace"
                  InvalidValueMessage="Must be a valid .NET namespace identifier." />
              </Fields>
            </Page>
          </Pages>
        </Wizard>
      </GatheringServiceData>
    </Recipe>
  </Recipes>
</GuidancePackage>
```

Addins in Visual Studio

- COM objects
  - Implement the IDTExtensibility2 interface
  - Communicate with the IDE through the core automation object model
- Any COM-consuming language
  - Visual C++, Visual Basic, Visual C#, etc
- Can be used to
  - Host your tool on a menu or toolbar in the IDE.
  - Create custom property pages for the Options dialog box on the Tools menu.
  - Create tool windows that act just like Visual Studio tool windows.
  - Dynamically enable and disable commands on menus and the Visual Studio Command bar.
  - Add contact and descriptive information to the Visual Studio Help About box.

Visual Studio SDK

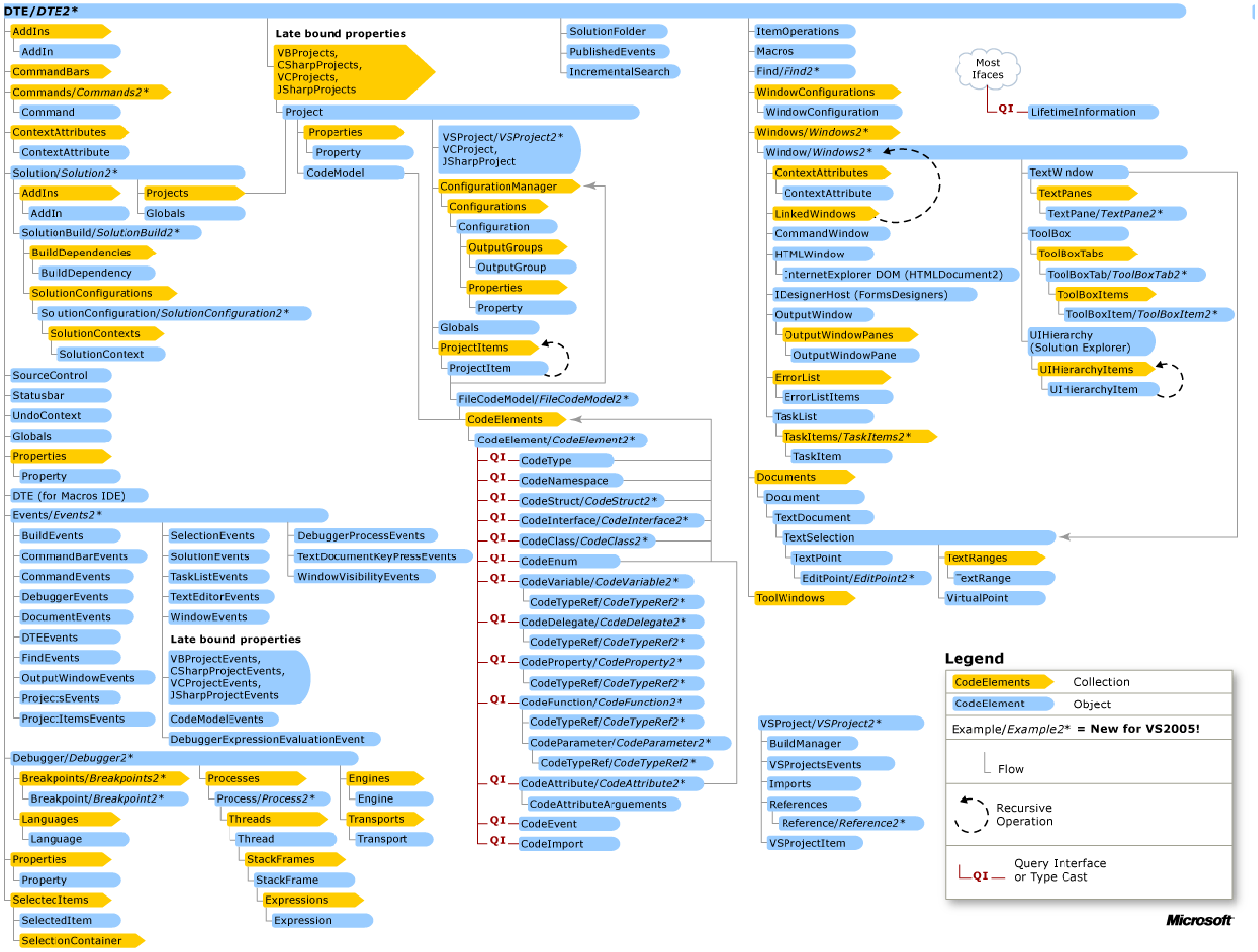
- Visual Studio has an object model
  - Call the Visual Studio API's directly
  - Use EnvDTE.dll and EnvDTE80.dll
- Powerful
  - Entire Visual Studio object model is exposed
- Difficult
  - Registering custom packages in Visual Studio
  - COM interop with EnvDTE object model

```
[uuid(093274850F-F8BA-46B9-
3FDA985CD9A)]
coclass Example
{
   [default] interface IExample;
}
```

```
using System.Runtime.InteropServices;
[Comimport]
[Guid(093274850F-F8BA-46B9-3FDA985CD9A)]
public class Example
{// Hier darf nichts stehen!
