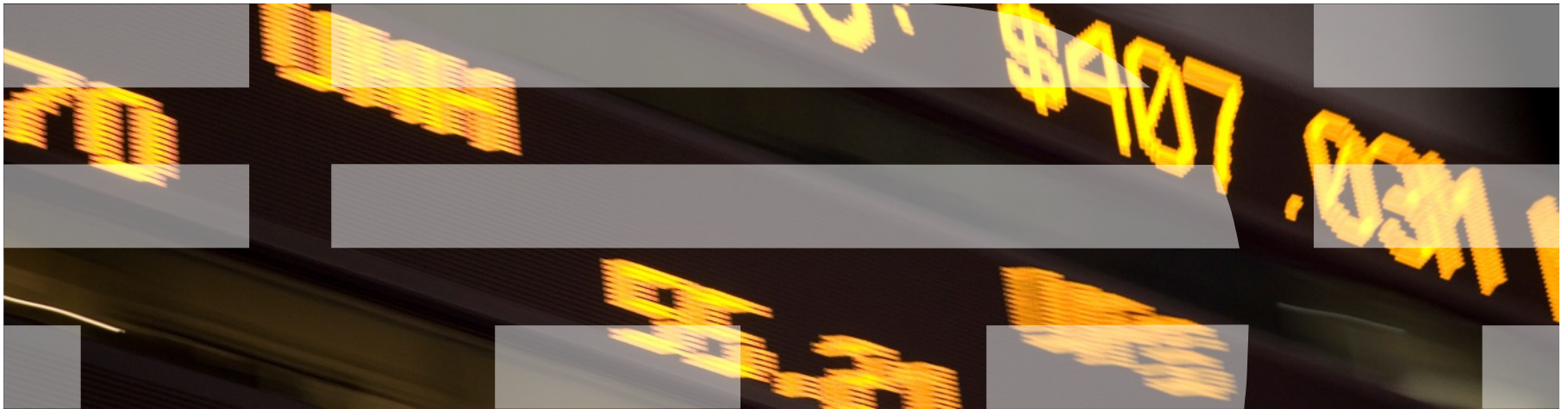


Multi-Platform Application Development

The Nimble Programmer

SHARE Session 8657



Abstract

In today's IT environments and into the foreseeable future, enterprise applications are multi-platform by design and implementation. The days of an application being composed on instructions that are run on only one platform, implemented in only one language, are gone forever.

Application programmers now and into the future must be able to move from platform to platform and language to language with ease and effectiveness. This is required in order to design, develop, debug, and maintain the rich applications which are available today and being enhanced in the future.

IBM's collection of application development tools enable application development teams to collaborate on designs, implementations, testing, and maintenance of these complex, multi-platform applications. Come and learn how to use these product features to support multi-platform application design, development, test, and support.

Application programmers can use common tools to work across multiple platforms, multiple runtime environments, and multiple programming languages. By using these common tools, teams are more productive, more efficient, and more accurate even as the environments within which they develop become more complex.

Agenda

- The Changing Application Landscape
- Multi-platform applications
- Past – Dedicated development teams
- Future – Nimble teams
- Tools Support the team
- Summary

Agenda

- The Changing Application Landscape
- Multi-platform applications
- Past – Dedicated development teams
- Future – Nimble teams
- Tools Support the team
- Summary

The Changing Application Landscape

- Applications today are a mix of technologies
 - Back-end servers
 - Database technologies
 - Mid-tier application servers
 - Multiple user interface types
 - Mash-ups of information from multiple sources
- Implemented in a variety of different programming languages
 - COBOL
 - PL/I
 - C/C++
 - Java
 - HTML
 - javascript
 - Perl
 - PHP ... and the list goes on and on

Applications seem to be user-interface heavy ... but not really

- There is a large emphasis on user interfaces
 - Multitude of device types
 - Old-school laptops/desktops
 - Mobile devices
 - Tablet computers
 - Fit-for-purpose machines (printers with touch-screens, desktop information stations, kiosks, ATMs, and so on)
 - Multiple ways of coalescing information
 - Collaboration sites (LotusLive, Facebook, MySpace, iGoogle)
 - News feeds
 - E-Mail
 - Instant Message
- And yet there is an increased attention to programmatic access to services
 - SOA
 - RESTful services

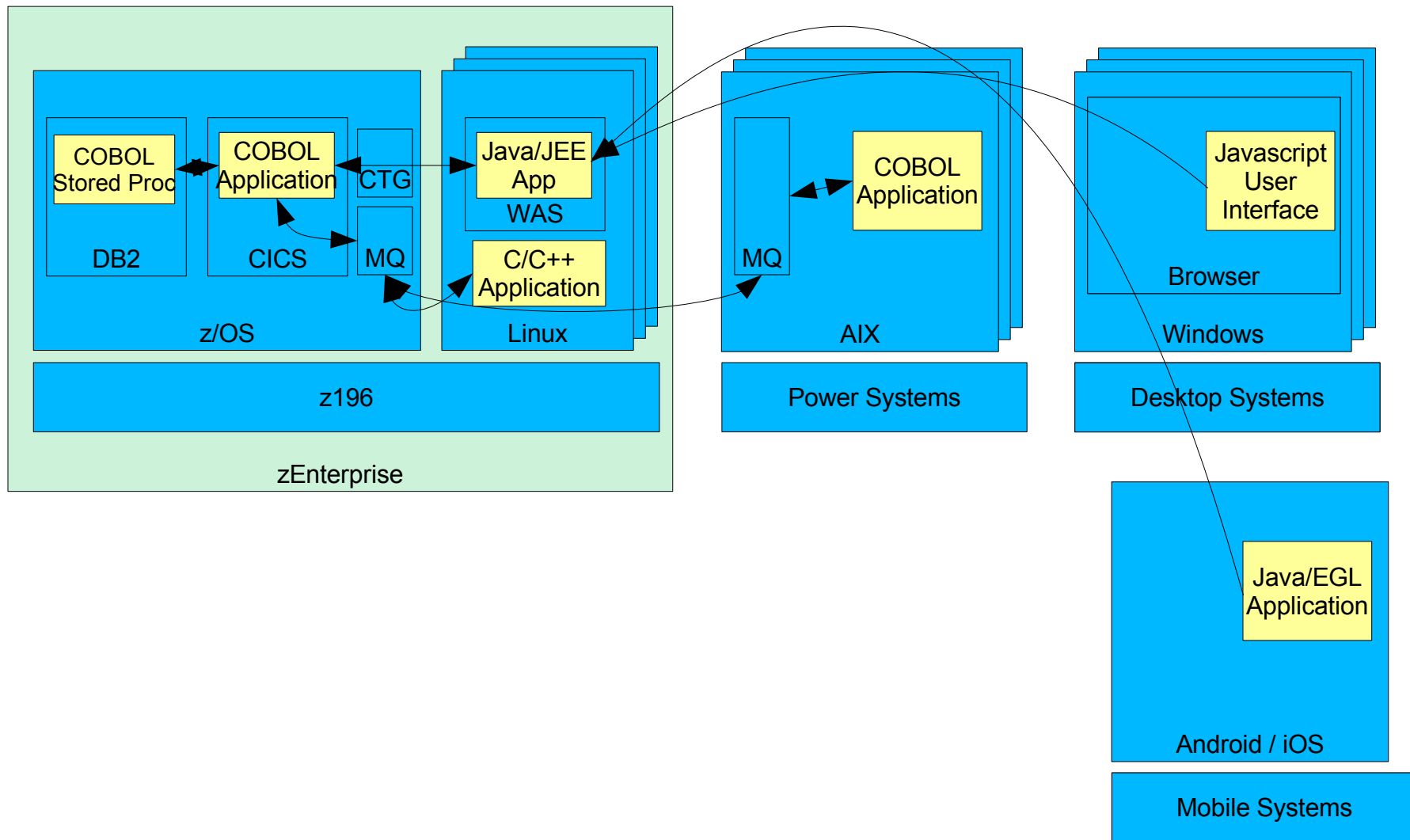
Agenda

- The Changing Application Landscape
- **Multi-platform applications**
- Past – Dedicated development teams
- Future – Nimble teams
- Tools Support the team
- Summary

