Table of contents

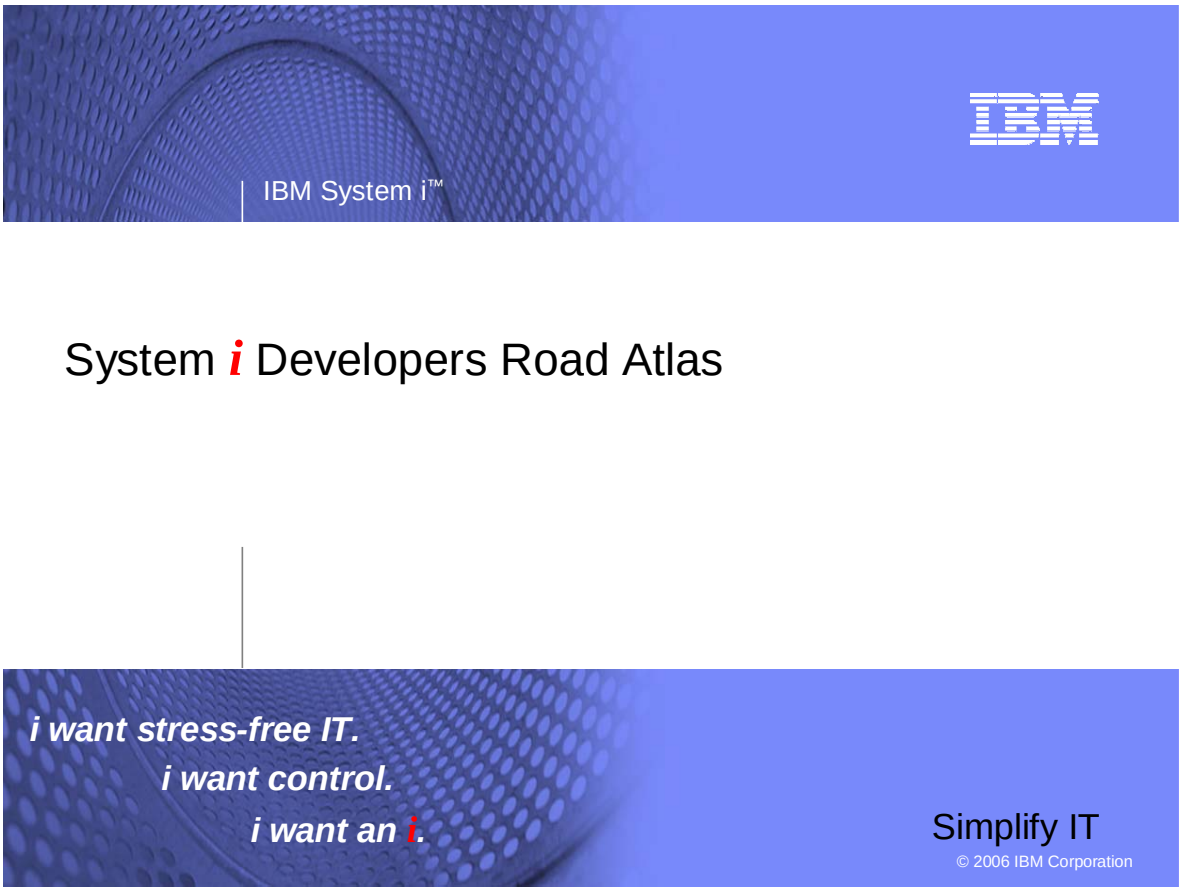
About the author 3
Alison Butterill Certified IT Specialist - WebSphere and System i ISV Enablement Team, IBM

Introduction 4
Agenda 5
Goals of the System i Developers Road Atlas 6
System i Developers Roadmap V3: 2005 architecture 7
Going on vacation? 9
Industry direction for development: SOA 10
System i Developers Road Atlas 11
System i Developers Road Atlas (a closer look) 12
System i Developers Road Atlas (traditional applications) 13
System i Developers Road Atlas (enhanced user experience) 14
System i Developers Road Atlas (modular architecture) 15
System i Developers Road Atlas (application integration and business process integration) 16
System i Developers Road Atlas (the details) 17
Planning the trip 18
Packing for the road trip 19
Improving programmer productivity 20
Remote System Explorer 21
Some programmer productivity plug-ins 23
System i Road Atlas (modernizing) 24
Starting point: Understanding traditional development 25
Driving directions: Refacing 26
Driving directions for refacing: Exploring GUI options 27
Driving directions for refacing: Creating GUIs for 5250 programs 28
IBM WebFacing Tool 30
Architecture of the IBM WebFacing Tool 31
WebSphere Host Access Transformation Services (HATS) 32
WebSphere HATS architecture 33
WebFacing Deployment with HATS Technology 34
iSeries Access for Web 35
Driving directions: Modularization 36
Modularizing the code 37
Modular architecture 38
Moving to modern compilers 40
Using ILE to modularize the code 41
System i Road Atlas to SOA (new development) 42
Languages available for i5/OS V5R4 43
System i Road Atlas to SOA (new development, continued) 44
WebSphere Development Studio Client: IDE 45
WebSphere Development Studio for i5/OS 46
WebSphere Development Studio Client: Advanced Edition 48
Enterprise Generation Language 49
PHP and System i 50
System i Road Atlas to SOA: Locating components elsewhere 51
Locating components 52
Lotus 53
The destination 54

Why integrate applications? 55
Why is business flexibility important? 56
What are the barriers to business flexibility? 57
SOA builds flexibility on your current investments: The next stage of integration 58
System i Road Atlas (distributed deployment) 59
System i Road Atlas (distributed deployment, continued) 60
WebSphere business integration 61
System i Tools Innovation Program: IBM and ISV tools 62
SOA in a box 63
Summary: The System i Developers Road Atlas 64
Resources 65
Trademarks and special notices 66

About the author

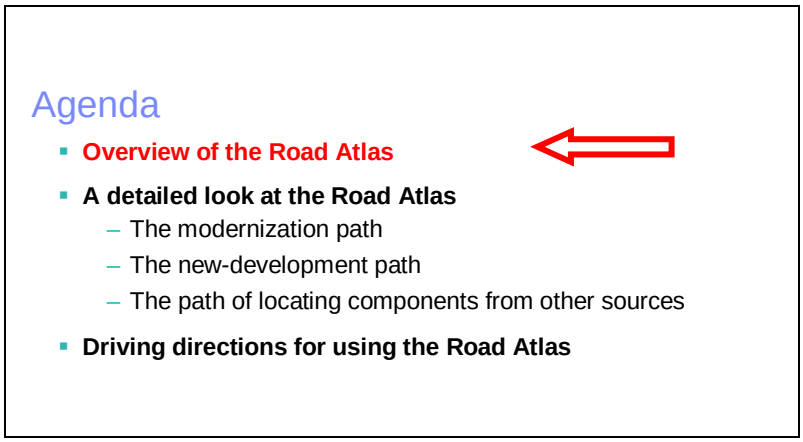
Alison is an IBM Certified Consulting I/T Specialist. Her current position is System i Application Development Offering and Strategy manager for the System i Brand Team. Alison has worked in the midrange area for more than 25 years. She holds a wide variety of positions at IBM, always specializing in the areas of application development and databases. In her present position, Alison works with a wide variety of IBM teams to develop strategic directiona and offerings for developers on the System i platform. One of the System i key initiatives, the System i Developers Road Atlas, is a key component of her current responsibilities. Alison is a regular speaker at technical conferences and COMMON conferences around the world and holds numerous Speaker Excellence medals from COMMON North America.



Introduction

Welcome to this online course, entitled “IBM System i Developers Road Atlas.”

The IBM® System i™ Developers Road Atlas (formerly called the *Roadmap*) is now at Version 4. The original version of the Roadmap provided an evolutionary path for developers to follow when modernizing their applications on the System i platform, gradually increasing their ability to accommodate the evolving needs of the business. Since the first version, IBM has listened to the experiences of developers and development shops as they have progressed down a path toward modernization. The result has been clarification and refinement of the messages that were contained in the original roadmap strategy.



Agenda

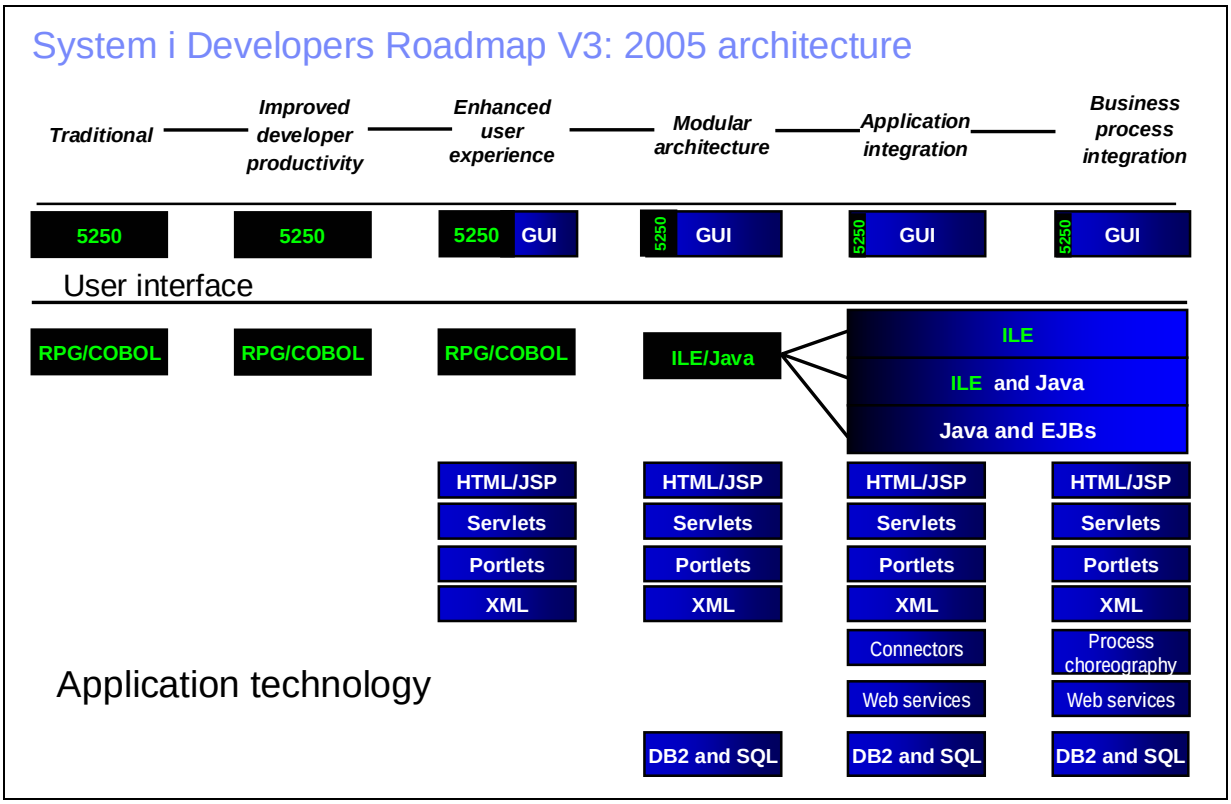
This course dedicates several panels for the purpose of explaining what the System i Road Atlas is — from a high-level perspective. You will learn about the history of its previous versions and why it is important. The largest section of this course is dedicated to focusing, in detail, on the choices and functional considerations for taking various paths toward application modernization. Then, as this course begins to conclude, you will learn how to manage your IT projects as you move toward a Web-enabled environment.

Goals of the System i Developers Road Atlas

The three main Road Atlas goals are as follows:

- Help users move away from a green-screen to a graphical user interface (GUI) environment. Most users are already very familiar with GUIs. They are comfortable with using a mouse and a windows-driven environment. Making the line-of-business (LOB) applications follow that same graphical paradigm increases users' level of comfort with these traditional applications.
- Enable the move from a traditional, multifunction-structured, large program to one that is more modern, which is critical. Moving to modern compilers allows developers to consider adding much more function and integration. These features and functions are only available with the most current language compilers. After the applications use modern compilers, it is possible to start realizing the benefits of implementing the IBM Integrated Language Environment® (ILE) architecture. ILE allows developers to implement some of the basic constructs of object-oriented technology. This includes things such as code modularity and reusability and easier distribution of maintenance activities among development staff.
- Facilitate, when necessary to meet the needs of the business, a move to a service-oriented architecture (SOA). This type of architecture allows applications to talk to each other, both within an enterprise or across organizations.

There are other key points to remember about the System i Developers Road Atlas; however, the strategic objective is to modernize the application code and the skills within an entire development shop.



System i Developers Roadmap V3: 2005 architecture

Version 3 of the Roadmap was used during 2005 and the first half of 2006. As you can see, it provided a clearer picture of the stages of modernization when compared with the previous two versions. It was a clarification of the more-subtle messages inherent in the first two versions. This was the first time the Roadmap included the RPG and COBOL languages (in the ILE form, however) as participants in the processes to support an On Demand Business. It was this version that changed the titles of the Roadmap steps to remove the “better” descriptors and instead reflected the true nature of the activity required in that step. The final two columns (steps) shown here were updated to reflect the industry trends of an On Demand Business, including the concepts of application and business-process integration. The technologies that support these newer ideas were also updated to include Web services as well as connectors and process choreography.

