



The Seven Habits of Highly Effective Technology Disruption

Python Survival in a Java & .Net World

Dana Moore
Senior Scientist
Damoore@bbn.com



Our Goal Today

Learn habits for extending the reach of Python and Jython despite management's best intentions to the contrary

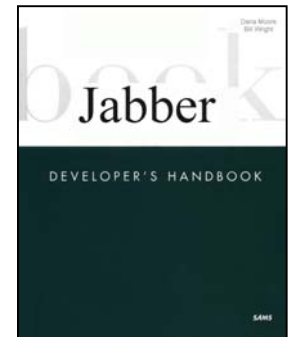
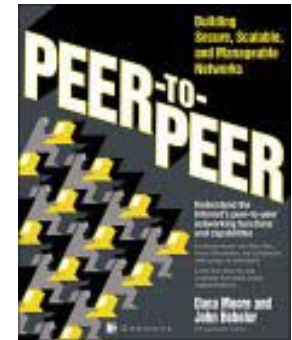
Learning Objectives

- As a result of this presentation, you will be able to:
 - Understand features and strengths of the most commonly used scripting languages
 - Better announce the reasons for preferring runtime-typed LWLs
 - Find logical points of adoption and preemption for exploitation as a Python developer



Dana Moore

- Senior Engineer at BBN Technologies
- Speaks frequently on software agent technologies and P2P topics
- Published *Peer to Peer: Building Scalable, Manageable, Secure Networks* (McGraw-Hill, 2002)
- Publishing *Jabber Developer Handbook* (SAMS, June 2003)
- Has written articles on various topics for
 - *IEEE Intelligent systems*
 - *IEEE Internet Computing*
 - *Journal of Object-Oriented Programming*
 - *Advanced Design Techniques*



Scripting is Higher Level Programming for the 21st Century

1. Component-based design best current practice development methodology
 2. Java is the right (i.e., best) way to *create* components.
 3. Scripting Languages are the right (i.e., best) way to *integrate* components
 4. Programmers can write roughly the same number of lines of code per year regardless of language [1]
- system programming languages allow applications to be written more quickly than assembly language
 - scripting languages allow applications to be integrated more quickly than system programming languages

Agenda

- Evolution of applications: from monoliths to components
- Scripting Language Comparison
 - (Python, TCL, Ruby, JavaScript)
- The Seven Habits
- Demo(s)