Multi-platform Applications

- Counter to what we were seeing in the early 2000s
 - Consolidation of popular environments
 - Rigid structures for inter-system communication
- We are seeing an explosion of systems, devices, interfaces, and languages
 - The programming community is becoming more diverse
 - More heterogeneous
 - More multi-platform
- Existing applications are not being de-commissioned
 - The data they manage is what is in demand
 - These applications are useful
 - Are depended upon by all those that have built on top and around them
- Development teams face a challenge
 - How to manage, maintain, and even enhance existing applications
 - And also create new and exciting, cutting edge applications

Multi-platform applications often look like this



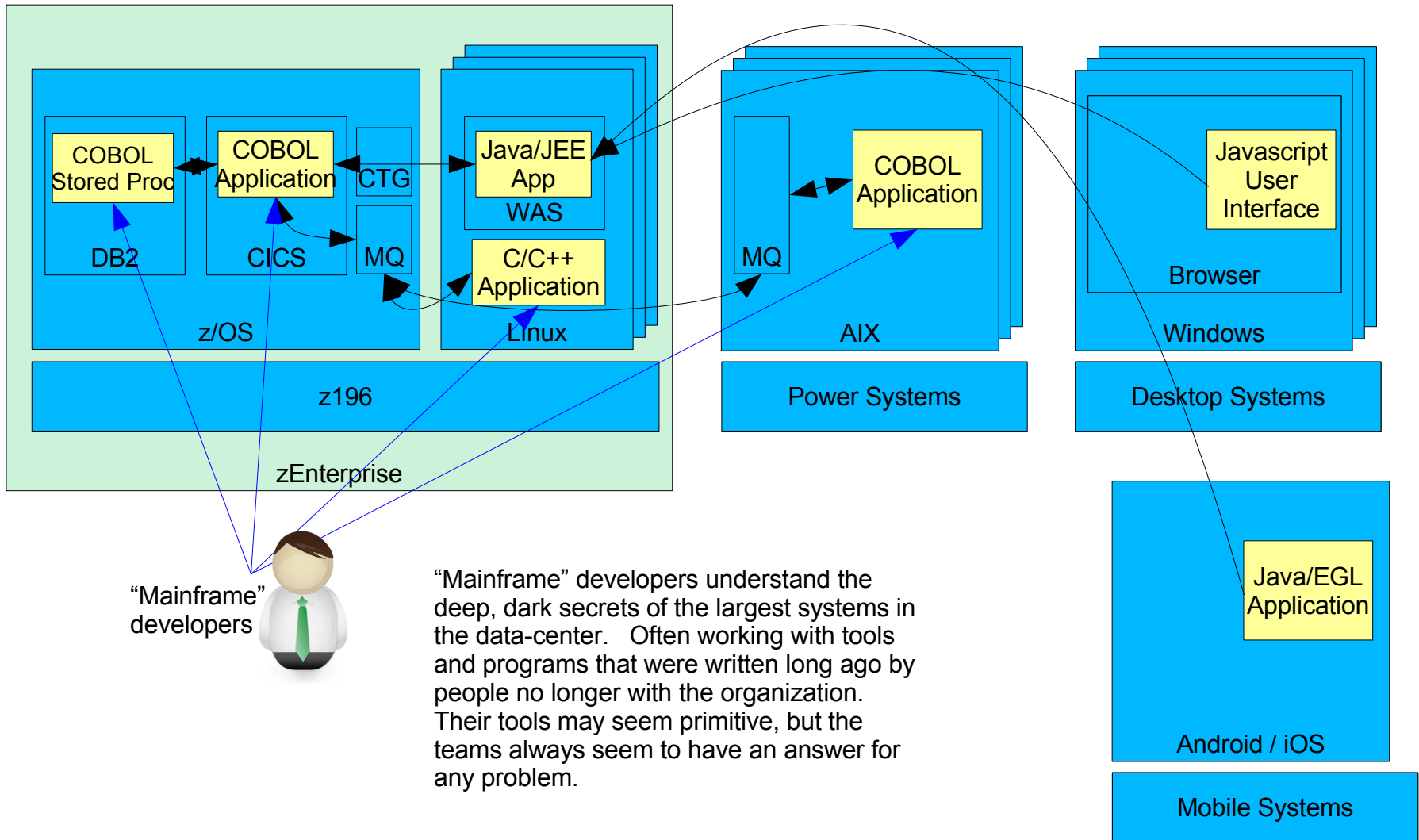
Agenda

- The Changing Application Landscape
- Multi-platform applications
- **Past – Dedicated development teams**
- Future – Nimble teams
- Tools Support the team
- Summary

Dedicated Development Teams

- Teams have been organized by platform
 - “Mainframe” developers
 - “distributed” developers
 - “web” developers
 - “mobile app” developers
 - ... and the “architecture group”

Development teams were often isolated

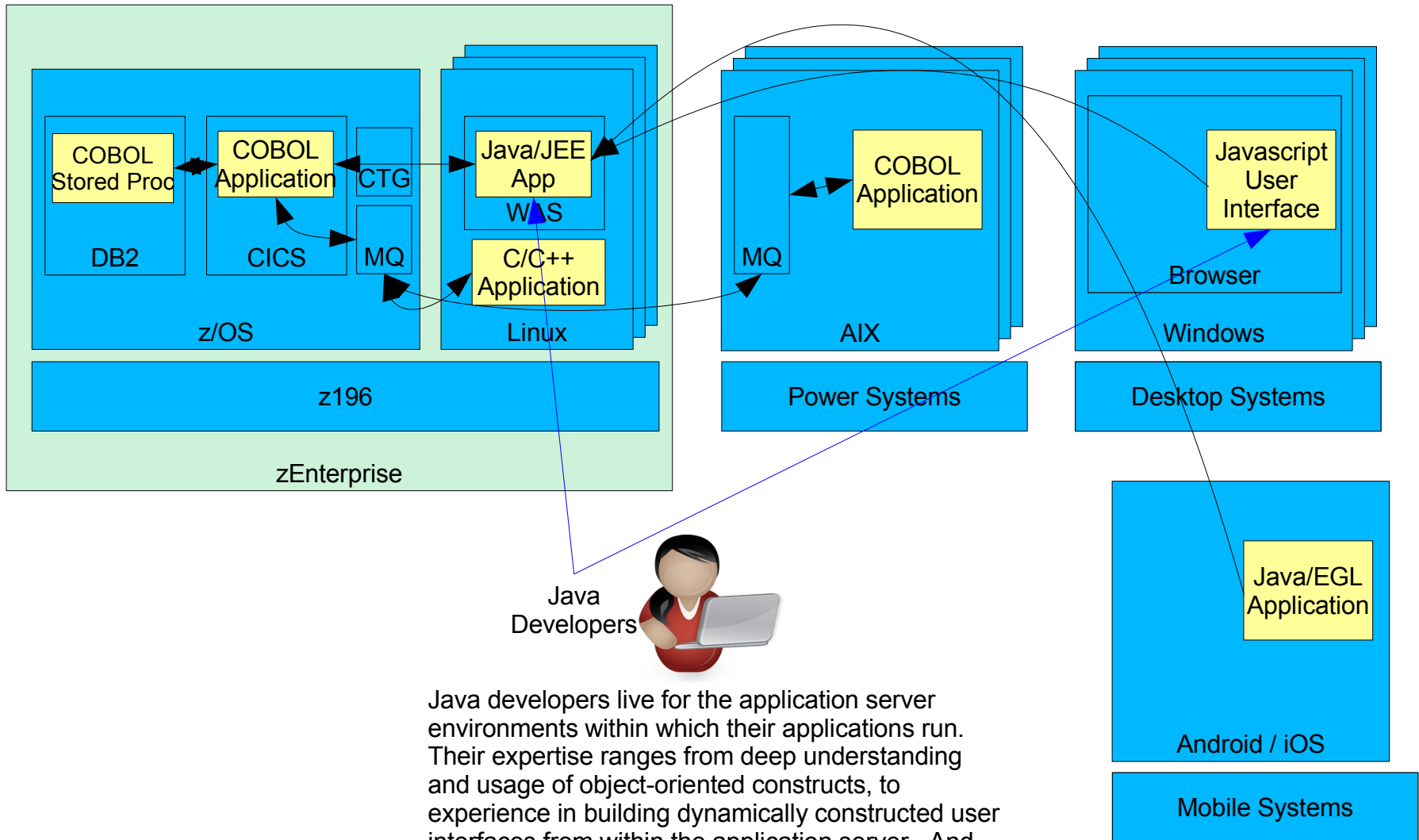


“Mainframe” developers



“Mainframe” developers understand the deep, dark secrets of the largest systems in the data-center. Often working with tools and programs that were written long ago by people no longer with the organization. Their tools may seem primitive, but the teams always seem to have an answer for any problem.