}
```

Von der COM-Klasse → Zum CLR-Code (C#)

Visual Studio Automation Object Model



- DTE/*DTE2**
  - AddIns
    - AddIn
  - CommandBars
  - Commands/*Commands2**
    - Command
  - ContextAttributes
    - ContextAttribute
  - Solution/*Solution2**
    - AddIns
      - AddIn
    - Projects
    - Globals
    - SolutionBuild/*SolutionBuild2**
      - BuildDependencies
        - BuildDependency
      - SolutionConfigurations
        - SolutionConfiguration/*SolutionConfiguration2**
          - SolutionContexts
            - SolutionContext
  - SourceControl
  - Statusbar
  - UndoContext
  - Globals
  - Properties
    - Property
  - DTE (for Macros IDE)
  - Events/*Events2**
    - BuildEvents
    - CommandBarEvents
    - CommandEvents
    - DebuggerEvents
    - DocumentEvents
    - DTEEvents
    - FindEvents
    - OutputWindowEvents
    - ProjectsEvents
    - ProjectItemsEvents
    - SelectionEvents
    - SolutionEvents
    - TaskListEvents
    - TextEditorEvents
    - WindowEvents
    - **Late bound properties**
      - VBProjectEvents, CSharpProjectEvents, VCProjectEvents, JSharpProjectEvents
    - CodeModelEvents
    - DebuggerExpressionEvaluationEvent
    - DebuggerProcessEvents
    - TextDocumentKeyPressEvents
    - WindowVisibilityEvents
  - Debugger/*Debugger2**
    - Breakpoints/*Breakpoints2**
      - Breakpoint/*Breakpoint2**
    - Languages
      - Language
    - Processes
      - Process/*Process2**
        - Threads
          - Thread
            - StackFrames
              - StackFrame
                - Expressions
                  - Expression
    - Engines
      - Engine
    - Transports
      - Transport
  - Properties
    - Property
  - SelectedItems
    - SelectedItem
    - SelectionContainer
  - **Late bound properties**
    - VBProjects, CSharpProjects, VCProjects, JSharpProjects
      - Project
        - Properties
          - Property
        - CodeModel
        - VSProject/*VSProject2**, VCProject, JSharpProject
        - ConfigurationManager
          - Configurations
            - Configuration
              - OutputGroups
                - OutputGroup
              - Properties
                - Property
        - Globals
        - ProjectItems
          - ProjectItem
            - FileCodeModel/*FileCodeModel2**
              - CodeElements
                - CodeElement/*CodeElement2**
                  - QI CodeType
                  - QI CodeNamespace
                  - QI CodeStruct/*CodeStruct2**
                  - QI CodeInterface/*CodeInterface2**
                  - QI CodeClass/*CodeClass2**
                  - QI CodeEnum
                  - QI CodeVariable/*CodeVariable2**
                    - CodeTypeRef/*CodeTypeRef2**
                  - QI CodeDelegate/*CodeDelegate2**
                    - CodeTypeRef/*CodeTypeRef2**
                  - QI CodeProperty/*CodeProperty2**
                    - CodeTypeRef/*CodeTypeRef2**