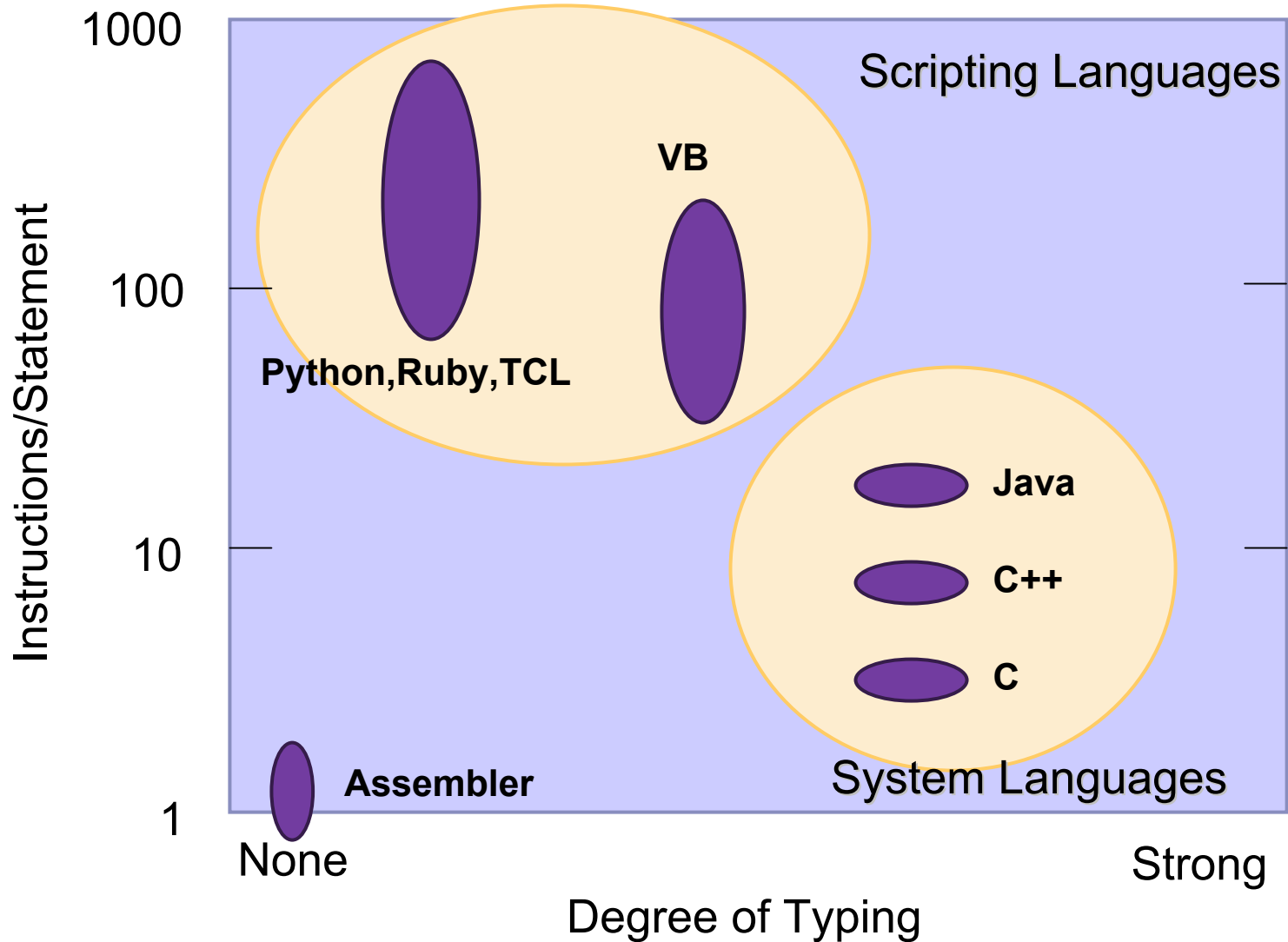
Why You Will Care

- Python and other runtime-typed languages will come under strong assault, especially from C# and the .Net common runtime library (CLR), which will receive the benefit of considerable largess and promotion on the part of Microsoft.
- It is important to begin working a strategy for survival and growth.
- This short talk suggests possible approaches

Habit 1: Know Thine Enemy and Thyself

- Know our own strengths and weaknesses and be able categorise the adversary
- We are faster off the mark, they pay a much higher price in planning and refactoring
- We are faster to react to dynamic requirement change and they are slower and heavier in such domains as
 - scientific analysis,
 - research programming,
 - web services, distributed systems
 - extreme programming environment.
- Their edit-text-crash-debug cycle is more tedious

Habit 1: Understanding Language Levels



Habit 1: Features of popular scripting languages.

Features		Tcl	Perl	Python	Java Script	Ruby	Visual Basic
Speed of use	Rapid development	✓	✓	✓	✓	✓	✓
	Flexible, rapid evolution	✓	✓	✓	✓	✓	✓
	Great regular expressions	✓	✓	✓		✓	
Breadth of functionality	Easily extensible	✓		✓		✓	✓
	Embeddable	✓	✓	✓		✓	
	Easy GUIs	✓		✓		✓	✓
	Internet and Web-enabled	✓	✓	✓	✓	✓	✓
Enterprise usage	Cross platform	✓	✓	✓	✓	✓	
	Internationalization support	✓		✓	✓		✓
	Thread safe	✓		✓			✓
	Database access	✓	✓	✓	✓	✓	✓

Habit 1: Span of Application Space

(simple → complex)



Little languages

Dynamic Runtime-typed languages

Strongly-typed languages

- `grep "spam" *.txt`
- `find /docs -name "*.txt" -print -exec grep "spam" {} \;`
- Web Services, distributed apps, software agents
- Web-Server farms in Microsoft Networking environment
- Intelligence Analysis

Leverage Habit 1

Promise management is that we will only use Python when:

- The objective of the application is to *integrate* and *coordinate* a set of existing components or applications.
- It needs a simple GUI.
- The application does a lot of string processing.
 - Especially XML, HTML scraping, Regular expressions
- The functionality of the application will evolve rapidly over time.
- It must be easy to extend and customize the application in the field.
- The application must run on a diverse set of platforms.