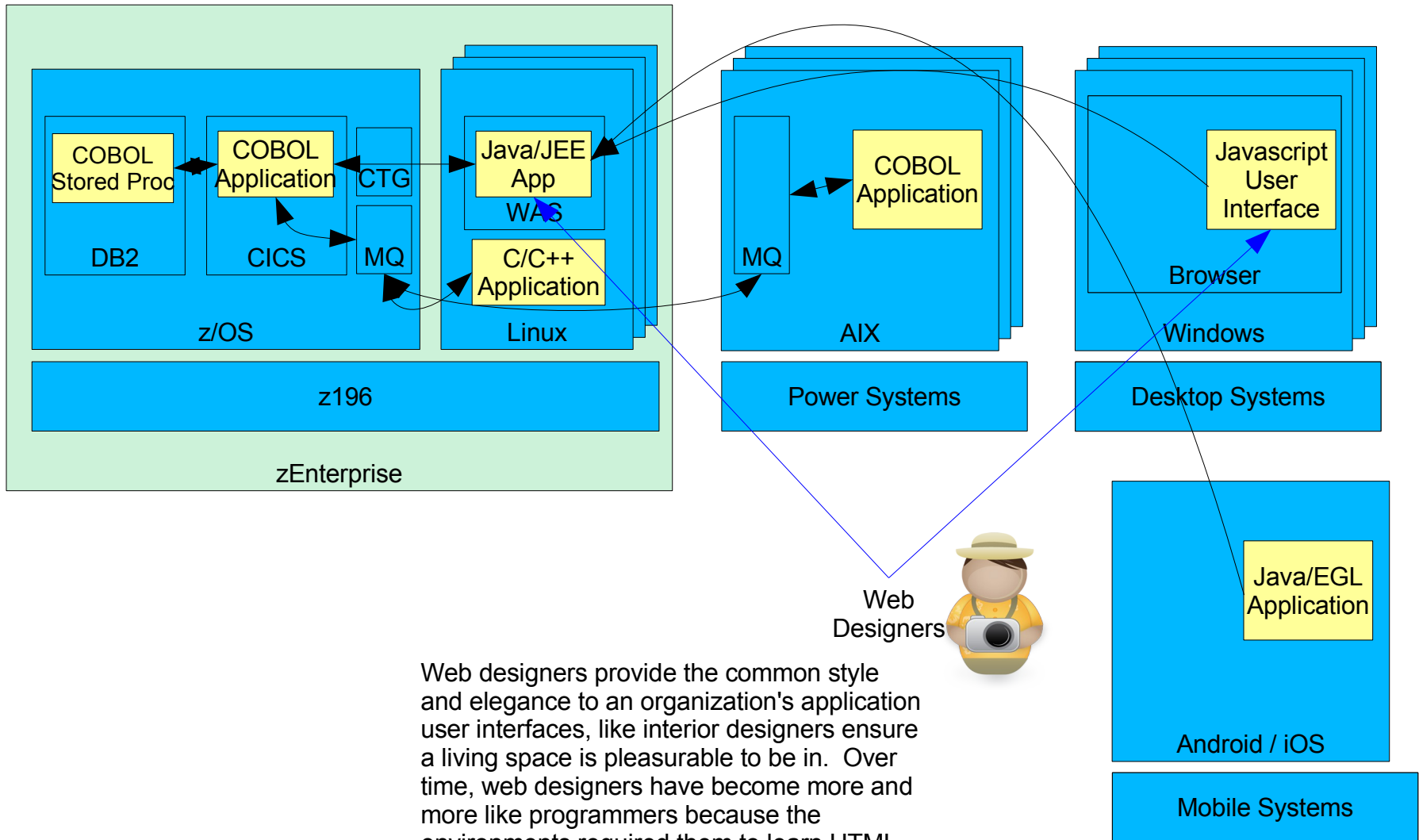
Development teams were often isolated



Java Developers

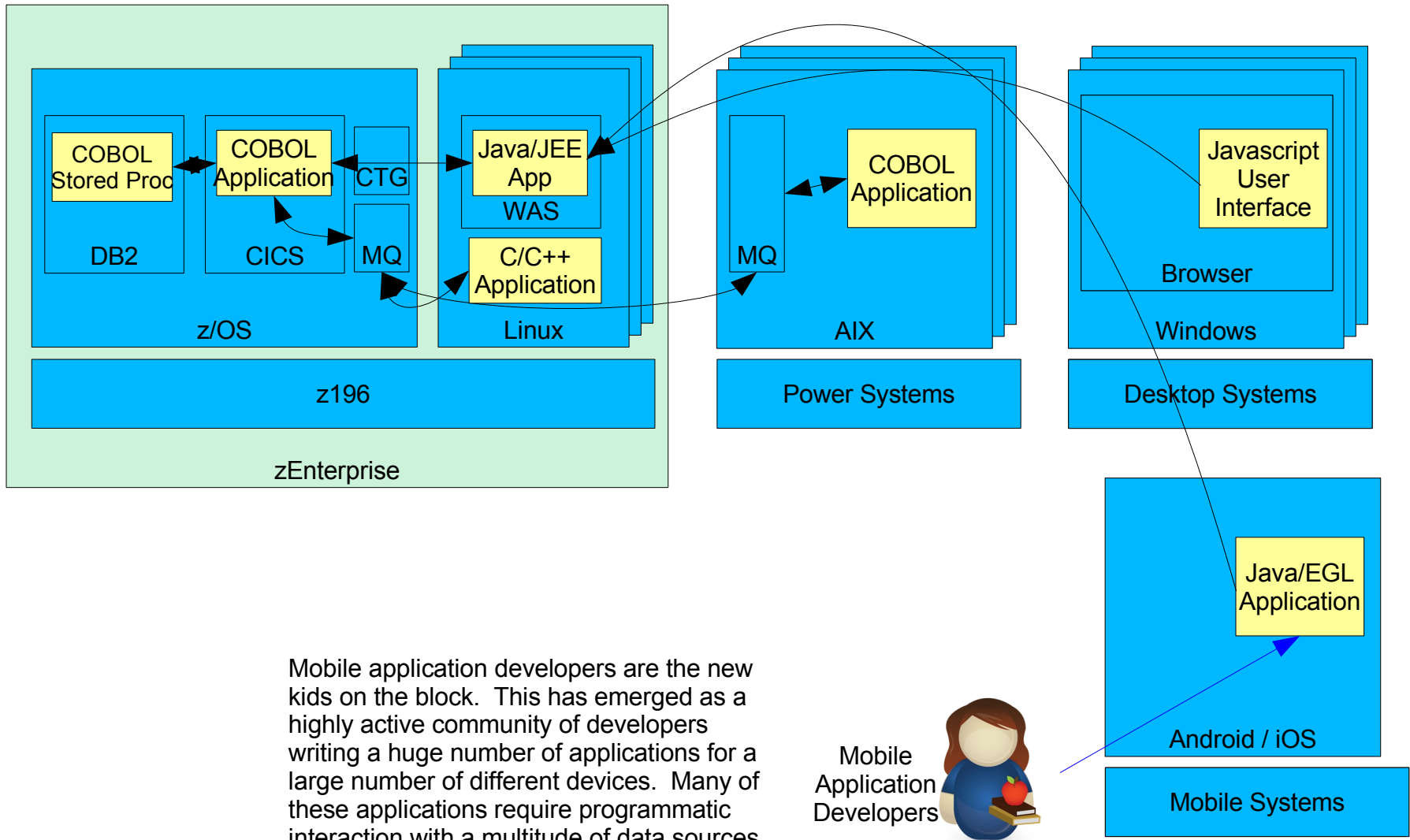
Java developers live for the application server environments within which their applications run. Their expertise ranges from deep understanding and usage of object-oriented constructs, to experience in building dynamically constructed user interfaces from within the application server. And Java applications are running in z/OS as well, as batch or online transaction processing programs.