Realities of Roadmap V3

- Showed that modernization was possible with ILE and RPG/COBOL
- Confirmed that RPG and COBOL are strategic languages for IBM and System i
- Illustrated various paths for modernization
- Focused only on modernization activities
  - No representation of new development
  - Limited discussion of SOA
  - No inclusion of industry trends
  - Too much focus on left-to-right (linear) path
  - No real home for many development tools (for example, 4GLs)
- Limited in scope

Realities of Roadmap V3

However, as much as this was a clarification of the modernization messages at that time, there was also still some room for enhancement of the processes themselves. For example, the industry was still discovering new technologies and new application-design paradigms. The Roadmap needed to discover them and include them as well.

ILE implementations of RPG and COBOL can play an integral role in developing a path to modern applications. These languages in their modular or componentized form can participate easily in a full on-demand application structure.

V3 did show that RPG and COBOL are strategic languages for the System i platform; this message remains key to development activity for the System i.

V3 showed two distinct paths for modernization: refacing the user interface as one option and modularization of the code as a second option. The problem with V3 became that it was hard to show the representation of these options in one roadmap with one left-to-right pathway.

Although it was thorough in its handling of the modernization message, V3 of the Roadmap did not incorporate other normal development activities, such as the development of new components or the integration of components from other sources. As well, the industry trend toward a service-oriented environment was not reflected specifically in the steps. And of course, new technologies had surfaced during the 18-month duration for V3 that needed to be added to the Roadmap to keep it current and useful.

Lastly, the old Roadmap with the columnar representation of the steps forced the IBM modernization-support team to remain limited in the scope of what could be included. By opening the structure in a more multidimensional manner, it would become easier to include future development technologies and new industry directions.

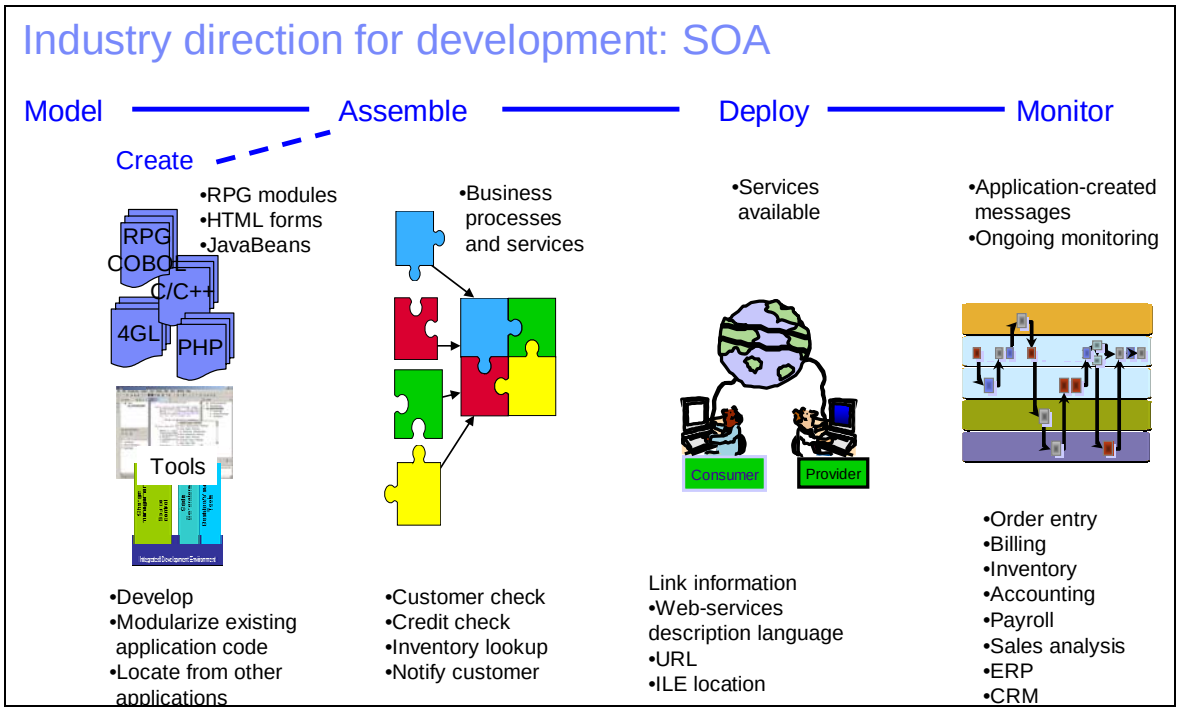
Going on vacation?

It is easy to draw an analogy between using the System i Developers Road Atlas and planning for a summer vacation road trip. What are all the considerations involved in planning for such a trip?

You need a good road atlas that shows all available roadways. You must determine where to go (your destination). If you have no idea of where you are going, it does not matter which road you take, and you better be prepared for a lot of surprises. You also need to know where you are located right now. If you do not know your current location, then a map is useless. You need to determine the details about the journey. For instance, is speed a concern? Do you need to make checkpoint stops along the way to confirm that everyone on your team still wants to continue on the journey? Or, is there another, more appropriate option? Is a rest stop a good idea? These are the decisions that you need to make; some are tactical choices and others are strategic.

After you know your destination and have determined your present location, you can successfully plan your trip, understanding the requirements for reviewing the status and goals and check-pointing along the way.

The same is true when building a path toward good applications. You have to understand the trends in the industry. On a vacation trip, you want to know the best destinations. The same is true for IT where you need to determine the destinations that are known to accommodate the requirements of your business.



Industry direction for development: SOA

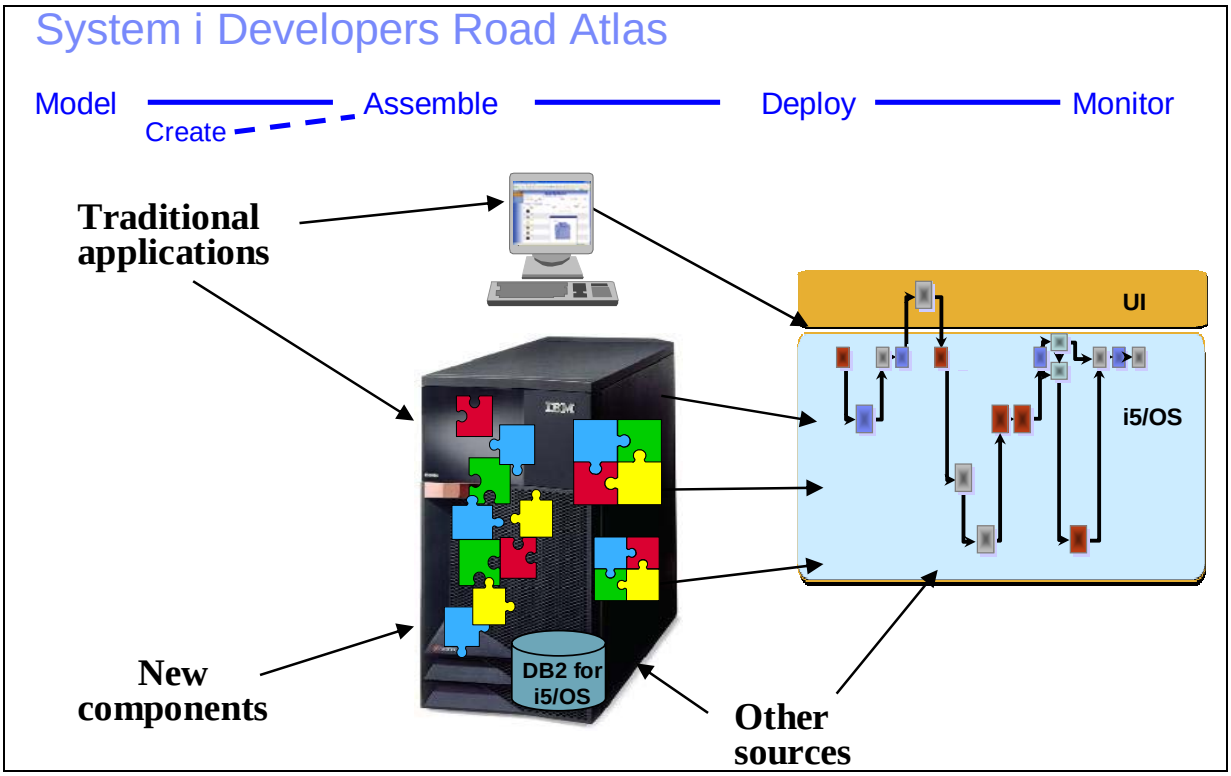
This chart shows a sample of the technologies for creating an SOA application. (**Note:** It does not imply which technologies are more strategic or important.) The SOA is intended for designing and building applications and involves four steps:

Model: The first step in creating an environment for your application that matches the key rationale for implementing SOA is to create a model of how the finished application will work. An application architect usually designs this model for your organization by laying out the key ingredients for the application from the perspective of a business function.

Assemble: The newly created model can be populated from its component parts. Consider the required function, which can comprise one or more coded modules. Some component parts might already exist, others must be created. Do not think of the components as individual RPG modules or Java™ components. The modules can be created in any language, by any tool and can be located on your System i model or on another server within the organization (or even in the external community of trusted partners).

Deploy: The next stage is to deploy the application by moving the components into the IT production environment. This might require the definition of new access methods (such as Web services and location information for system components). This information is needed by the application architects. From this information, the next stage occurs: the building of the application.

Monitor: The last stage of an SOA environment involves monitoring the application as it runs. Components might be swapped into or out of an application as newer business practices develop. A good SOA design supports an uncomplicated swapping of old-for-new components (independent of the technology used, the coding language or the platform where the component runs).

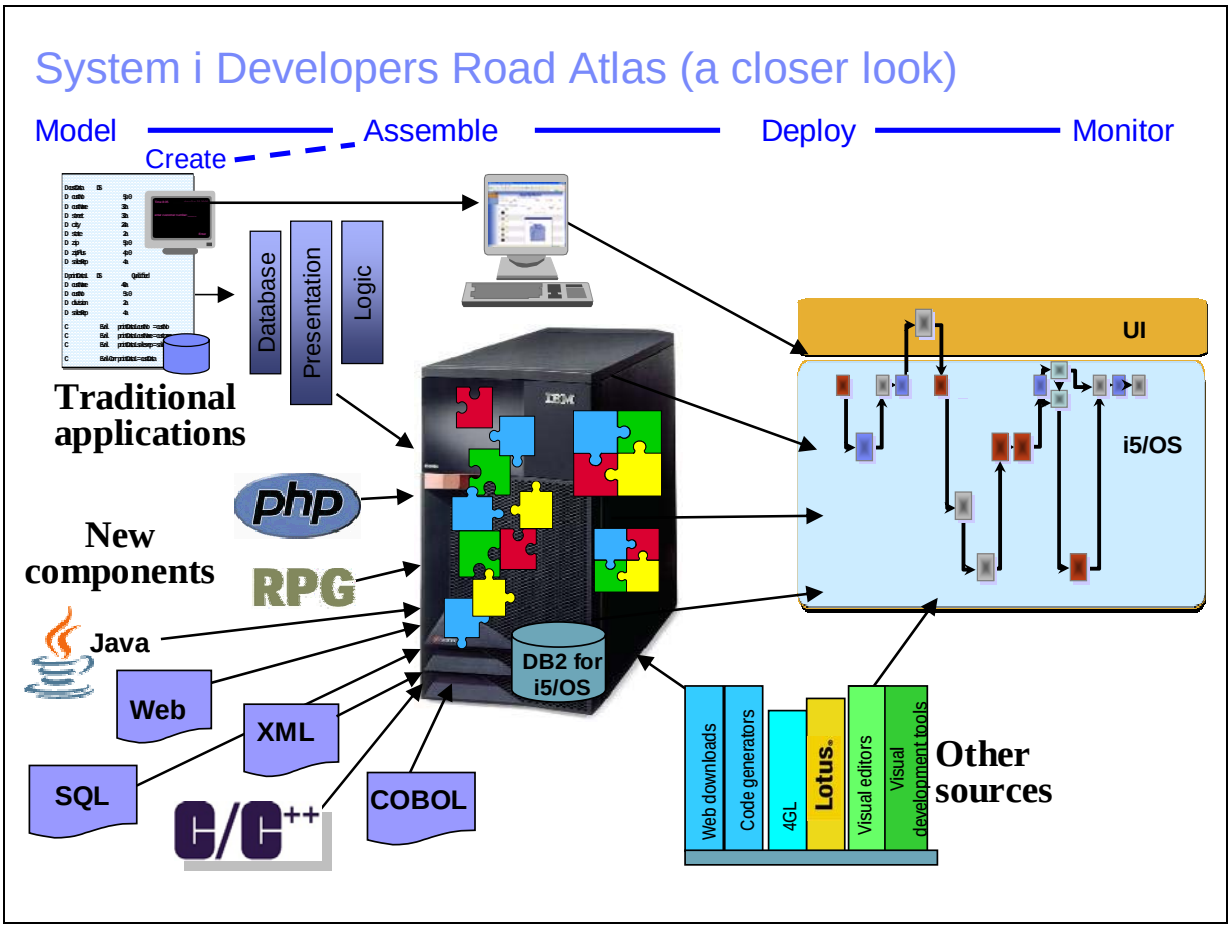


System i Developers Road Atlas

This chart is the System i Developers Road Atlas. Notice that it is not being called the Roadmap any longer. The connotation of a road atlas more accurately implies the flexibility to lay out all the possible routes for creating applications on the System i platform. Many choices are available. Just as with a road atlas for your vacation trip, the System i Developers Road Atlas is comprehensive and, therefore, a little confusing at first glance. However, it is possible to chart your course by following the steps that were laid out earlier:

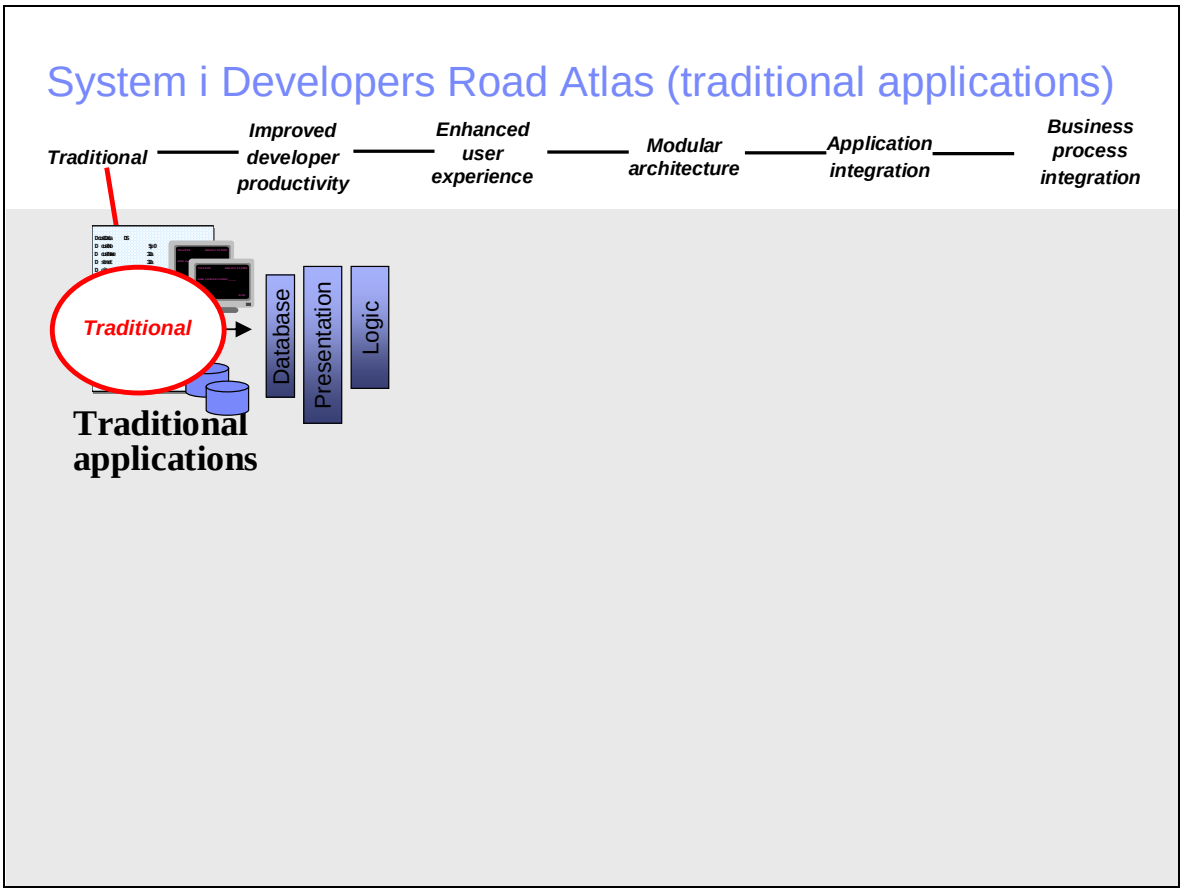
- Determine your destination
- Determine your current location
- Decide the method to get there; consider both strategic and tactical options

There are three main starting points. However, components gleaned from any of these positions meld together through the use of good SOA to create an on demand, flexible application. The key is the architecture.



System i Developers Road Atlas (a closer look)

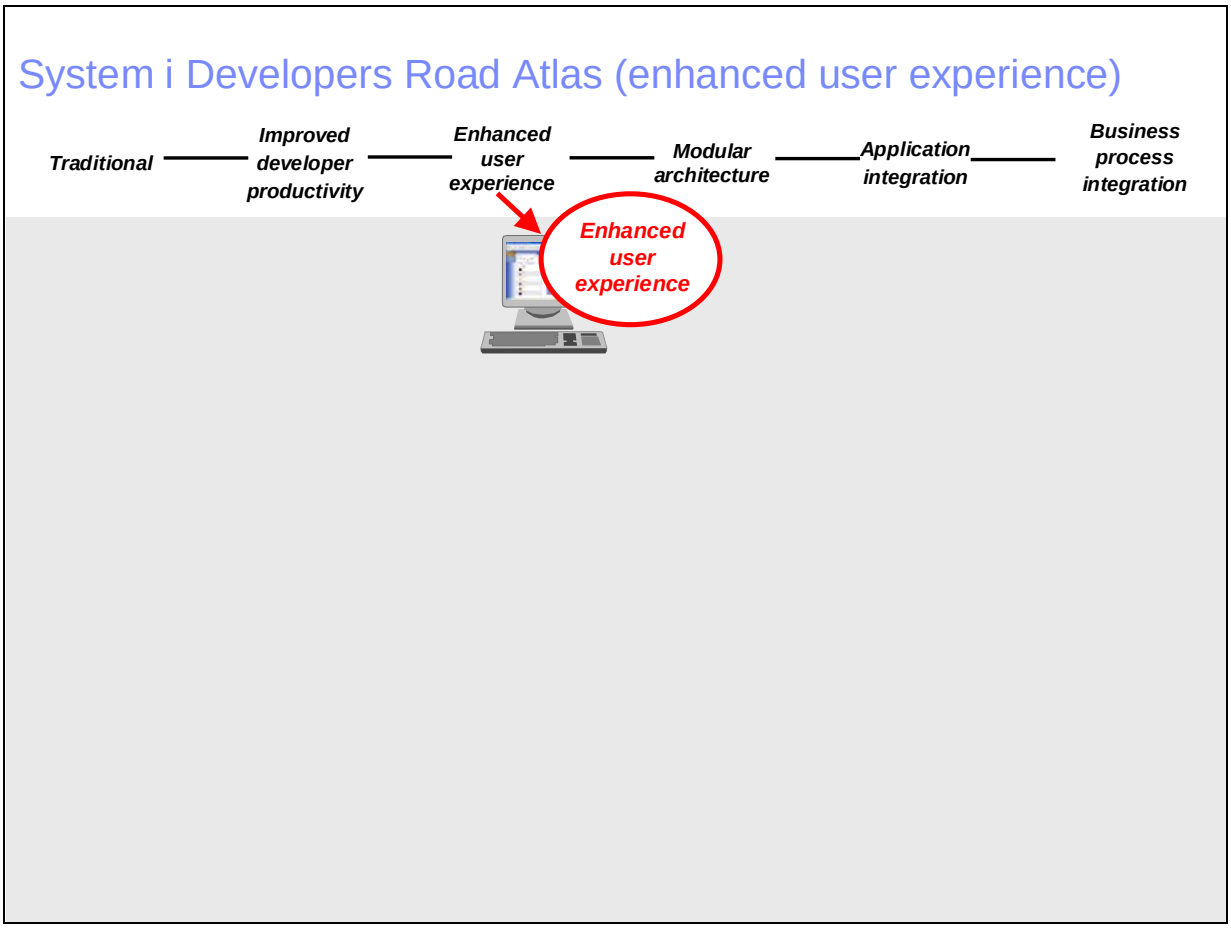
This chart illustrates (in greater detail) the various options that fall into each of the major starting points.



System i Developers Road Atlas (traditional applications)

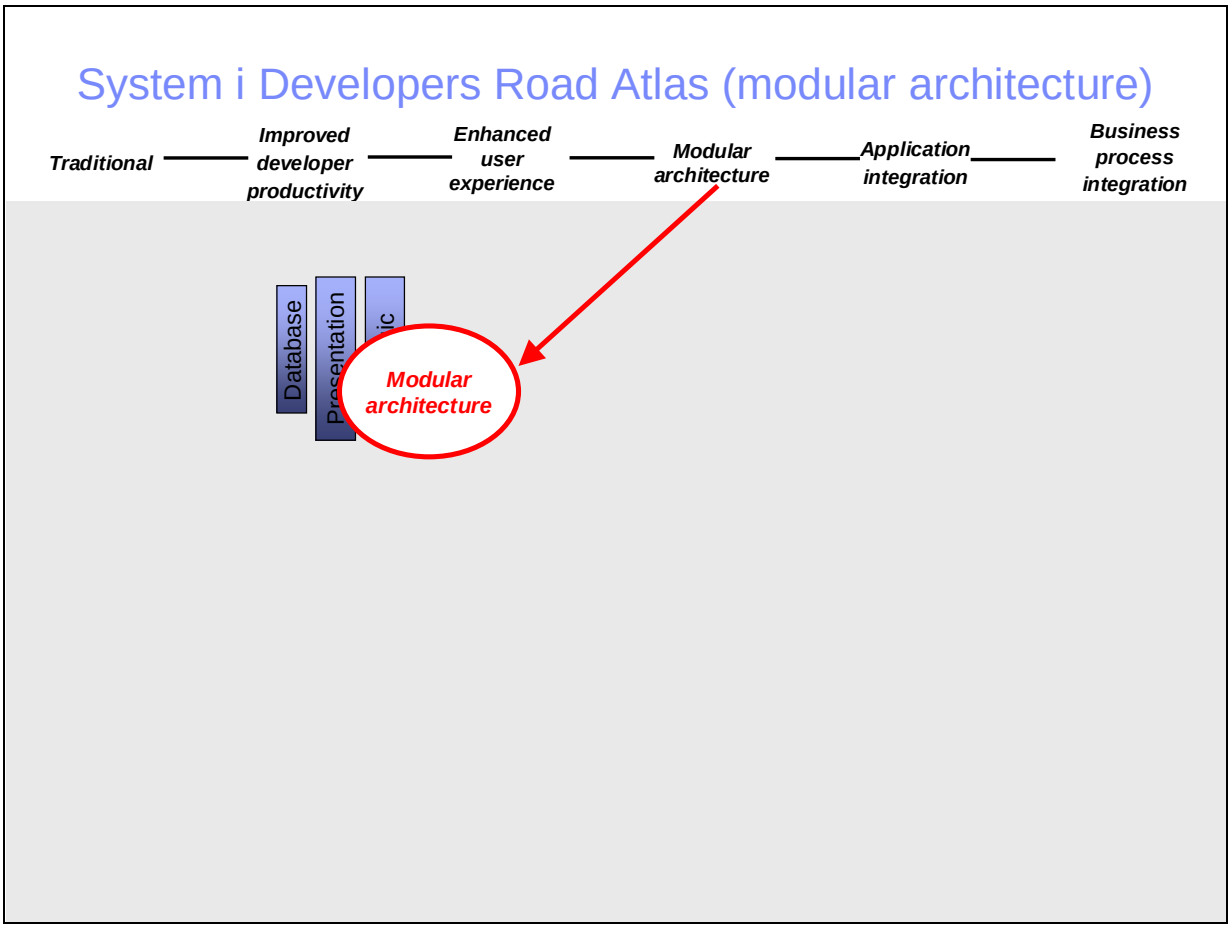
To assist those who are familiar with and understand the columns (pillars) of the System i Developers Roadmap V3, glance briefly at the mapping of this earlier version to the Developers Road Atlas V4.

Notice that V3 traditional applications are still traditional applications in V4. The starting point is the same.