Leverage Habit 1

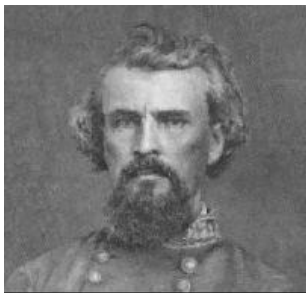
We will willingly abdicate problem domains where:

- The application implements complex algorithms or data structures.
- Execution speed is critical (e.g. the application must frequently scan datasets with tens of thousands of elements).
- The functions of the application are well-defined and slow to change.

Habit 2: Get There the Fastest with the Mostest

Rapid prototyping is a universally acknowledged forte of Python. You can use this knowledge plus human nature in the following way:

- Assure management you will only use Python to model the application until requirements “settle out.”
 - Design a little, implement a little, build a little, test a little, demo a little



Gen. Nathan Bedford Forrest, CSA
Forrest once summed up his military theory as "Get there first with the most men."

Habit 2: Get There the Fastest with the Mostest

- Caveat: Technical brilliance will never shine thru the clouds of management cluelessness

```
Python 2.2.2 (#37, Oct 14 2002, 17:02:34) [MSC 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> one_nattily_dressed_salesperson = 1000.0
>>> one_hundred_brilliant_developers= 1.0
>>> if one_nattily_dressed_salesperson > one_hundred_brilliant_developers:
...     print "Sad, but true!"
...
Sad, but true!
>>>
```

Leverage Habit 2

Success Needs No Justification

Remember the cardinal rule:

“It is far easier to ask for forgiveness
than to beg for permission”



-- Rear Admiral Grace Murray Hopper, USN

Habit 3: Fill The Gaps

- Java believes: “Run on any platform,” but doesn’t play well with other languages

- Raise your hand if you enjoy JNI



- MS believes “Any language, but it must be my operating system”



- Raise your hand if you’ve ever thought about implementing ActiveX on linux

- These are what we call “gaps,” folks

Leverage habit 3: Using Jython

- 100% Java port of C-Python Interpreter. Python:
 - clean syntax
 - supports a Supports modules, classes, packages
 - full object-oriented programming model which makes it a natural companion for Java
- Embedded scripting – ability to add flexible logic to Java apps
- Control of Java components from scripting context
- RAD support
- Rich libraries (web services, graphics, regular expression, etc.)
- But... Jython doesn't support Win32 extensions directly

Control Jython from Java

```
1 import org.python.util.PythonInterpreter;
2 import org.python.core.*;

3 public class Klass {
4     public static void main(String []args) throws PyException
5     {
6         PythonInterpreter interp = new PythonInterpreter();
7         interp.exec("import sys");
8         interp.exec("print sys")

9         interp.set("a", new PyInteger(42));
10        interp.exec("print a");
11        interp.exec("x = 2+2");
12        PyObject x = interp.get("x");
13        System.out.println("x: "+x);
14    }
15 }
```



Demo

Accessing COM from Java

Your Assignment: Access a COM Server from Java



Mission Impossible?



Java Classes

Jython



XML-RPC



MSAgent.dll

CPython



Jython Side (Client)

```
1 import xmlrpclib
2 from java import awt
3
4 class Marionette:
5     def __init__(self):
6         self.agent =
7             xmlrpclib.Server('http://localhost:8000')
8         print self.agent.initAgent()
9         print self.agent.startAgent()
10
11     def saySomething(self, anExpression):
12         self.agent.say(anExpression)
13     def doSomething(self, anAction):
14         self.agent.play(anAction)
15     def hideMe(self):
16         self.agent.hide()
```