Development teams were often isolated



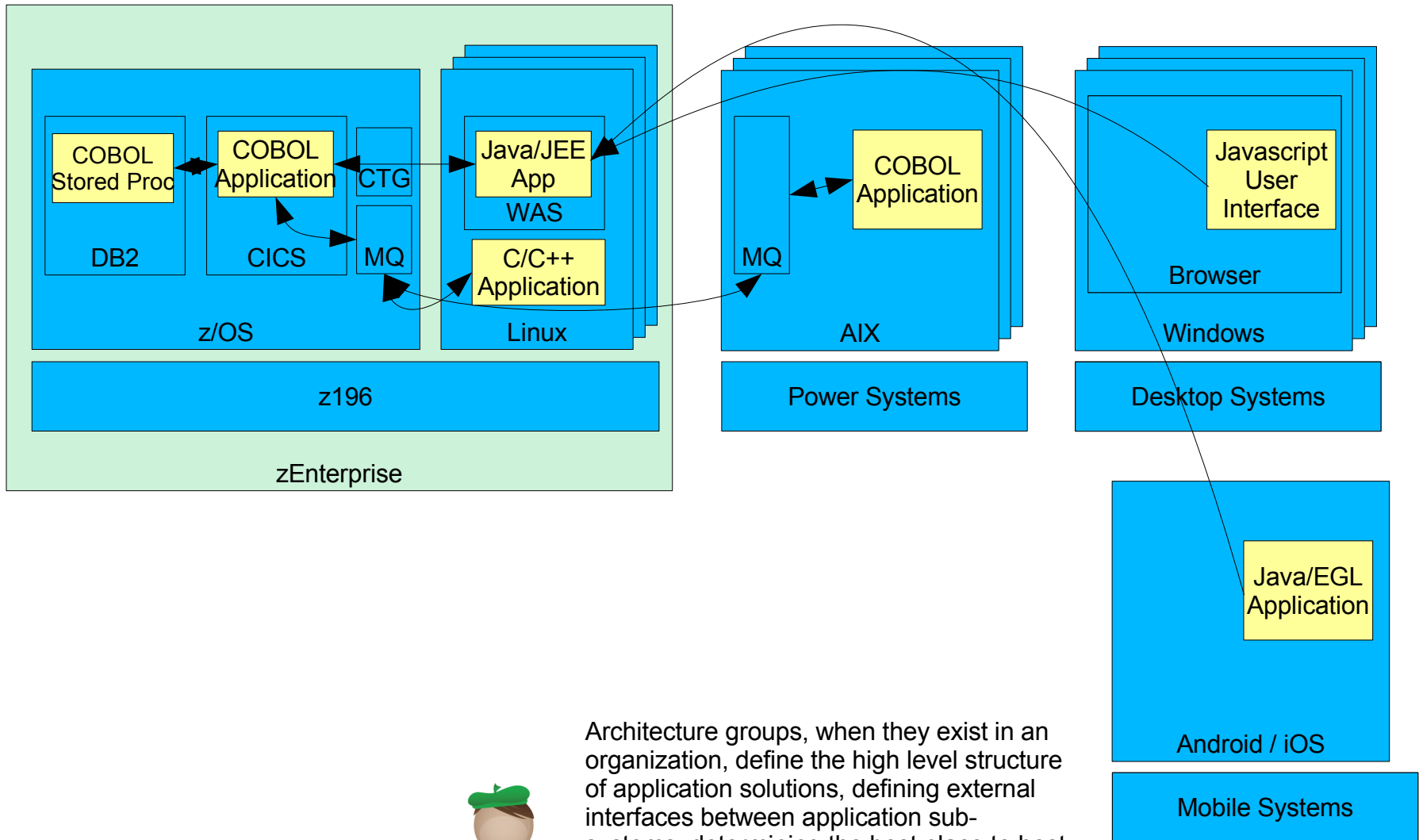
Web designers provide the common style and elegance to an organization's application user interfaces, like interior designers ensure a living space is pleasurable to be in. Over time, web designers have become more and more like programmers because the environments required them to learn HTML, CSS, XSLT, and other constructs

Development teams were often isolated



Mobile application developers are the new kids on the block. This has emerged as a highly active community of developers writing a huge number of applications for a large number of different devices. Many of these applications require programmatic interaction with a multitude of data sources.

Development teams were often isolated

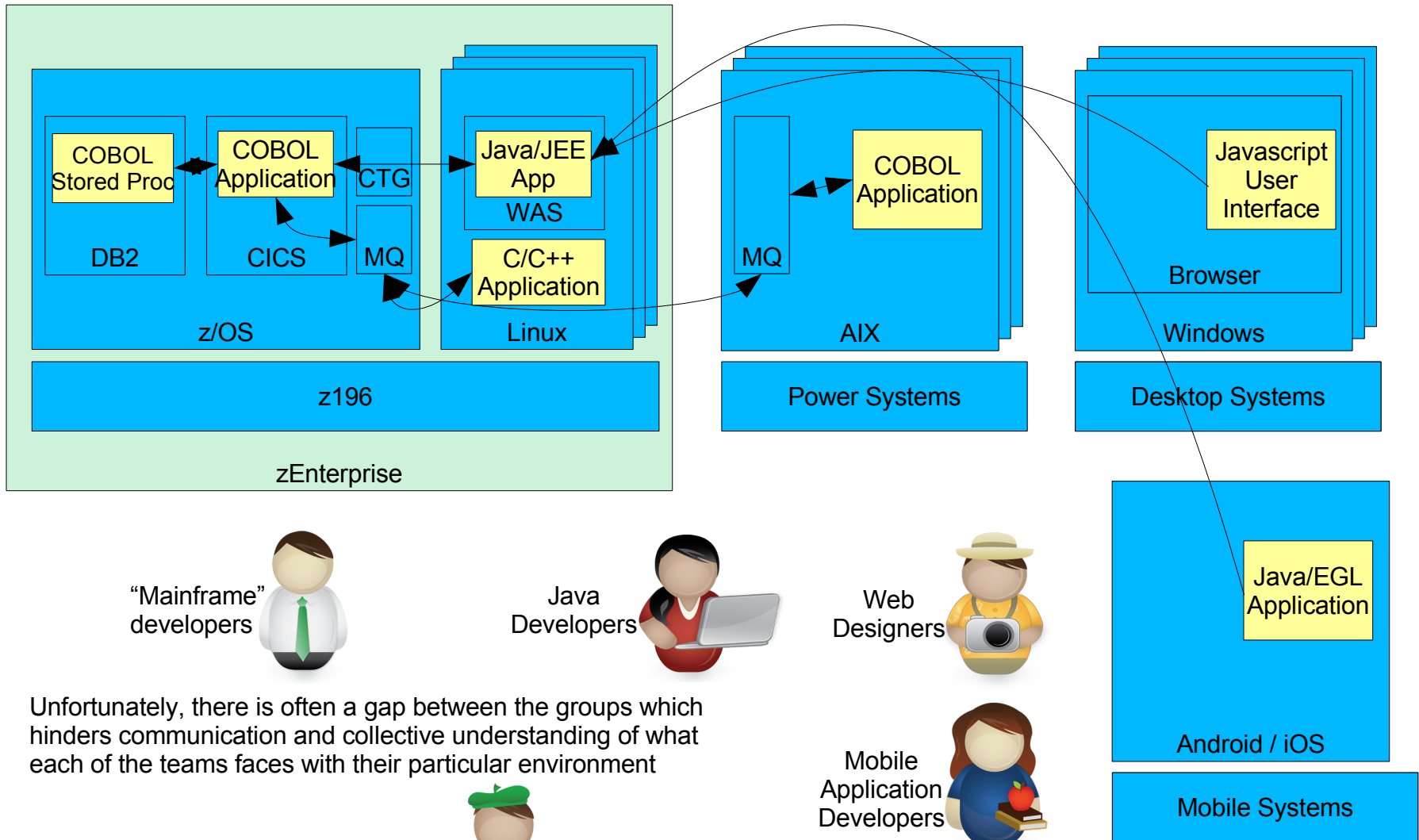


Architecture Group



Architecture groups, when they exist in an organization, define the high level structure of application solutions, defining external interfaces between application sub-systems, determining the best place to host elements of the application, and handing down designs to be implemented by multiple teams