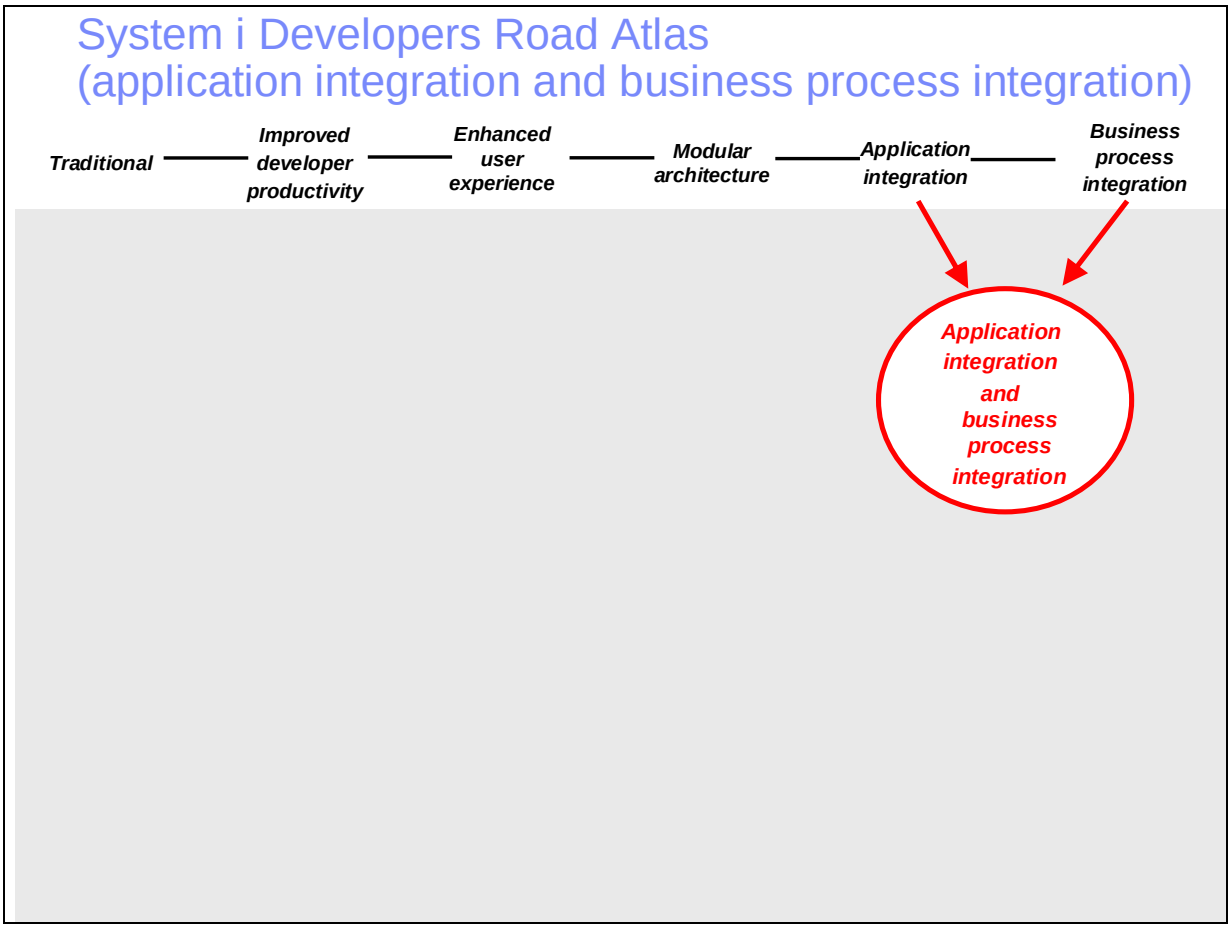
System i Developers Road Atlas (enhanced user experience)

The V3 enhanced user interface is still in the System i Developers Road Atlas V4. It is the process of putting a new face on existing application code. It is noninvasive to the underlying code.



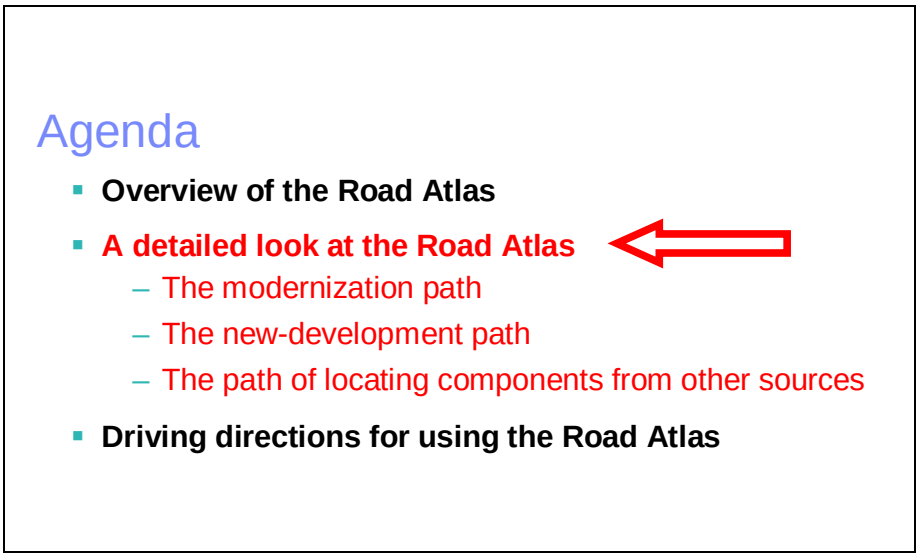
System i Developers Road Atlas (modular architecture)

The V3 modular architecture is a key piece of the System i Developers Road Atlas V4. However, it is now positioned so that it is independent of the refacing technologies. You will understand the reasons behind this separation later in the presentation.



System i Developers Road Atlas (application integration and business process integration)

Previously, the columns of the roadmap called Application Integration and Business Process Integration were separated by where the integration happened, either inside or inside and outside your enterprise. In this version of the roadmap, SOA is the integration point for applications. This means that it becomes irrelevant as to where a component resides and the technology that it uses. This is a much cleaner and more realistic view of distributed business functions.

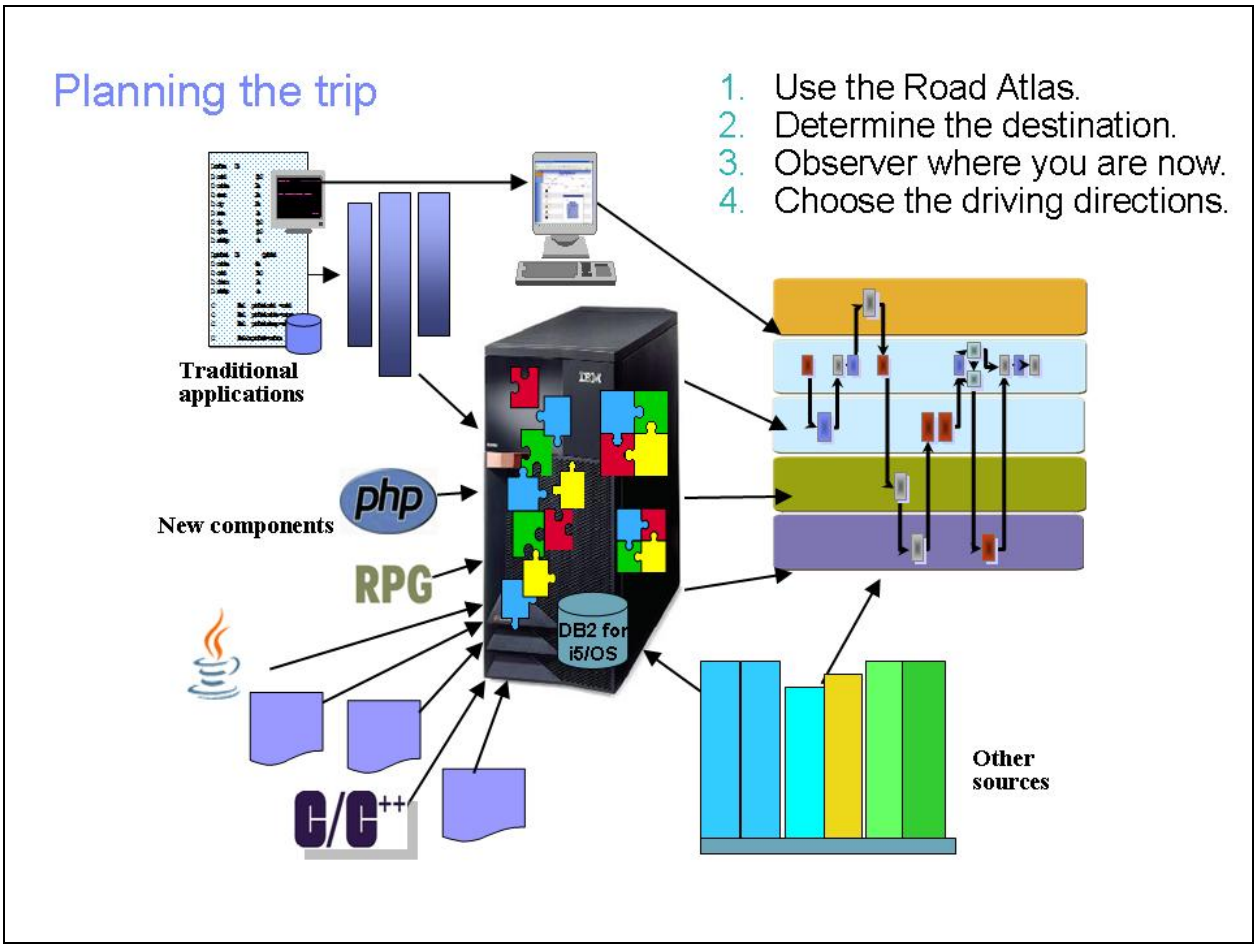


System i Developers Road Atlas (the details)

Now, you will take a closer look at the detailed graphic of the new Road Atlas and the pathways that are available with this enhanced version. This Road Atlas looks a little more complicated than the previous versions, but it can be broken into three, easy-to-understand paths:

- The modernization path
- The new-development path
- The path of locating components from other sources

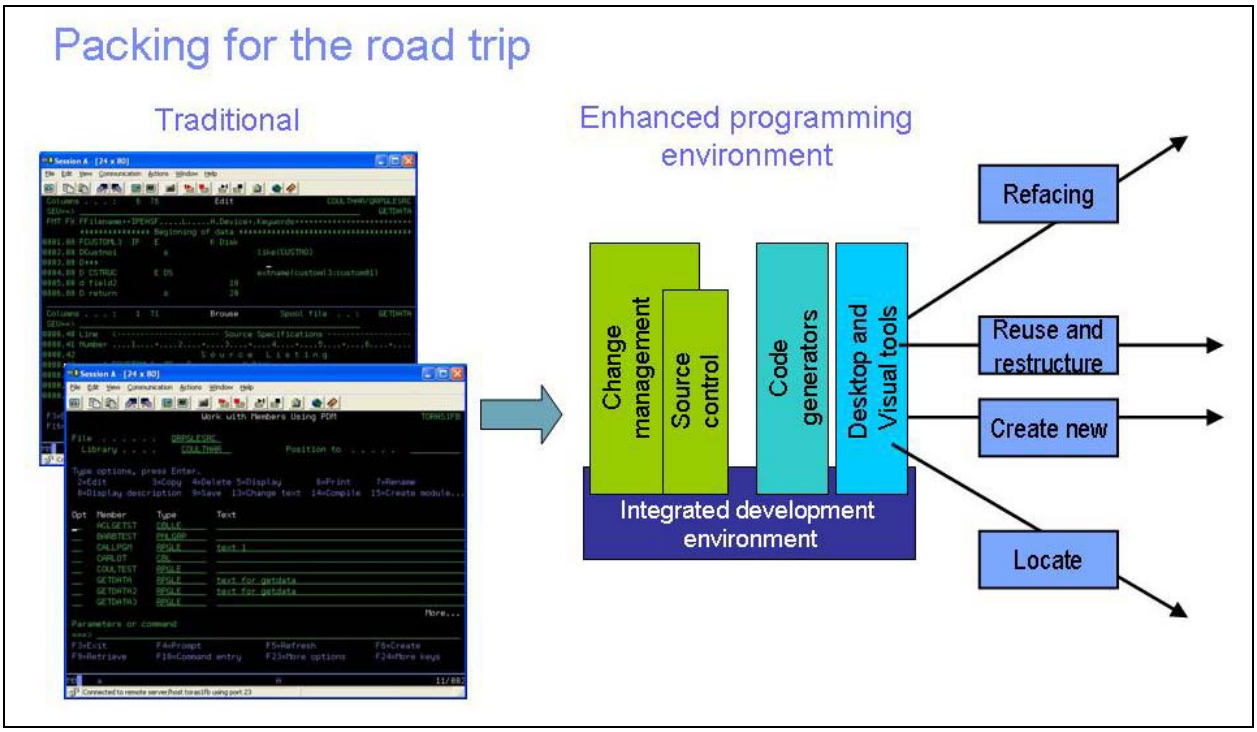
Each of these paths are discussed in more detail on the following charts.



Planning the trip

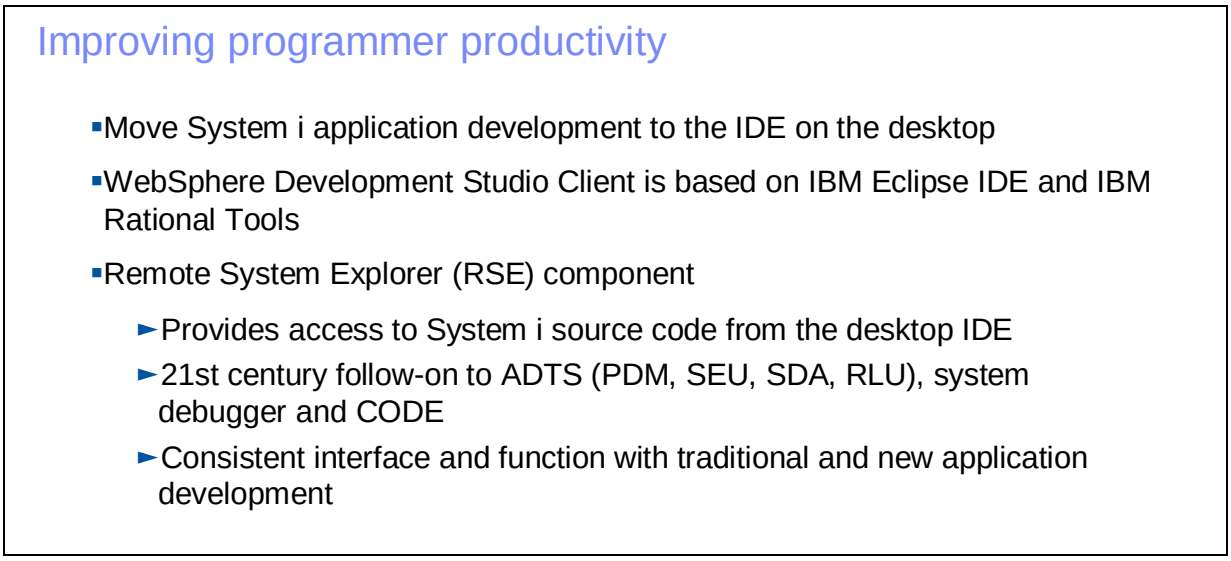
Look at this diagram to see how you might use the same “planning a vacation trip” methods to design a modernization course for your System i applications:

1. Find a good road atlas.
2. Determine the destination.
3. Determine where you are now.
4. Create some driving directions on how to get from your present starting point to your desired destination.