Jython Side (Client, Cont'd)

```
17 if __name__ == '__main__':
18     def animate(e):
19         animation.caretPosition=0
20         puppet.doSomething(animation.getText())
22     def speak(e):
23         speech.caretPosition=0
24         puppet.saySomething(speech.getText())
25     puppet = Marionette()
26     panel0= awt.Panel(layout=awt.BorderLayout())
27     panel1= awt.Panel(layout=awt.GridLayout())
28
29     label0 = awt.Label('Animation:')
30     animation = awt.TextField(text='Greet', actionPerformed=animate)
31
32     label1 = awt.Label("Say This:")
33     speech = awt.TextField(text='Hello Folks', actionPerformed=speak)
34
35     panel1.add(label0)
36     panel1.add(animation)
37     panel1.add(label1)
38     panel1.add(speech)
39
40     panel0.add(panel1, 'Center')
41
42     import pawt
43     pawt.test(panel0, size=(300,56))
44     animate(None)
```



C-Python Side (the Server)

```
5 class Anthropomorph:
6     def __init__ (self, agent="Merlin"):
7         self.agentName=agent
8     def initAgent(self):
9         self.dispatchID = Dispatch("Agent.Control.2") # load control
10        self.dispatchID.Connected=1 # Connect!
11        fullName = self.agentName+'.acs'
12        self.dispatchID.Characters.Load(self.agentName, fullName)
13        self.anAgent = self.dispatchID.Characters(self.agentName)
14        return "Anthropomorph: initAgent done."
15    def startAgent (self):
16        self.anAgent.Show()
17        self.anAgent.Play("Greet")
18        self.anAgent.MoveTo(820,600,1000)
19        time.sleep(5) # padding so that obj can perform
20        return "Anthropomorph: startAgent done."
21    def play(self, gesture, delay=5):
22        self.anAgent.Play(gesture)
23        time.sleep(delay) # padding so that obj can perform
24        return delay
25
26    def say (self, phrase, delay=5):
27        self.anAgent.Speak(phrase)
28        time.sleep(delay)
29        return delay
30
31    def hide(self, delay=0):
32        time.sleep(delay)
33        self.anAgent.Hide()
34        return delay
```



C-Python Side (the Server)

```
1 from win32com.client import *
2 import SimpleXMLRPCServer
3 import os,sys,time,string
  .
  .
  .

34 server =
    SimpleXMLRPCServer.SimpleXMLRPCServer
    ("localhost", 8000)
35 server.register_instance(Anthropomorph())
36 print 'SimpleXMLRPCServer @ localhost:8000'
37 server.serve_forever()
```



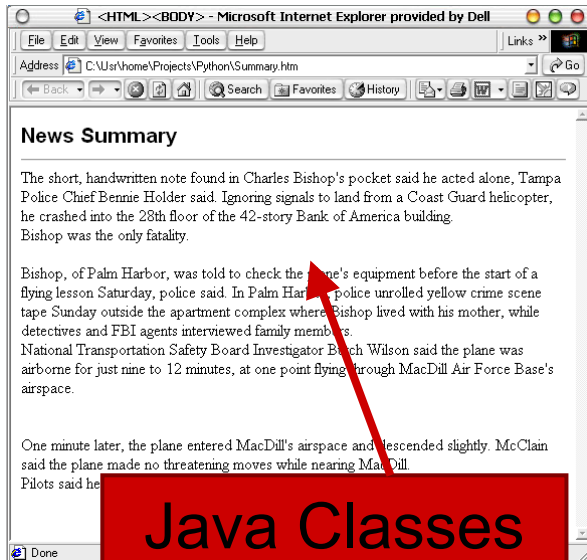
Demo

Controlling MSWord from Java

Your Assignment: Summarize a MS Word Doc and present in a JFC HTML renderer



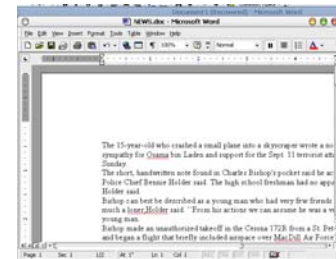
Mission Impossible?