Development teams were often isolated



Unfortunately, there is often a gap between the groups which hinders communication and collective understanding of what each of the teams faces with their particular environment



Architecture Group

Agenda

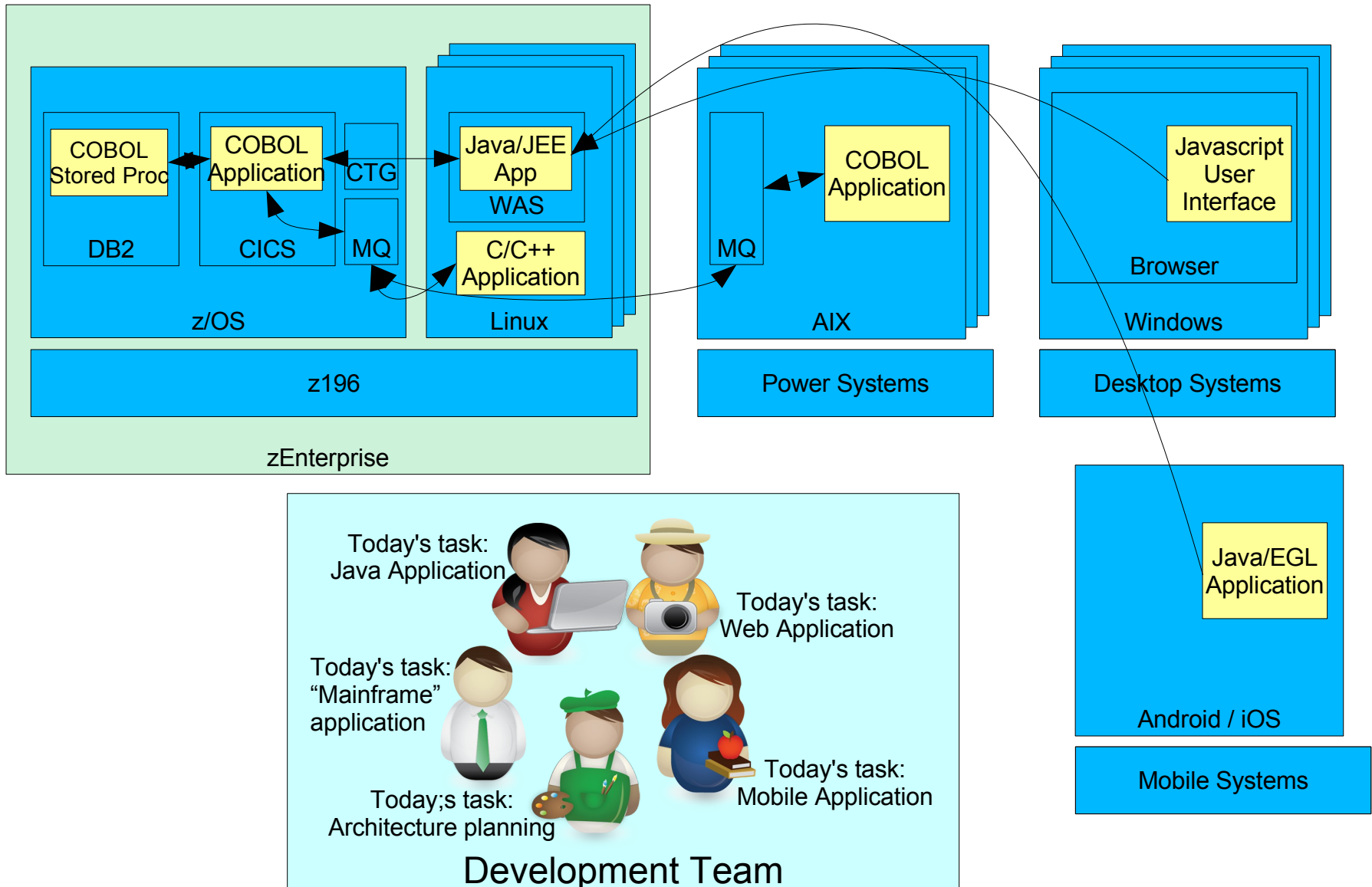
- The Changing Application Landscape
- Multi-platform applications
- Past – Dedicated development teams
- **Future – Nimble teams**
- Tools Support the team
- Summary

Emerging trends – Nimble teams

- Either by necessity or desire, Application developers are
 - increasingly multi-lingual
 - Increasingly multi-platform knowledgeable
 - Increasingly multi-environment enabled
- At the same time, organizations are challenged
 - Address new business opportunities in different geographies, with different clients
 - Create cutting-edge applications using the latest features of the latest devices
 - Maintain and extend existing applications upon which new applications depend
- Programming teams will need to shift quickly
 - From project to project
 - From platform to platform
 - From runtime environment to runtime environment ... **including those on z/OS!**

- Application programming teams **must** become Nimble!