                  - QI CodeFunction/*CodeFunction2**
                    - CodeTypeRef/*CodeTypeRef2**
                    - CodeParameter/*CodeParameter2**
                      - CodeTypeRef/*CodeTypeRef2**
                  - QI CodeAttribute/*CodeAttribute2**
                    - CodeAttributeArguements
                  - QI CodeEvent
                  - QI CodeImport
  - SolutionFolder
  - PublishedEvents
  - IncrementalSearch
  - ItemOperations
  - Macros
  - Find/*Find2**
  - WindowConfigurations
    - WindowConfiguration
  - Windows/*Windows2**
    - Window/*Windows2**
      - ContextAttributes
        - ContextAttribute
      - LinkedWindows
      - CommandWindow
      - HTMLWindow
        - InternetExplorer DOM (HTMLDocument2)
      - IDesignerHost (FormsDesigners)
      - OutputWindow
        - OutputWindowPanes
          - OutputWindowPane
      - ErrorList
        - ErrorListItems
      - TaskList
        - TaskItems/*TaskItems2**
          - TaskItem
      - TextWindow
        - TextPanes
          - TextPane/*TextPane2**
      - ToolBox
        - ToolBoxTabs
          - ToolBoxTab/*ToolBoxTab2**
            - ToolBoxItems
              - ToolBoxItem/*ToolBoxItem2**
      - UIHierarchy (Solution Explorer)
        - UIHierarchyItems
          - UIHierarchyItem
  - Documents
    - Document
      - TextDocument
        - TextSelection
          - TextPoint
            - EditPoint/*EditPoint2**
          - TextRanges
            - TextRange
          - VirtualPoint
  - ToolWindows

Most Ifaces
- QI LifetimeInformation

VSProject/*VSProject2**
- BuildManager
- VSProjectsEvents
- Imports
- References
  - Reference/*Reference2**
- VSProjectItem

Legend

CodeElements	Collection
CodeElement	Object

Example/*Example2** = **New for VS2005!**

Flow

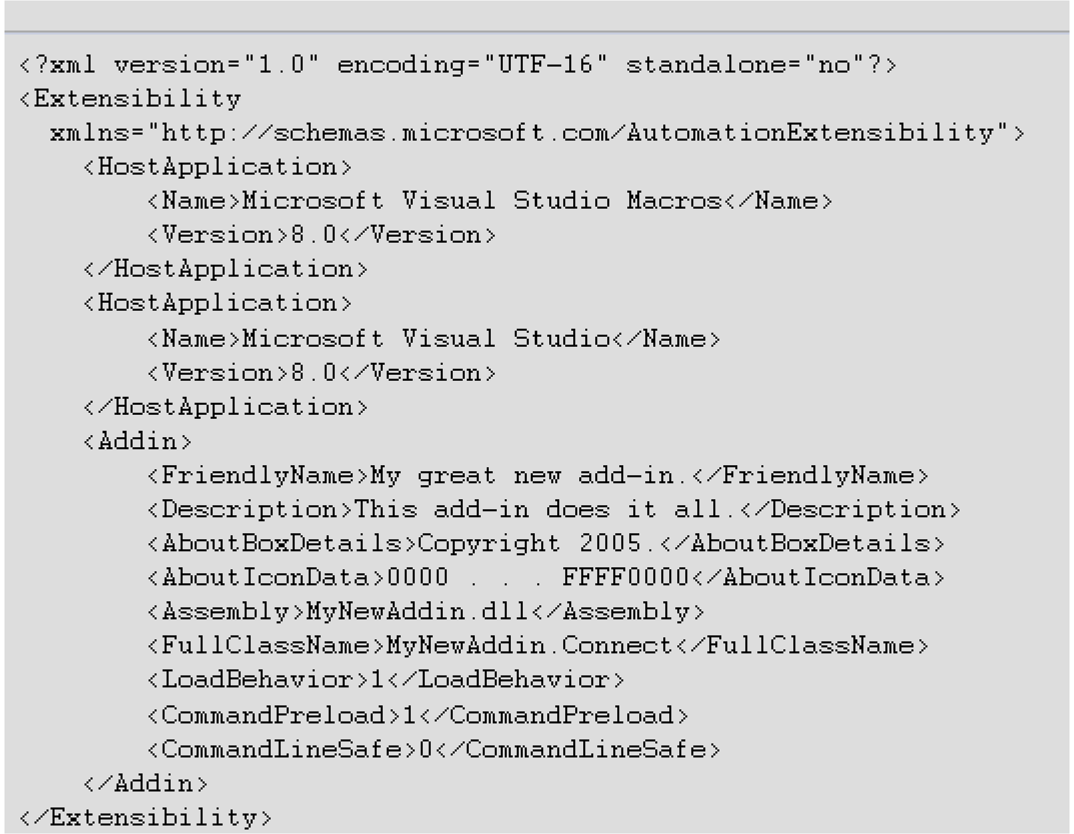
Recursive Operation

QI — Query Interface or Type Cast

Microsoft

Registering Addins

- To make Addins available for activation in the Add-In Manager
- Through a Addin XML file



```
<?xml version="1.0" encoding="UTF-16" standalone="no"?>
<Extensibility
  xmlns="http://schemas.microsoft.com/AutomationExtensibility">
    <HostApplication>
        <Name>Microsoft Visual Studio Macros</Name>
        <Version>8.0</Version>
    </HostApplication>
    <HostApplication>
        <Name>Microsoft Visual Studio</Name>
        <Version>8.0</Version>
    </HostApplication>
    <Addin>
        <FriendlyName>My great new add-in.</FriendlyName>
        <Description>This add-in does it all.</Description>
        <AboutBoxDetails>Copyright 2005.</AboutBoxDetails>
        <AboutIconData>0000 . . . FFFF0000</AboutIconData>
        <Assembly>MyNewAddin.dll</Assembly>
        <FullClassName>MyNewAddin.Connect</FullClassName>
        <LoadBehavior>1</LoadBehavior>
        <CommandPreload>1</CommandPreload>
        <CommandLineSafe>0</CommandLineSafe>
    </Addin>
</Extensibility>
```

Discussion!