Packing for the road trip

Just as with a vacation trip, preparing to take the trip is important. On a vacation, you pack your suitcase before leaving. For a development road trip, you need to pack your bag with things to help you along the way. In this case, it includes a fully functional suite of development tools that run on a desktop. This is called an integrated development environment (IDE). In today's world, the most common one is the Eclipse IDE. This is an open-source platform with a robust suite of built-in Java tools. If your shop is predominantly a Java road trip, then the Eclipse IDE might work just fine. However, if you require some specific System i extensions, then you need to load the System i tools suite called IBM WebSphere® Development Studio client. This provides you with an IDE for developing RPG and COBOL code, as well as for data-definition specifications (DDS).



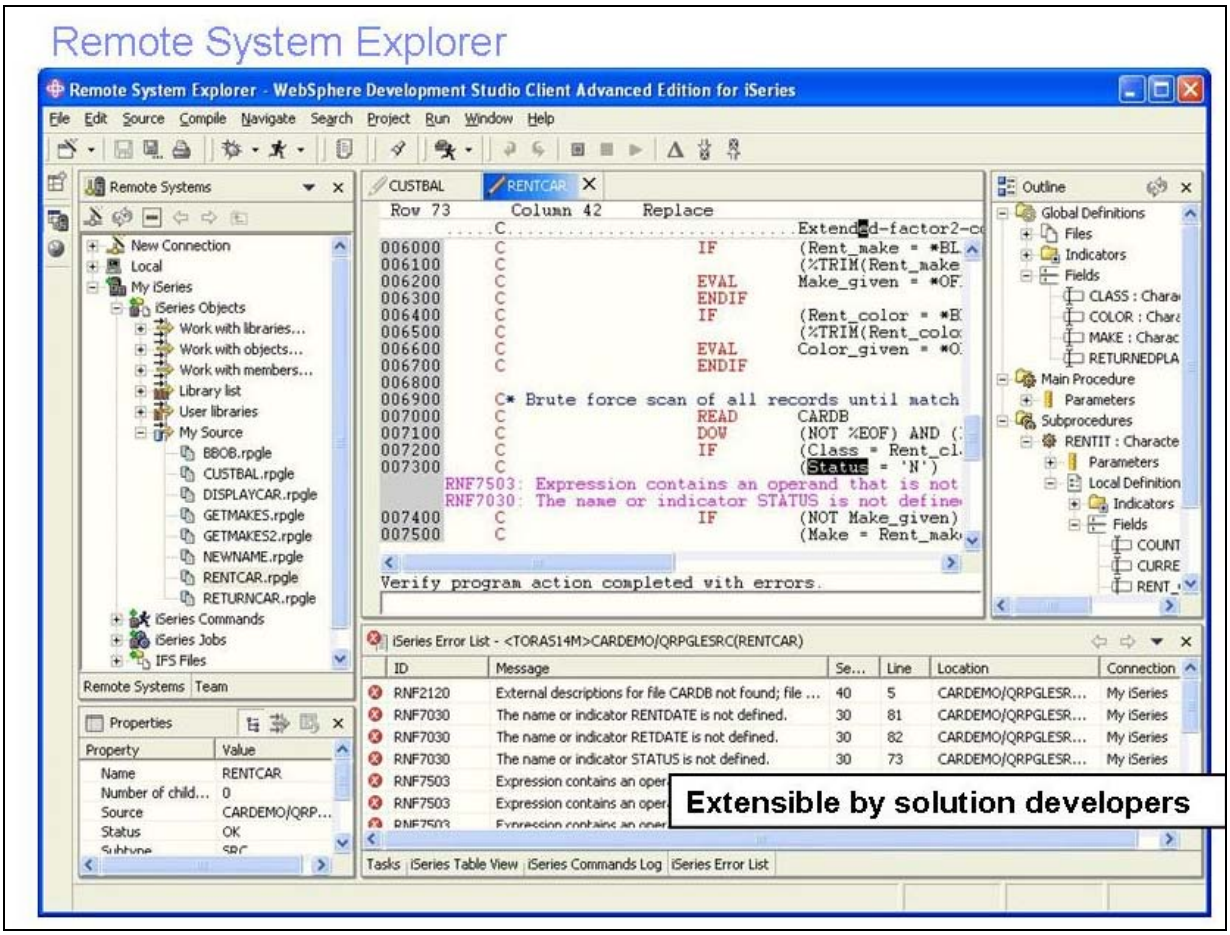
Improving programmer productivity

The first step in the System i Developers Road Atlas does not require any changes to the applications in use today. Rather, it involves replacing the traditional development tools with more functional and modern tools to support the same code base. Ultimately, the application code is still written in a traditional language (such as RPG or COBOL) and still has a green-screen user interface that uses DDS.

Remote System Explorer (RSE) is the 21st century follow-on to Program Development Manager (PDM), Source Entry Utility (SEU), Source Design Aid (SDA), Report Layout Utility (RLU) and the system debugger. It offers a very productive environment; each development component is highly integrated with the others and with the Eclipse IDE. RSE is also the point of integration for all System i tool vendors, many of whom have already released plug-ins to complement the functionality provided by IBM.

RSE comes with WebSphere Development Studio Client.

Many IBM vendor tools might fall into this area, including tools to assist developers with change management, utilities and documentation, printing, debugging and testing.



Remote System Explorer

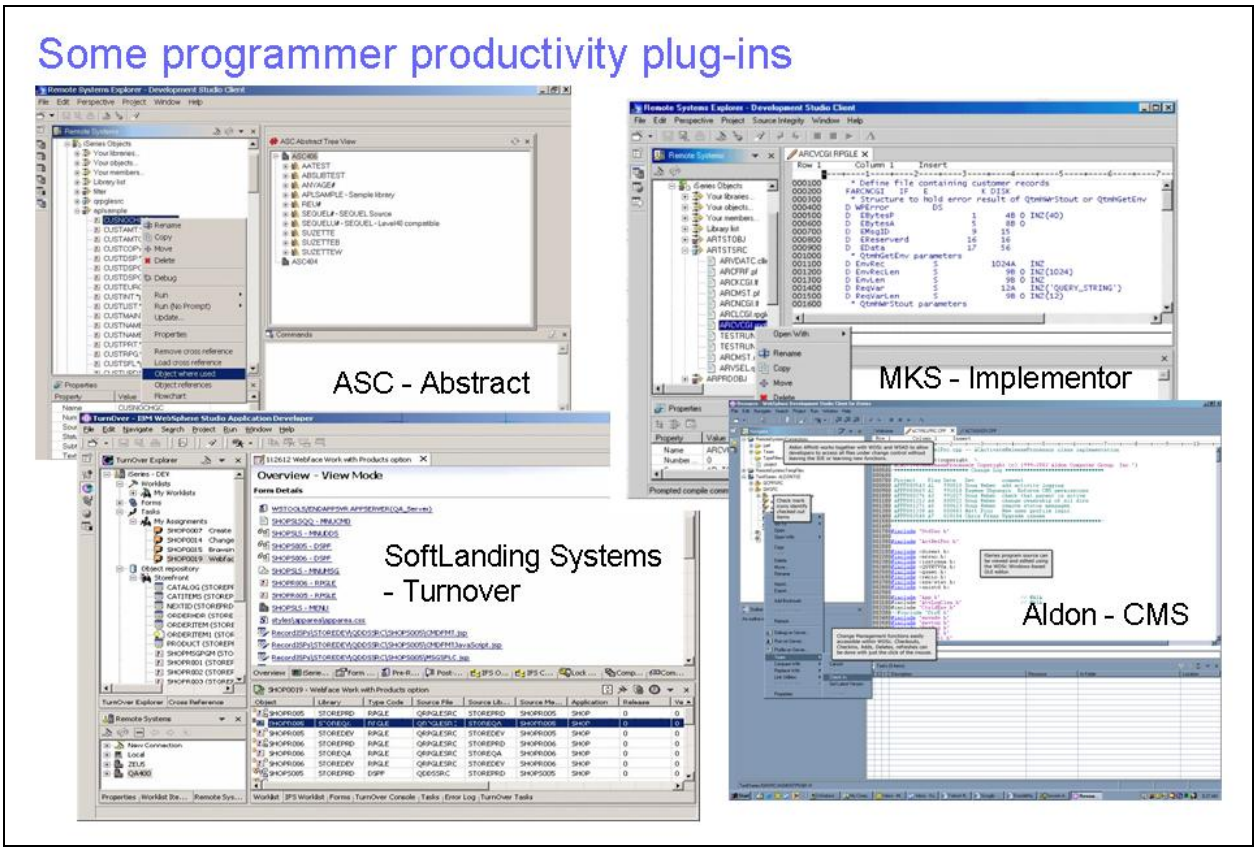
This screen capture shows the collection of views and editors in the RSE perspective, where you create connections to remote System i, UNIX®, Microsoft® Windows® or Linux® servers. Developers can open multiple perspectives and flip between them using icons in the bar on the left.

There is a connection to a System i model where IBM i5/OS® objects are expanded so you can work with them; you can also work with members in a similar manner to PDM. Look at the “My iSeries” tree to notice that you can also work with commands, jobs and the integrated file system (IFS). As you select objects in the Remote Systems view, the property sheet (lower left) shows information about the selected object (which, here, is RENTCAR). Some of the object’s property-sheet information is directly editable for your convenience.

Many right-click actions are provided for all object types and members, including source members. The option to open the member in the RSE editor is shown on this screen capture. Beyond what IBM supplies, user-defined actions are possible. The editor is rich in function, far exceeding SEU although retaining features such as entering **D** in the prefix area to delete a line. The editor supports syntax-checking and cursor-sensitive F1 language help. It also has a built-in program verifier for RPG, COBOL and DDS, which does a full error check and reports the results in the IBM iSeries™ Error List (shown in the bottom right). Double-clicking an error in the list automatically positions the cursor in the editor at the offending line and, optionally, inserts

the error lines for context, also as shown here (error numbers RNF7503 and RNF7030). The same error feedback is used when the member is remotely compiled.

The top right shows the outline view (an at-a-glance hierarchical picture of the source member that is active in the editor). This is handy for understanding the program and for navigational purposes. The editor supports content assistance for RPG, which allows you to press Ctrl+Spacebar to acquire a list of available options that relate to the cursor‘s current position. Tabs are at the bottom of the panel for some of the many other views, such as the iSeries Table View. It is an alternative to the tree view, offering a PDM-like, sortable table view. As with PDM, this is where the command line exists for entering and running IBM OS/400® commands. The iSeries Command Log shows all commands that run explicitly or implicitly for this session. And, as with other Eclipse perspectives, you can rearrange the views to fit your preferences.

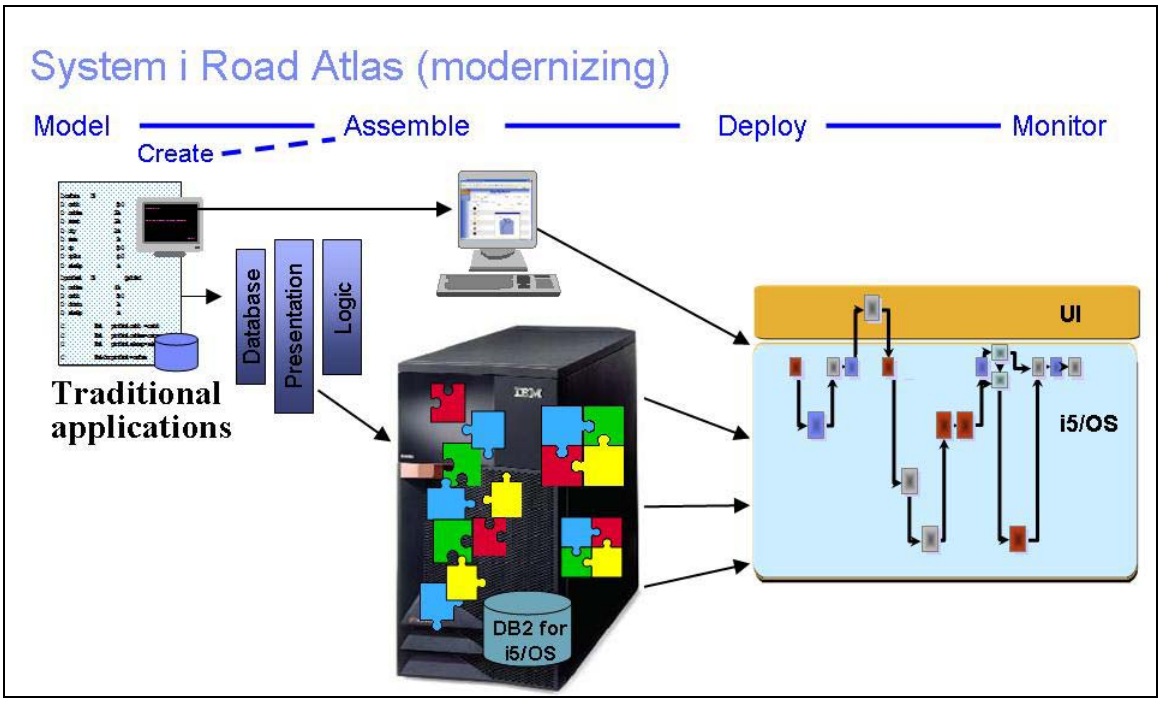


Some programmer productivity plug-ins

This chart shows four of the many vendors who have produced plug-ins to the WebSphere Development Studio Client desktop environment. The three change-management tools and the ASC abstract tool plug in to the IDE. This provides a huge productivity boost to programmers who work in the IDE. It means that all development project activities can occur in the same environment without exiting to the green screen to do things such as checking source code in and out, creating projects or promoting code up the project hierarchy.