Development teams will need to shift from task to task



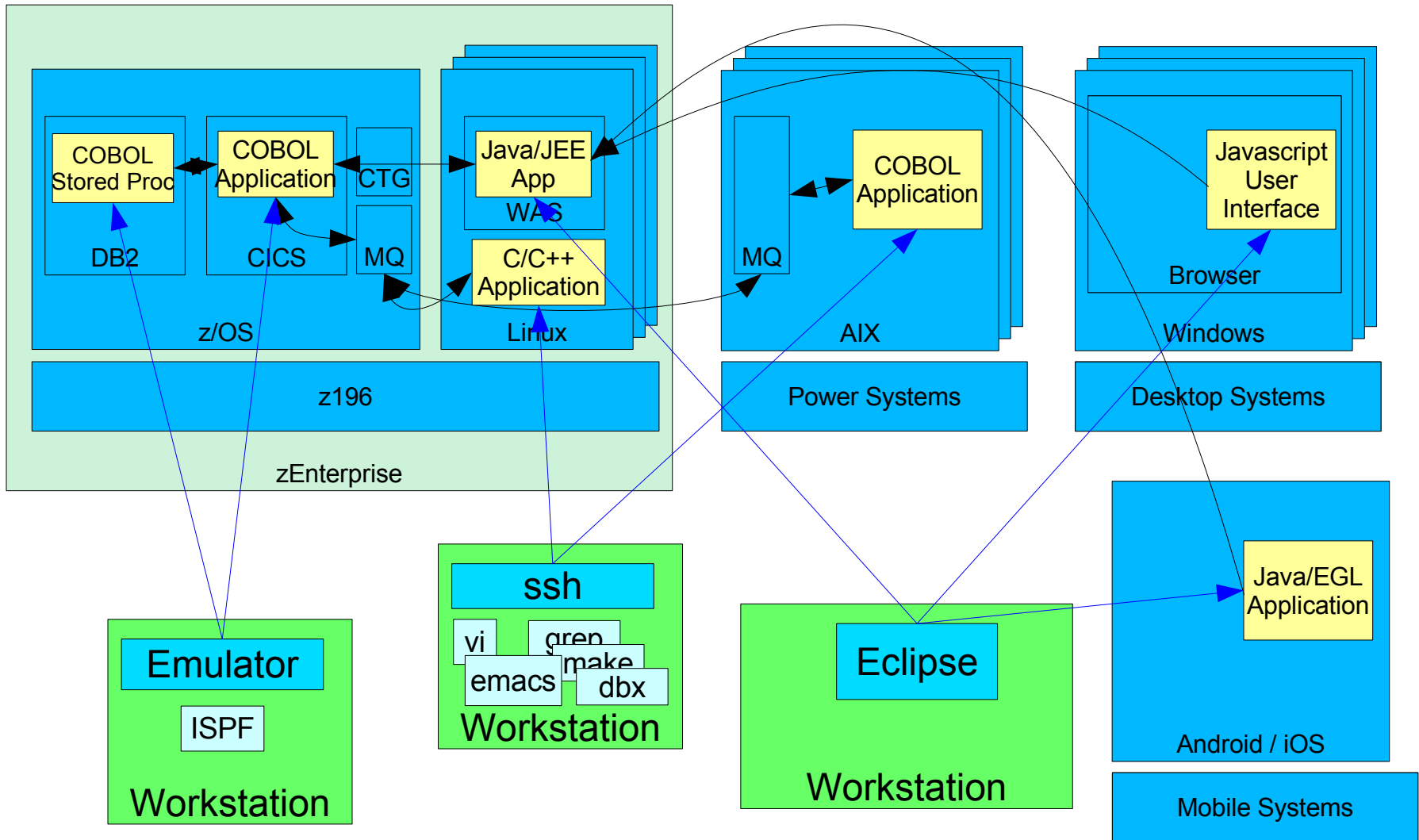
Challenges to being Nimble

- There are real challenges to being a nimble application developer
 - Must learn new environments quickly and not forget other environments they've used
 - Must be able to pick up an existing application's source code and work effectively
 - Must be able to switch between environments quickly
- There are benefits as well
 - Obtain skills that are relevant to a wide range of applications and environments
 - Be able to contribute to a variety of projects
 - Increase self-marketability

Agenda

- The Changing Application Landscape
- Multi-platform applications
- Past – Dedicated development teams
- Future – Nimble teams
- Tools Support the team
- Summary

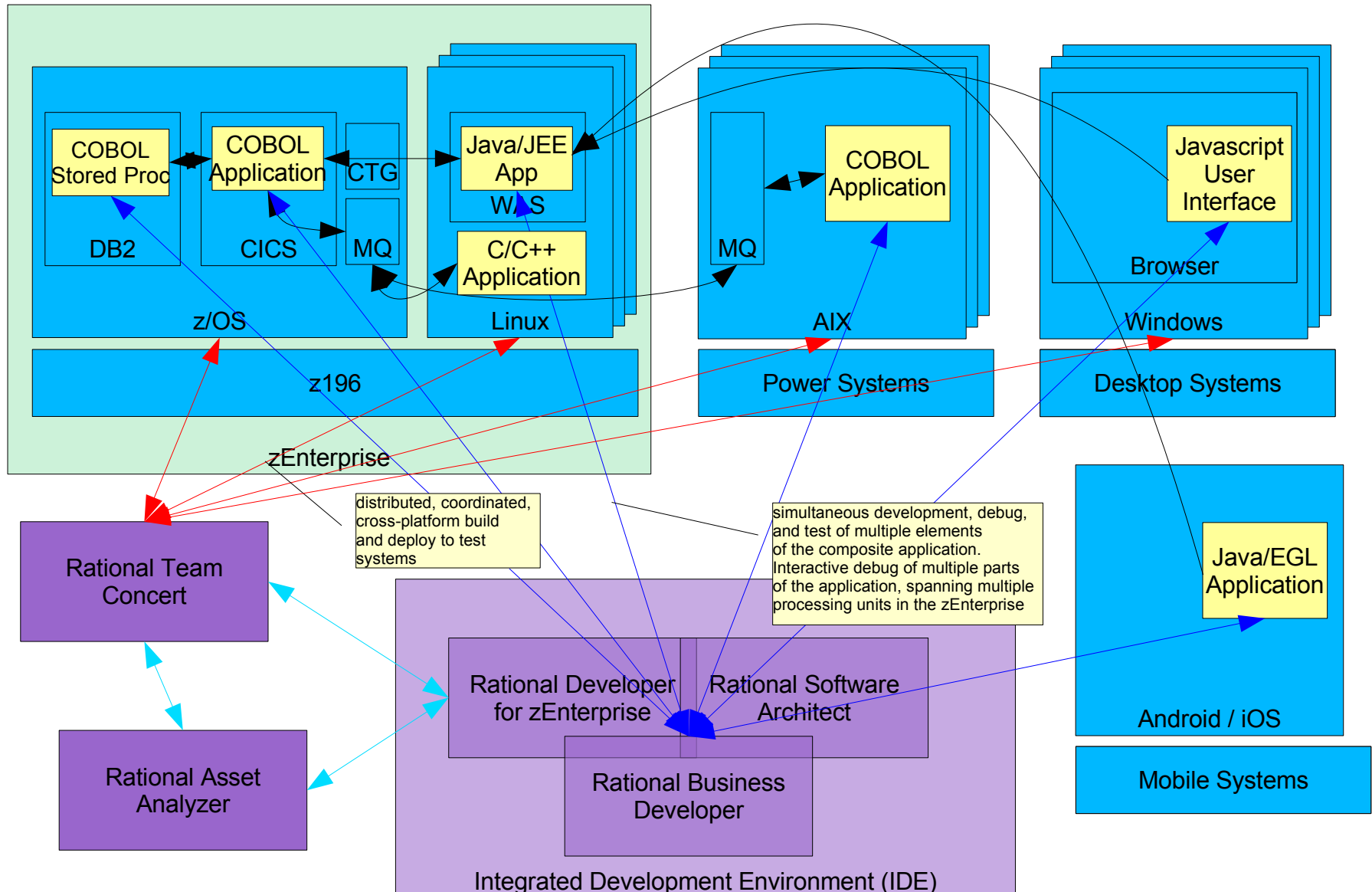
Historically, application development tools have been platform-centric



Tools – A Silver Bullet (in more ways than one)

- Integrated Development Environments (IDEs have advanced incredibly in the past several years
 - Cross-platform differences are reduced
 - Access to multiple systems simultaneously is expected
 - Multiple language support is now common-place
 - Integration of multiple development tools into a single development environment is now reality
- At the same time, effective use of an IDE requires education and experience
 - On first sight, there is an overload of information
 - On second sight, there are “hidden” features – Where do I click?
 - On third sight, there are sometimes endless UI elements to interpret and understand
- But past the learning curve ...
 - Using an IDE allows programmers to concentrate on the application
 - Regardless of programming language
 - Regardless of runtime environment

Tool-assisted Multi-platform Application Development



Some features of a multi-platform IDE

The screenshot displays the Rational Team Concert IDE interface. The main window shows a work item titled "Enhancement 104724" with the summary "No 'man' page exists for the scm command-line tool". The status is "Triaged".

Details:

- Type: Enhancement
- Severity: Normal
- Found In: 2.0.0.2
- Creation Date: Feb 1, 2010 9:17 AM
- Created By: Tim Hahn
- Team Area: Source Control / Rati...cert
- Filed Against: Source Control
- Tags: (empty)
- Owned By: Unassigned
- Priority: Unassigned
- Planned For: Backlog
- Estimate: (empty) Correction: (empty)
- Time: (empty)
- Due Date: None

Description:

It is VERY common for command-line tools to have an associated "man" page on Linux and AIX systems. the "scm" command-line tool, however, does not have such a page.

```
[tih@aruba bin]$ man scm
No manual entry for scm
[tih@aruba bin]$
```

A man page is usually MORE informative than the interactive help (scm -? or scm -h or scm --help) but often not as detailed as a command reference (book or Infocenter format).