Java Classes

Jython

SOAPLib



Word.dll

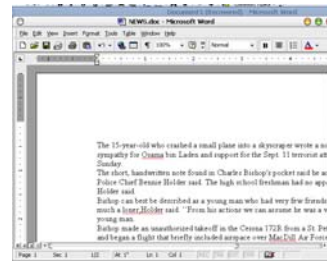
CPython

Relevant Client Side Code

The C-Python MSWord Summariser:

Class Summariser:

```
. . .  
0 def getSummary(self):  
1     word =  
2         win32com.client.dynamic.Dispatch("Word.Application")  
3         word.Visible = 0  
4         doc = word.Documents.Add(wordFile)  
5         doc.Range().InsertAfter(someText) # some text from file  
6         summary = doc.AutoSummarize(Length=24, Mode=2)  
7         doc.Close(SaveChanges = 0)  
8         word.Quit()  
9         # 'summary' is a Word.Range object. Return its string  
10        return summary.Text  
11  
12 . . .
```



Habit 3: Other Kinds of Gaps

- Design, development, configuration management, testing, and functional assessment all have gaps
- Interesting places in the development cycle:
 - Integration and build
 - Use case and engineering test
 - Driving functional assessment
- General rule: look for areas where there are platform issues and incompatibilities and demonstrate Python's ability to work across platforms and APIs

Habit 4: Practice Splendid Isolation

Using CPython and Jython it's easy to drive other objects as service components.

- Allows you to move to centre of the action rather than on the periphery.
- Start by showing that you can script Java with Jython for unit testing,
- End by showing that you can drive an entire distributed architecture with Zope, Twisted, XML-RPC, or SOAPpy.



Habit 5: Live Close to the Metal

One criticism often levelled at runtime interpreted languages is that they are “slow.” Is this true?

- Many of CPython’s core capabilities are implemented in C compiled for a given platform
- You can run ‘closer to the metal’ than a purely bytecode interpreted language
 - and yet not lose the elegance and expressiveness of an O-O language.
 - you can have the ease and rapid payback of shell scripting when you want to test a concept or a procedural approach + encapsulation, inheritance and expressiveness of O-O when you’re ready for it.

Habit 6: Replacement Therapy

Now that you have management attention...

- Look for ways to substitute equivalent Python libraries for those grown-fat Java libraries and bloated virtual machine, especially
 - GUI apps (vs JFC)
 - Distributed services (vs RMI, JavaBeans, J2EE, Java Servlets, .Net CLR)
 - Textual Processing

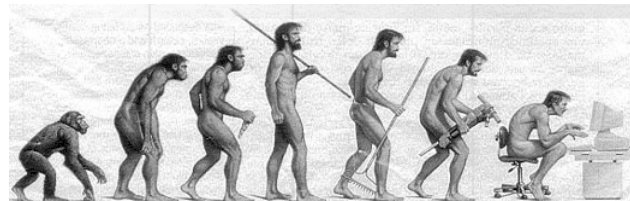
Habit 7: Aim for Disruption, Not Improvement

Horse breeders and buggy makers never designed or delivered a single automobile.

- While the mainstream is focused on optimising solutions to yesterday's problems, identify and solve tomorrow's:
 - Portability, mobility, P2P, “Do what I mean”, *real* software intelligence, highly scalable, highly survivable, highly secure, *ad hoc* networking, etc... These are all still unsolved problems

Summary

- Scripting languages, especially O-O languages, are an excellent platform for adding dynamism to Java
- Excellent infrastructure for web services in several of the languages:
 - Twisted and Zope – Python
 - AOLServer - TCL
 - WebBrick - Ruby
- Natural next step in the evolution of application design and delivery



If You Only Remember One Thing...

“Although component-based design is widely accepted as a development methodology, it is not as well known that scripting languages are essential to this approach. System programming languages such as C, C++, and Java provide excellent tools for creating components, but they are not well-suited to integrating and extending components: their compiled nature and strong typing make them too static and inflexible for component integration. In contrast, scripting languages are interpreted and weakly typed. This makes them much more flexible (an essential attribute when combining components that weren't originally designed for each other) and provides dramatically faster development and evolution of applications.”

-- John K. Ousterhout



python:
software foundation

PyCon
DC 2003

Q&A



The Seven Habits of Highly Effective Technology Disruption

Python Survival in a Java & .Net World

Dana Moore
Senior Scientist
Damoore@bbn.com