IBM tool vendors also have plug-ins to the WebSphere Development Studio Client that offer more function to developers. IBM validates the claims of the vendors and lists the validated companies and products on the Ready for Rational Web site. (The Links section of this course provides a link to this site.)

WebSphere Development Studio Client is built on top of Eclipse and on top of the IBM Rational® ToolSet. Therefore, many of the plug-ins for either of these environments also work with the WebSphere Development Studio Client product. The list of Eclipse plug-ins is called “Plug-in Central.” (The Links section of this course provides a link to this site.)



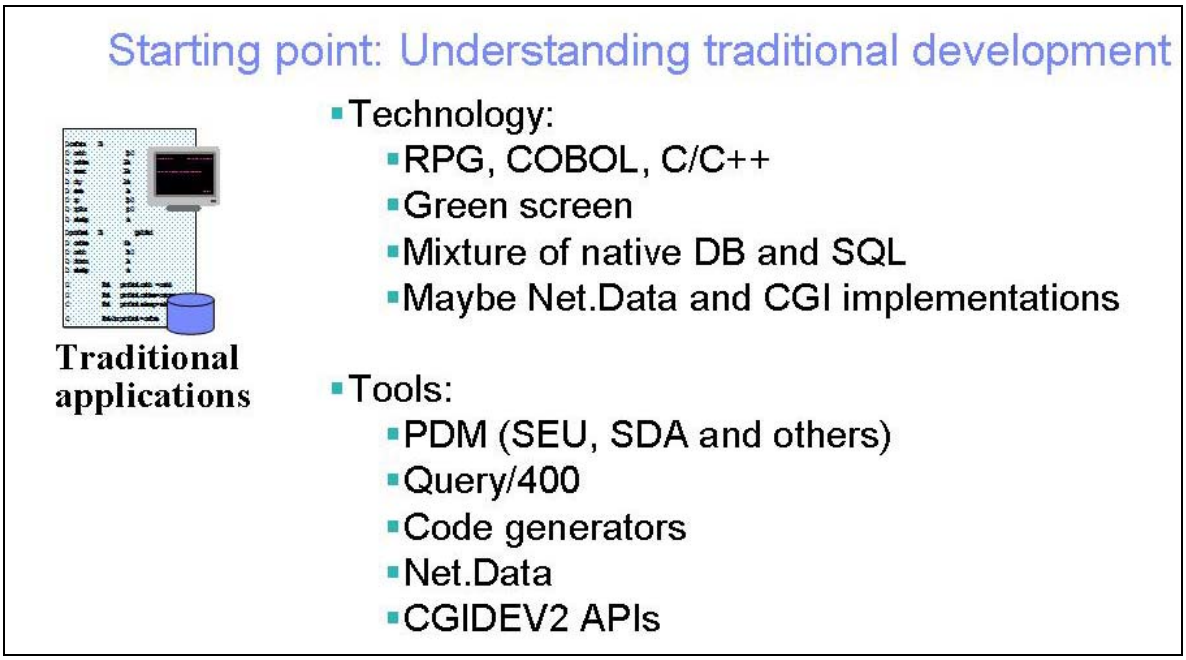
System i Road Atlas (modernizing)

This is the same modernization pathway that has been evolving over the past few years. It begins with traditional applications and ends with the implementation of an application in an SOA environment.

The main difference from previous versions of the modernization pathway is that the Road Atlas V4 has two distinct paths: one for *refacing* and another for *modularization* of applications. This split was the result of feedback from IBM solution providers and clients who explained that, although refacing is sometimes used as a tactical strategy to buy time for the more-invasive solution of modularizing, it often does not provide reusable components. V3 portrayed these two activities as sequential steps, but reality has proven that they are not.

The refacing path: This path shows how to provide an alternative interface to an existing application that, otherwise, requires minimal if any changes to the underlying application. There are various ways to implement a graphical image on an existing application. This can be done using a browser interface, a thick client, a thin client or a variety of combinations in between. In all cases, the traditional approach of using a 5250 green screen is replaced with a user interface more like Web or PC-based applications. Many tools from IBM and other vendors can provide this type of GUI.

The modularization path: Following this path means making aggressive alterations to your applications to separate the function in your code. It means separating business logic from data access, from the user interface and, in some cases, from the printing modules, as well. Creating modular code allows developers to implement many of the constructs of modern programming techniques (such as reusability). Only a few tools can assist in this area. Some of them simply move traditional code to a more modern compiler (such as moving from RPG III to RPG IV structures. A few other tools read the application code and assist developers in understanding their application programs (business functions, user-interface components and others).



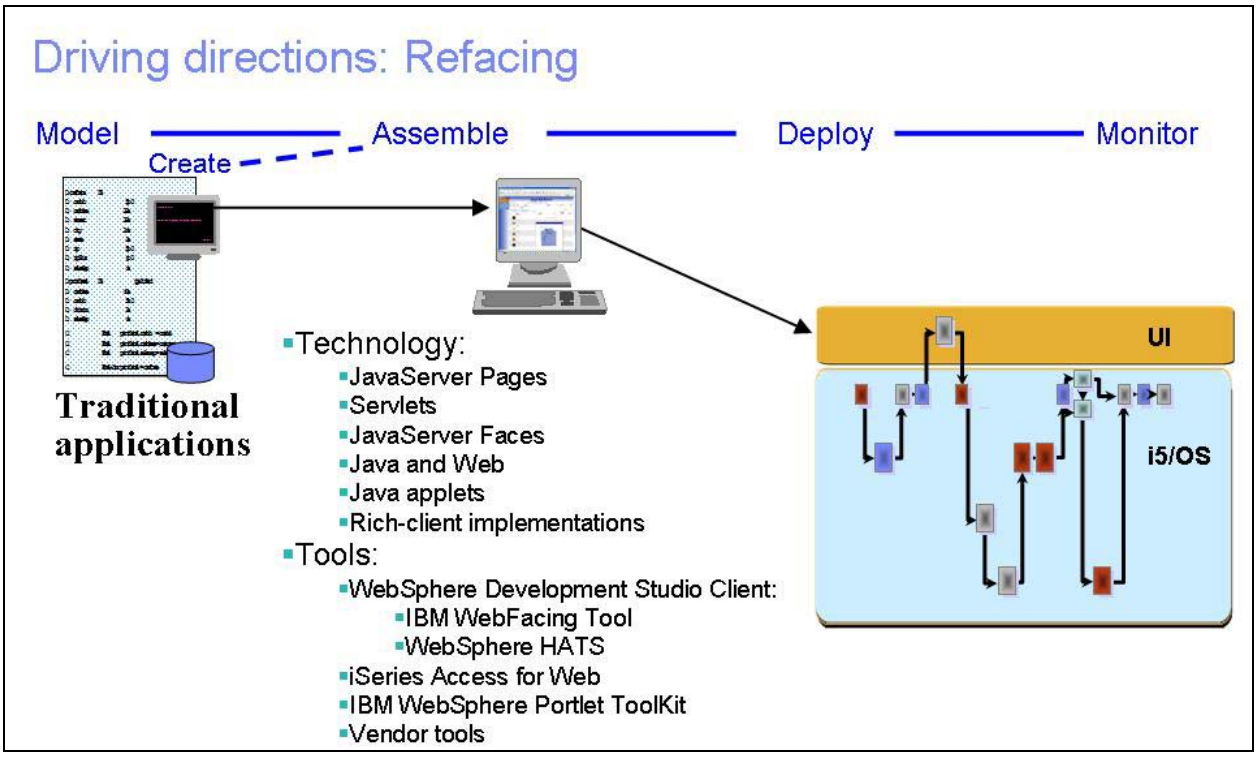
Starting point: Understanding traditional development

The first and very critical task is to determine the location from which you are starting your new development effort.

Many System i shops have code that has existed for some time. RPG and COBOL programs were written to reflect the standards in the industry during the 1970s and 1980s: monolithic, top-down, structured code. These applications are based on green-screen interfaces and might have implemented techniques that were contemporary to that period, such as using nonprogrammable-terminal user interface (NPTUI) drop-down and pop-up windows. Applications typically use traditional mechanisms for data retrieval such as native-language read and write tasks. They might use SQL, but usually only for query types of activities.

The tools used to develop these applications were also driven by green screens and based on the IBM Application Development ToolSet (ADTS) utility suite. Most programmers are familiar with SEU, SDA and other tools. These tools were shipped with the initial delivery of all IBM AS/400® systems in 1988. The last functional enhancements to the tools were added approximately 12 years ago with OS/400 Version 3 Release 6. The IBM laboratory in Rochester, Minnesota, now refers to these tools as “stabilized” and is not enhancing them any further.

Sometimes, developers have also used code generators and tools such as IBM Net.Data® or the IBM CGIDEV2 APIs. Each of these technologies has had success in the traditional environment and can certainly be used to maintain applications. However, Common Gateway Interface (CGI) and Net.Data, which is built on top of CGI, are not strategic in the IBM portfolio of development environments. Net.Data and CGIDEV2 are both supplied on an “as-is” basis and are supported for defects only. No new IBM development effort is allocated, meaning that there no planned new enhancements (in the form of features and functions) to these tools.

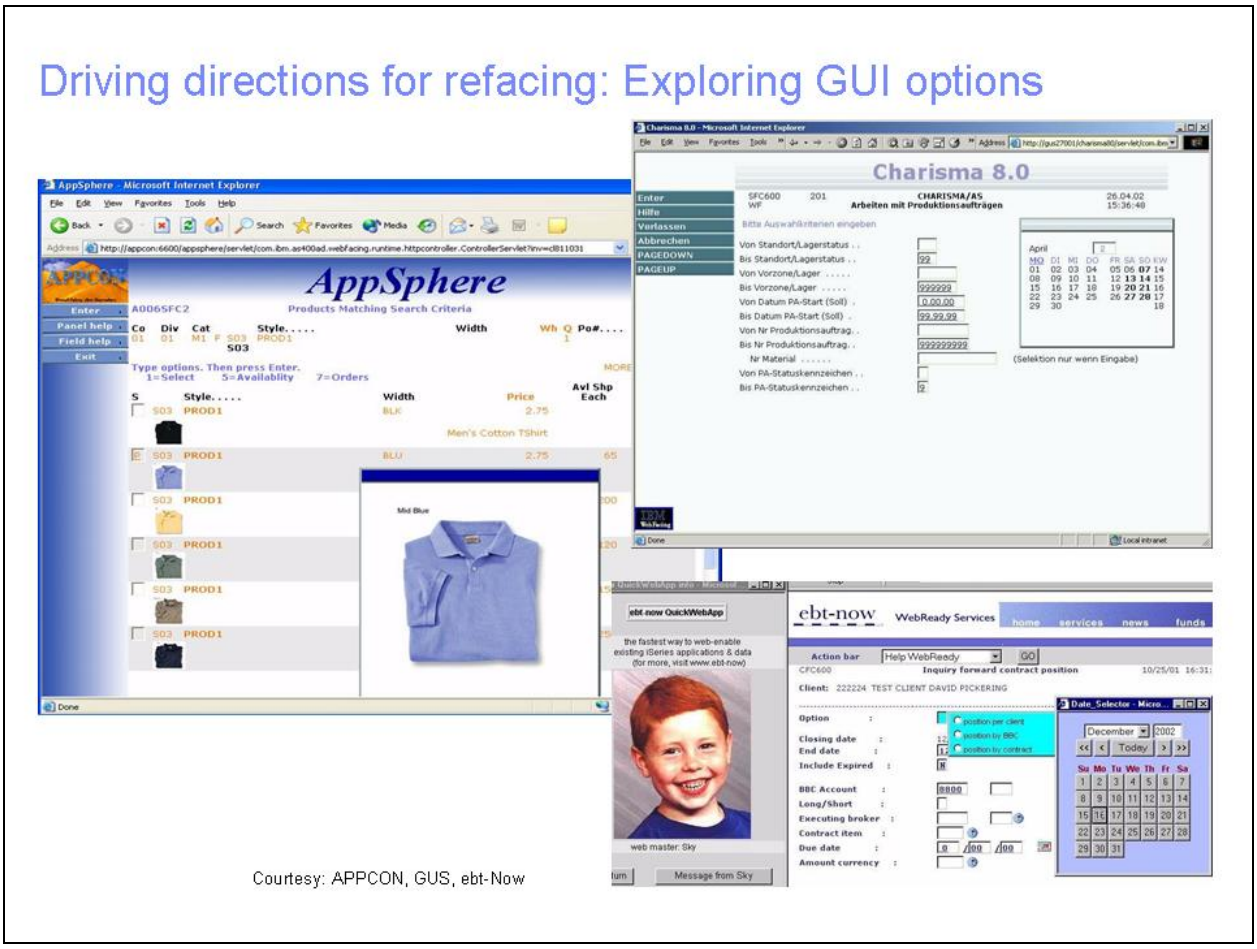


Driving directions: Refacing

From the starting point of a traditional application, there are two possible paths for arriving at the service-oriented destination.