Discussion (5 comments):

2. Tim Hahn, Feb 2, 2010, 11:21 AM
I believe that the right information exists in the Infocenter. However, if users are using the command-line tool (interactively), then they are likely to be very savvy command-line tool users in general. And these types of users are very used to using "man" pages to get quick access to textual help information.
3. John Camelon, Feb 3, 2010, 3:31 PM
+1 for having some documentation in man format
-1 for having different content than what is in infocentre ... it is not worth the effort to maintain 2 sets of doc
4. Evan Hughes, Feb 4, 2010, 4:09 PM
To do a decent job at man pages we'd also have to have an installer that put things in the expected place.
5. Tim Hahn, Feb 5, 2010, 10:58 AM
@echughes I agree with the point about getting the installer to put the man page information into the appropriate place. I had opened another defect to get a symbolic link to the "scm" tool placed in /usr/bin on Linux systems which is related to this notion of a better installer. I've added a link to that work item.

On the comment about having information different from the Infocenter - I think that it should be possible to source the Infocenter help and the man page help from the same source text. This might take some re-formatting somehow at build time to send the text to a man page format (ASCII text, nroff format) and InfoCenter format (HTML, I believe).

Quick Information:

- Subscribers (6): DC, DK, EH, JC, SP, TH
- Related (1): 104723
- Mentions (1)

Work Item List:

Id	Owned By	P	Status	Resolution	Resolution Date	S	Summary
64793	Unassigned		Resolved	Invalid	Nov 25, 2008 10:13 A	A	Issue with certificate
13965	Rosalind T Radcliffe		New				Drive RAA scans from RTC build engine
13966	Rosalind T Radcliffe		New				RTC Build Integration with RBD
13913	Rosalind T Radcliffe		New				Improve RDz Integration
10472	Unassigned		Triaged				No "man" page exists for the scm command-line tool
10472	Unassigned		Triaged				scm command-line tool is not added to user's command search path on install
15390	Unassigned		Triaged				Need prereq/coreq functionality for workspace integrity

Some features of a multi-platform IDE

The screenshot shows the Rational Software Architect IDE interface. The main workspace contains a hand-drawn diagram titled "Remote Desktop Approach". The diagram is divided into two sections by a vertical line. On the left, a stick figure labeled "User (Employee)" is connected to a box labeled "workstation" which contains a "Remote Desktop Client". On the right, another box labeled "workstation" contains a "remote desktop server" and an "IDE". The "remote desktop server" is connected to the "Remote Desktop Client" and the "IDE". The "IDE" is connected to "local files" (represented by a cylinder) and two external systems, "System 1" and "System n".

The IDE interface includes a Project Explorer on the left showing a project structure with "Miscellaneous Models" and "SampleSCAProject". The Properties window at the bottom shows settings for the sketch, including the font "Tahoma" and a "Set Background" button. The status bar at the bottom indicates "153907" and "jazzdev.torolab.i...".

Some features of a multi-platform IDE

The screenshot displays the IBM Rational Software Architect IDE interface. The main editor window shows the source code for `HelloWorld.java` in the package `com.ibm.us.hahnt.helloworld`. The code includes a class `HelloWorld` with a `main` method that prints the number of arguments and each argument, followed by "Hello World".

```
package com.ibm.us.hahnt.helloworld;

public class HelloWorld {

    /**
     * This is just a VERY simple program to get myself re-aquainted with the RAD/RSA development environment
     *
     * @param args
     */
    public static void main(String[] args) {

        for (int i=0; i<args.length; i++) {
            if (i==0) {
                System.out.printf("number of arguments = %d\n", args.length);
            }
            System.out.printf("arg[%d] = %s\n", i, args[i] );
        }
        System.out.println("Hello World");
    }
}
```

The Outline view on the right shows the project structure with `main(String[]) : void` selected. The Console view at the bottom shows the output of the program:

```
<terminated> HelloWorld [Java Application] /opt/ibm/SDP/jdk/bin/javaw (Feb 18, 2011 10:13:28 AM)
number of arguments = 4
arg(0) = first
arg(1) = second
arg(2) = third
arg(3) = fourth
Hello World
```

The status bar at the bottom indicates the current file is `com.ibm.us.hahnt.helloworld.HelloWorld.main(String[] args) : void - HelloWorldInJava/src` and the connection status is `Connect: (0%)`.

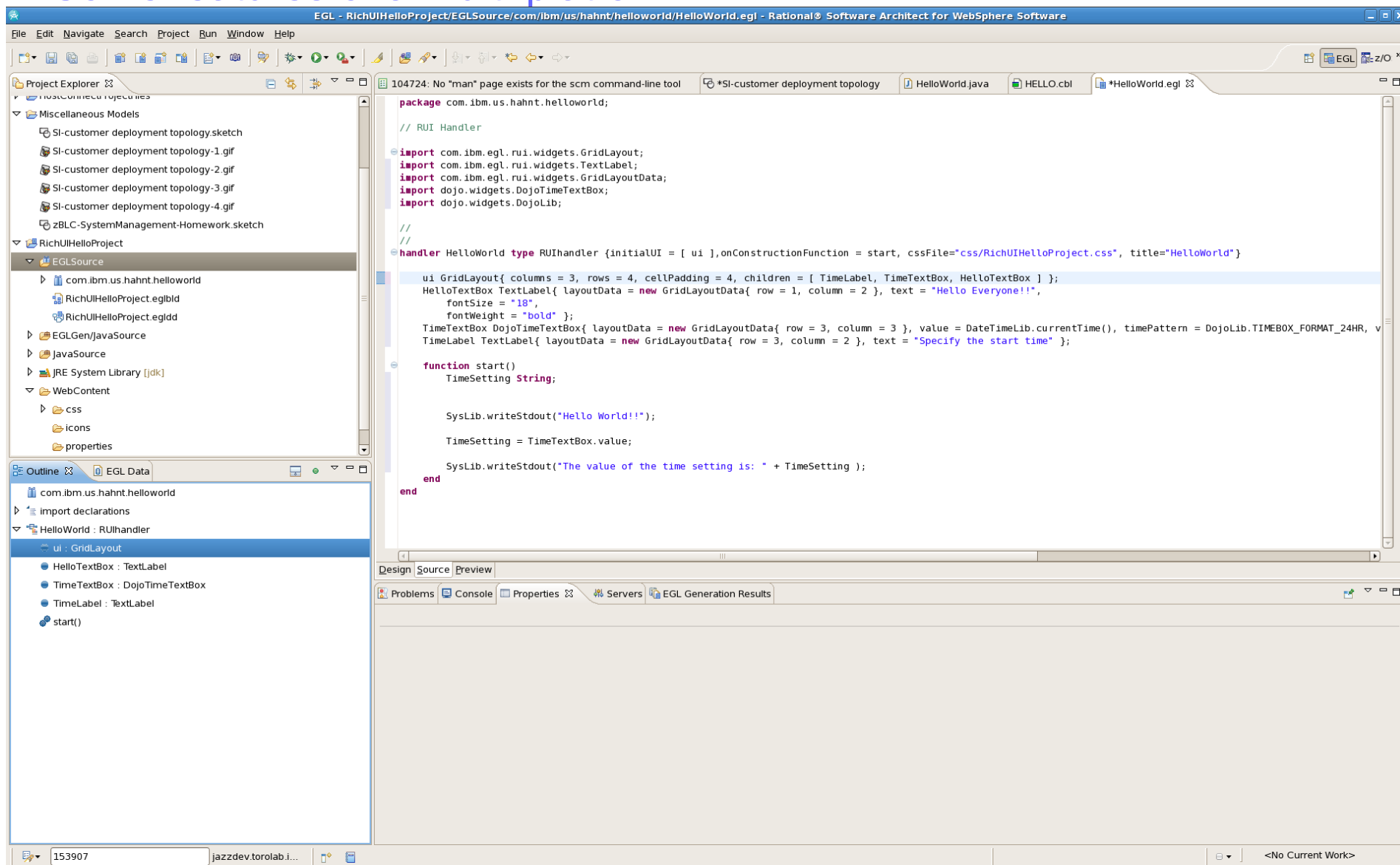
Some features of a multi-platform IDE

The screenshot displays a multi-platform IDE interface with the following components:

- Editor:** Shows COBOL code for 'HELLO.cbl'. The code includes an identification division, environment, data, and procedure divisions. The procedure division contains logic to increment a counter and display a message.
- Project Explorer:** Lists project files such as 'HelloWorldInJava', 'SampleSCAProject', and 'SimpleCOBOLProgram'.
- Remote Systems:** A tree view showing the project structure on a remote system, including 'HELLO.cbl' and various error logs.
- Properties/Outline:** Shows the structure of the 'PROGRAM: HELLO' with sections like 'IDENTIFICATION DIVISION', 'Environment division', 'Data division', and 'Procedure division'.
- Remote Error List:** A table showing file system mappings for the subsystem 'JES'.

Resource	Parent filter pool	Parent filter	Number of filter strings	Connection-private
Retrieved Jobs	CN-mvs040.rtp.raleigh.ibm.com-com.ibm.z	Not applicable	1	Yes
My Jobs	aruba.com.ibm.zos.jes	Not applicable	1	No

Some features of a multi-platform IDE



The screenshot displays the Rational Software Architect IDE interface. The main window shows the source code for an EGL application named 'HelloWorld'. The code is as follows:

```

package com.ibm.us.hahnt.helloworld;

// RUI Handler

import com.ibm.egl.rui.widgets.GridLayout;
import com.ibm.egl.rui.widgets.TextLabel;
import com.ibm.egl.rui.widgets.GridLayoutData;
import dojo.widgets.DojoTimeTextBox;
import dojo.widgets.DojoLib;

//
handler HelloWorld type RUIhandler {initialUI = [ ui ],onConstructionFunction = start, cssFile="css/RichUIHelloProject.css", title="HelloWorld"}

ui GridLayout{ columns = 3, rows = 4, cellPadding = 4, children = [ TimeLabel, TimeTextBox, HelloTextBox ] };
HelloTextBox TextLabel{ layoutData = new GridLayoutData{ row = 1, column = 2 }, text = "Hello Everyone!!",
    fontSize = "18",
    fontWeight = "bold" };
TimeTextBox DojoTimeTextBox{ layoutData = new GridLayoutData{ row = 3, column = 3 }, value = DateTimeLib.currentTime(), timePattern = DojoLib.TIMEBOX_FORMAT_24HR, v
TimeLabel TextLabel{ layoutData = new GridLayoutData{ row = 3, column = 2 }, text = "Specify the start time" };

function start()
    TimeSetting String;

    SysLib.writeStdout("Hello World!!");

    TimeSetting = TimeTextBox.value;

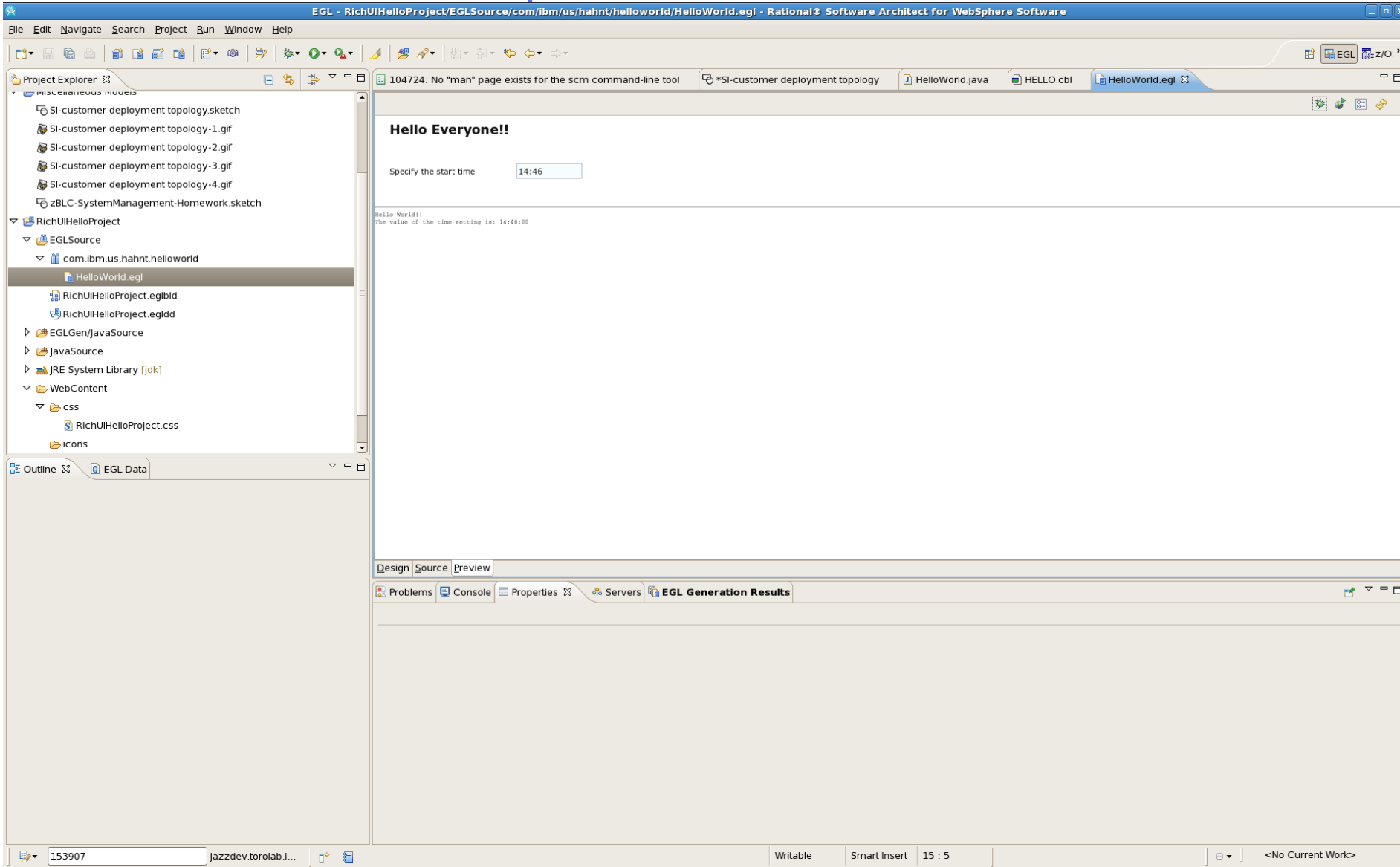
    SysLib.writeStdout("The value of the time setting is: " + TimeSetting );

end
end

```

The interface also shows the Project Explorer on the left, the Outline view at the bottom left, and the Console at the bottom right. The status bar at the bottom indicates the current work is '<No Current Work>'.

Some features of a multi-platform IDE



Agenda

- The Changing Application Landscape
- Multi-platform applications
- Past – Dedicated development teams
- Future – Nimble teams
- Tools Support the team
- **Summary**

Summary

- Computing environments, and the applications that run in them continue to rise in complexity
 - Multiple platforms
 - Multiple languages
 - Multiple runtime environments

 - And existing environments are not going away!!
- The days of being a siloed, single-purpose, application developer are numbered
 - Younger professionals are already flexible to multiple environments

 - Flexibility to apply to multiple projects is a valued skill for the organization
- **The future is wide open for nimble development teams**
 - Maintain and enhance existing applications with speed and precision
 - Design and Implement new applications on cutting edge devices

 - Using software development tools that enable multi-platform application development



www.ibm.com/software/rational

© Copyright IBM Corporation 2011. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, Telelogic, the Telelogic logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

Useful Links

- Rational Developer for zEnterprise Information:
 - <http://www.ibm.com/software/rational/products/developer/zenterprise/>
- Rational Team Concert Information:
 - <http://www.ibm.com/software/rational/products/rtc/>
- Rational Asset Analyzer Information
 - <http://www.ibm.com/software/rational/products/raa/>
- Rational Application Development “Cafes”:
 - <https://www.ibm.com/developerworks/rational/community/cafe/>
- Jazz Team Blog:
 - <http://jazz.net/blog/>
- My information
 - email: hahnt@us.ibm.com
 - Blog: <https://www.ibm.com/developerworks/mydeveloperworks/blogs/applicationmodernization>