The first path is to put a graphical interface on the existing application. The available technologies allow a range of options for implementing a graphical interface. Thin-client and rich-client designs offer a wide range of implementation techniques, depending on the requirements of the application and the skill of the programmers. Browser-based interfaces typically allow you to add a graphical interface without changing the underlying traditional application. This is done by using technologies such as JavaServer Pages™ (JSP™) and JavaServer™ Faces (JSF™), as well as a Java servlet to drive the activity. Rich-client options range from implementing business logic as well as rich-graphical capabilities in logic that resides on the desktop. Many developers remember the trend toward client/server from previous years. Today's Java technology-based client solutions offer a much more evolved and easier-to-manage version of the original client/server model. The code for business logic and graphical presentation resides on the desktop and interacts with the server-side code.

Regardless of the method used, a variety of tools can assist you in building the user interface. This course will explore those options in more detail. Many of the tool options also support your need to extend the application to create or consume Web services. The business requirements for the solution drive the decision of whether to participate with Web services or to leave the application as it is and instead, assist you in simply creating an enhanced user interface.



Driving directions for refacing: Exploring GUI options

Improving the user experience refers to changing the user interface from a green screen to a GUI. This step of the Road Atlas is noninvasive to the underlying application code. Typically, no changes are required to accomplish the task of adding the graphical interface. The GUI can be achieved by many means, including screen-scraping or redirecting the user-interface data stream to use a different rendering. A third GUI option involves developing a rich client that communicates with the server-side application. All of these vehicles accomplish the same task of creating a more modern look and feel to the underlying application code.

The other technology that begins to come into play here is *portal technology*. After an application has moved beyond the green-screen into a graphical interface, it is possible to allow users to tailor their Web interfaces to work with portal products. Portal technologies allow the personalization of each user's browser space, supporting the individual's desire to have a subset in the browser window and to show various applications within each window. (Another chart illustrates this more clearly later in this course.)

Driving directions for refacing: Creating GUI for 5250 applications

- Add a GUI to existing application code
  - Browser-based GUI
    - IBM WebFacing Tool: Development conversion
    - WebSphere Host Access Transformation Services (HATS): Runtime conversion
    - iSeries Access for Web: Runtime conversion
    - JavaServer Faces
  - Rich-client GUI development with WebSphere Development Studio Client
  - Portal Interface exploits browser real estate and lets you tunnel to specific function
  - XML
- Avoid impact to underlying application logic
- Add function with new technologies (XML, Java) with no impact to existing programs

Driving directions for refacing: Creating GUIs for 5250 programs

This step pertains specifically to creating a better user interface for an existing application. There are three IBM options for refacing an application to support browser-based interface technology: the IBM WebFacing Tool for i5/OS, IBM WebSphere Host Access Transformation Services (HATS) and IBM iSeries™ Access for Web. All three tools produce a Web-user interface from a 5250 user interface (UI); there is no impact to underlying application logic. They generate UIs that run on IBM WebSphere Application Server - Express, on any operating system that supports WebSphere Application Server. When in a browser environment, any of these generated UIs can also run inside a portlet; this helps to customize the browser space. Lastly, a rich-client interface can be handcrafted by using Web and Java tools, or it is possible to use the rich-client development environment within the WebSphere Development Studio Client tools. (**Note:** Subsequent pages provide more detail regarding WebSphere Development Studio Client.)

The IBM WebFacing Tool for i5/OS converts Display File (DSPF) DDS source, at development time, into a Web application that uses JSPs. The IBM CODE Designer tool refines the conversion to add Web settings (through special comments) into the DDS source, which affects the result of the conversion. For example, fields can be hidden or replaced with HTML tags, where the tag contents are derived from the field contents. For example, you can send an IFS image file name to a hidden 5250 field, which is then converted to an HTML image tag to show the image in the Web page. CODE Designer is the follow-on to SDA and offers a 5250 “what you see is what you get” (WYSIWYG) view of the application UI.

The IBM WebFacing Tool for i5/OS is part of, and leverages the other tools in, the WebSphere Development Studio Client (including RSE) for RPG and DDS logic. Applications created with this tool do not produce a 5250 data stream. Rather, the i5/OS runtime environment intercepts the data written by the application to the record formats and, instead, sends it to the WebFacing Tool servlet, which inserts it directly into the generated JSPs. These applications do not use interactive cycles when run on any System i models delivered after January 2003. And, an important cost-related note, there is no runtime fee for these applications.

HATS, another IBM refacing option, converts a 5250 or 3270 data stream, at run time, to a browser interface that runs in WebSphere Application Server. Because it is a runtime

conversion, it instantly transforms screens to run in a Web browser. In a repeatable manner, HATS developers can easily refine the conversion results to improve the Web UI. The HATS development environment plugs into, and ships with, IBM WebSphere Development Studio Client for i5/OS. The runtime environment for HATS is IBM WebSphere Deployment with HATS Technology. (**Note:** There is more information on subsequent charts.)

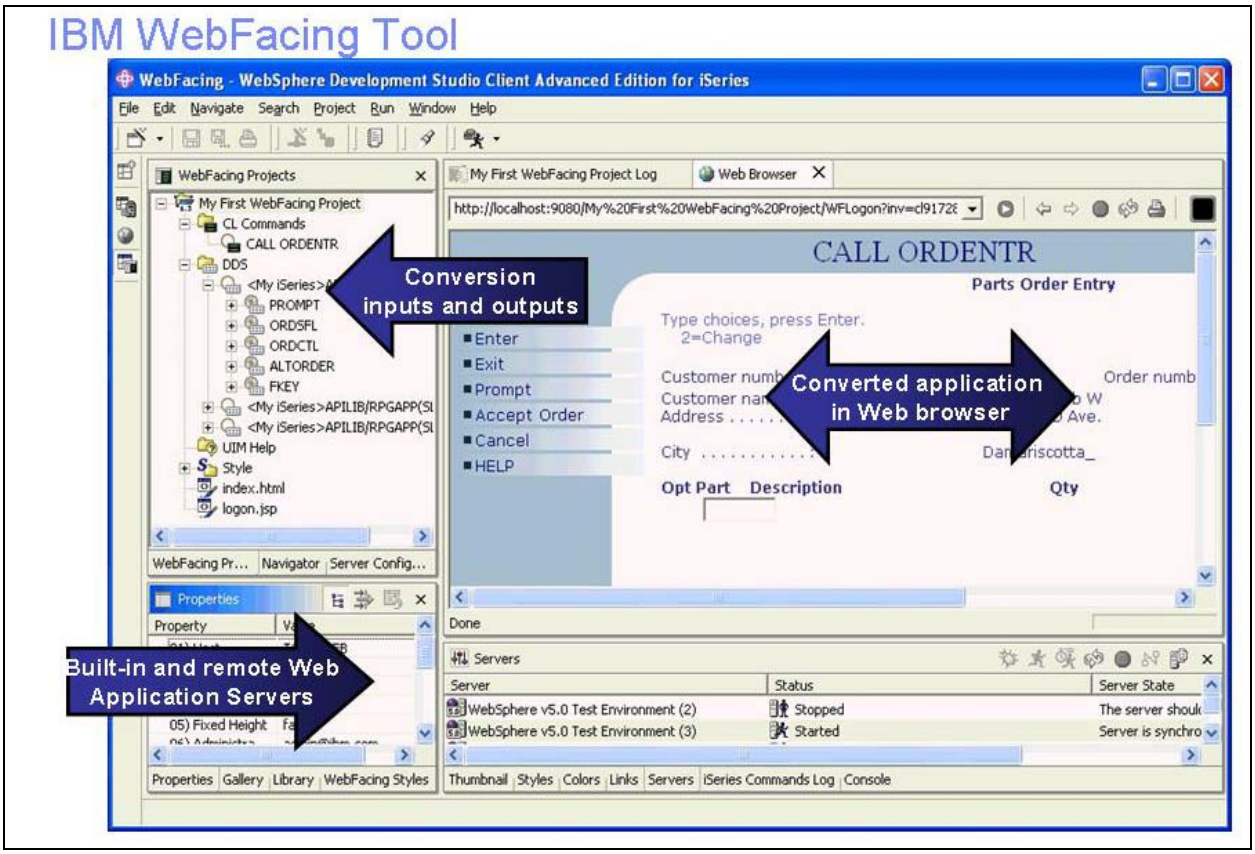
At first glance, iSeries Access for Web seems similar to HATS in its implementation. iSeries Access for Web does 5250-to-HTML conversion at run time, as does HATS. However, one of the key strengths of iSeries Access for Web is the other things that it does, in addition to data stream transformation. It has many operational capabilities that allow a user to browse job and output queues, display-message queues and others. While browsing a spooled file, you can view the output in PDF form and then e-mail it other users. It is a powerful tool for remote operations, as well as for its graphical-transformation value.

IBM WebSphere Portal Server runs inside WebSphere Application Server and allows individual users to customize their browser space. The two main ways that this works are as follows:

- It allows you to subset the browser real estate.
- It lets you simultaneously monitor multiple applications in the browser.

The other option available with a portal solution is to create a window on a specific application (or, a tunnel if that is an easier way to think of it). The tunnel provides a preconfigured and designed solution with a specific view for one set of purposes. Inside IBM, portals are used for things such a human resources and IT support. Many options (applications) are shown on one browser window, yet they all pertain to the same area of interest: human resources or IT support.

Finally, a rich-client interface creates a conversation back and forth with the existing application. Although it is possible to implement a simplified rich client with little change to the application code on the server, this is typically only a temporary solution. Most rich-client solutions require changes to the existing code on the System i platform. Rich clients usually have UI code as well as business logic, both of which reside on the desktop. This requires some of the code to be moved from the server to the client; therefore, the underlying application must be modified.

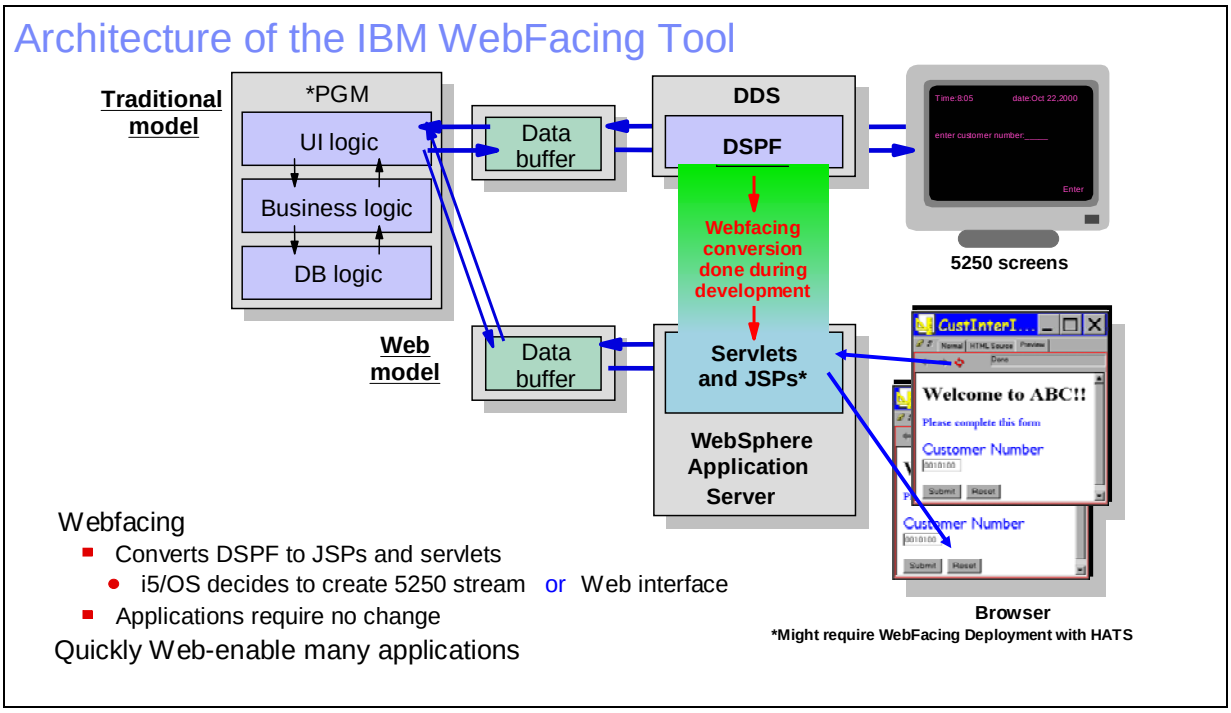


IBM WebFacing Tool

This screen capture of the IBM WebFacing Tool development environment is provided in WebSphere Development Studio Client. The entire collection of views and editors is called the *Webfacing* perspective. Developers can open multiple perspectives and flip between them using the icons in the bar on the left. This looks familiar to all the other IBM development tools discussed in this course.

Developers use a wizard to create a project. They specify the DDS and the user-interface member (UIM) Help Panel members to convert. The resulting project allows developers to have editing access to the original DDS and to the generated files. It is easy to test the converted output in the IBM WebFacing Tool for i5/OS by right-clicking the project and selecting **Run on Server**. During development, this option runs the built-in, preconfigured copy of WebSphere Application Server in the test environment on the desktop (as can be seen in the bottom right window on the screen capture). For production, the application must be moved to WebSphere Application Server on the System i model (or, in fact, moved to any WebSphere Application Server implementation).

WebSphere Development Studio Client V6.0.1 requires the IBM WebFacing Deployment with HATS Technology runtime environment in order to run Webfaced applications using any dynamic conversion work. At the next release of the tools, this will become a requirement for all applications that use the Webfacing technology. There is still no requirement for the interactive i5/OS feature, but there will be a run-environment requirement with WebFacing Deployment with HATS Technology.

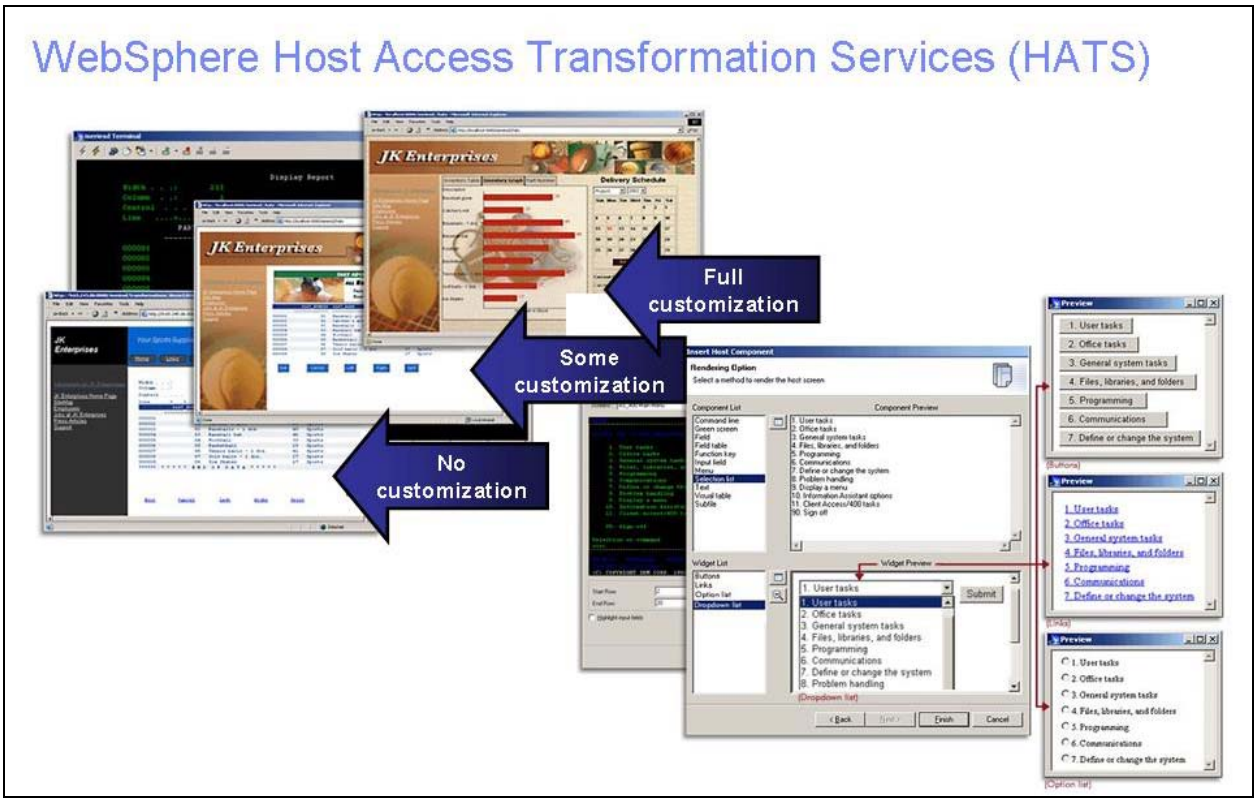


Architecture of the IBM WebFacing Tool

Before explaining the architecture of the IBM WebFacing Tool, it is best to understand the architecture of a traditional green-screen application. (**Note:** To do this, follow this diagram from left to right across the top of the chart.) When a program runs a WRITE operation, it passes the contents of its I/O buffer to the operating system, which merges the buffer with the DSPF object. This produces a 5250 data stream that is sent to the green-screen terminal.

Using the IBM WebFacing Tool, a developer points to the DSPF source code. The converter reads the source and creates a corresponding JSP for each display file format. These JSPs are placed into an application server. When a request comes from a browser to invoke the program, the application server initiates the request to start the program. The unchanged program runs and when the WRITE operation occurs, the program dumps the contents of the program I/O buffer to the operating system (just as in the previous example). However, the operating system interrogates the status of a switch that designates whether the request was initiated from a browser or from a green screen. If the request came from a browser, the I/O buffer is handed back to the application server and to the servlet. This data is merged with the JSP and HTML output is sent to the browser.

In this way, it is possible for two users to work side by side, both running the same application code: one user can work on a green screen and other can work from a browser.

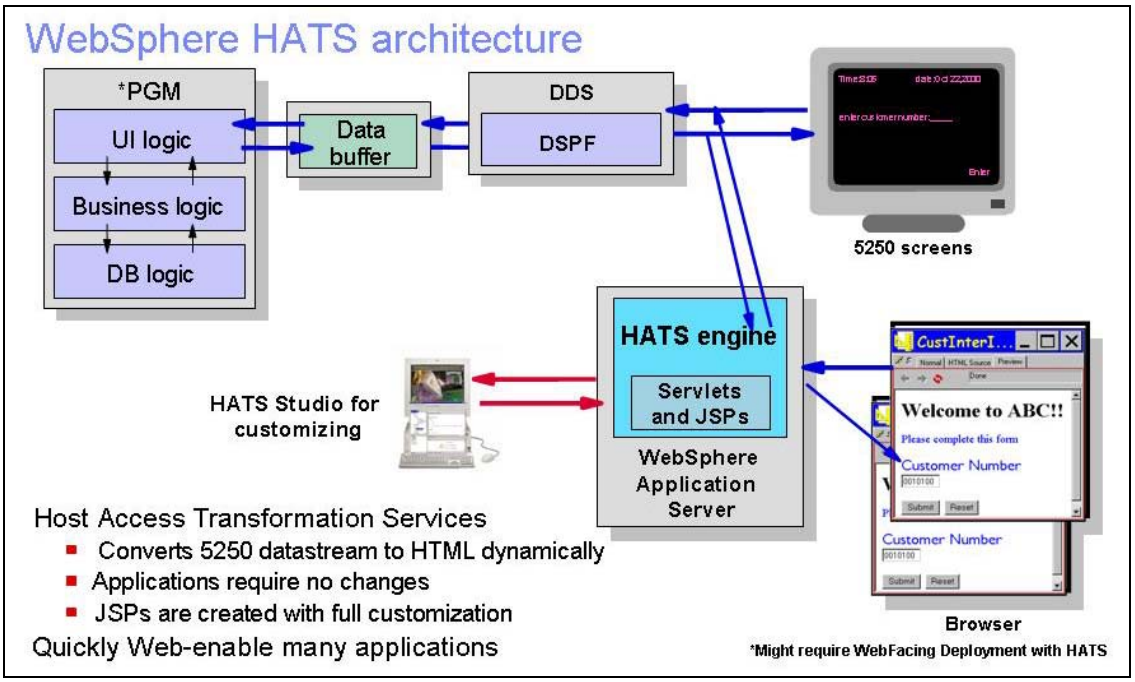


WebSphere Host Access Transformation Services (HATS)

The set of four progressive screen captures shown on the left side of this chart illustrates the results obtained from doing various levels of customization by using HATS. The original green screen is in the background, with the levels of customization increasing from the bottom left picture to the top right.

By looking at the diversity among these screens, you can see that HATS provides a great deal of built-in customization capability, including the ability to turn subfiles into graphs and to insert calendar date-pickers.

The three screen captures on the right show the Eclipse development environment for HATS. This environment is used for the customization of the generated UIs. Alterations require no HTML skills and are accomplished with easy-to-use wizards and dialogs. As with the IBM WebFacing Tool for i5/OS, HATS leverages the built-in test environment for WebSphere to simplify the process of displaying the results.



WebSphere HATS architecture

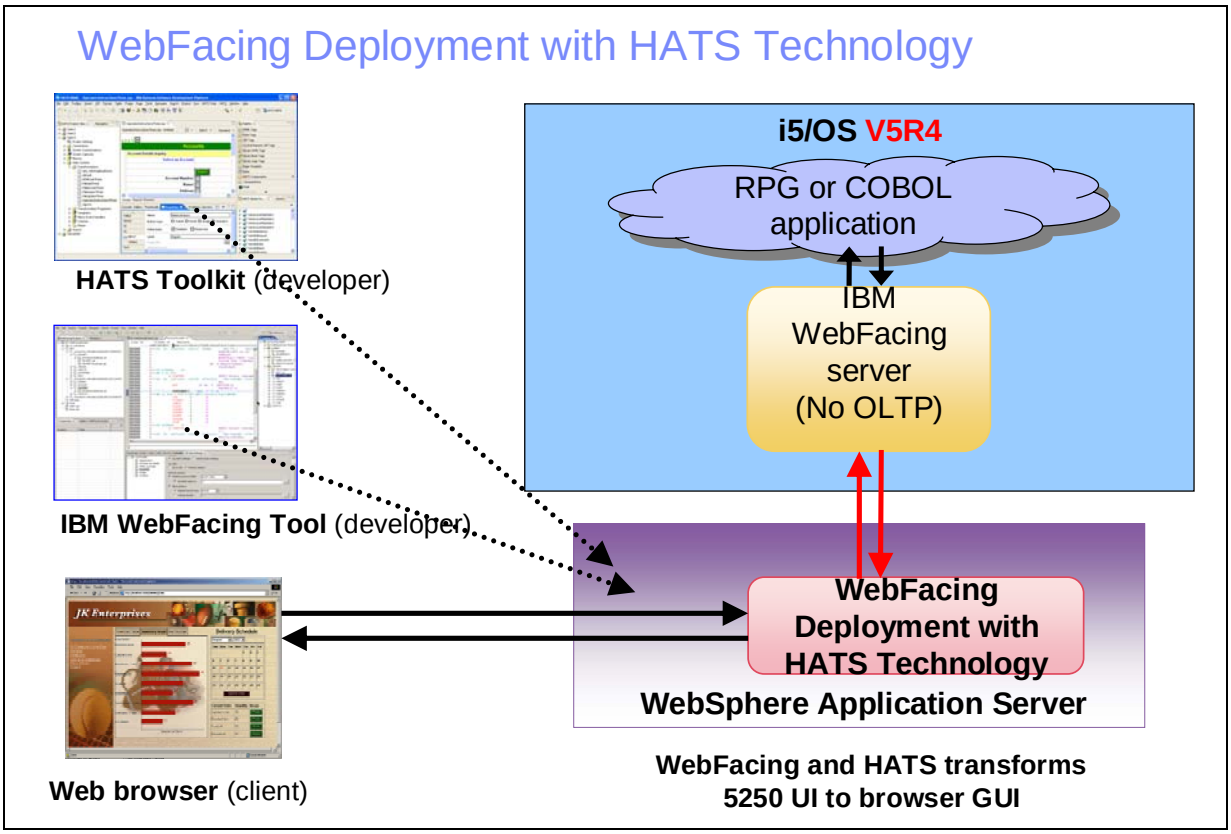
Before reviewing the HATS architecture, it is best to see the architecture of a green-screen application. (**Note:** Follow this diagram from left to right.) As a program runs a WRITE operation, it passes its I/O buffer contents to the operating system, which merges the buffer with the DSPF object to create a 5250 data stream that is sent to the green-screen terminal.

A developer uses WebSphere HATS to define a few rules for transformation — including rules for handling menus and function keys. When a program runs a WRITE operation, it passes the contents of its I/O buffer to the operating system, which merges the buffer with the DSPF object. This produces a 5250 data stream. The HATS runtime engine then intercepts the data stream and transforms it (according to the defined rules) to produce HTML that is sent to the browser.

For more detailed customization, a developer can use the HATS Studio tool to review the specific screen images and can create notebook tabs and redefine the flow of the application. These detailed customizations are stored as individual JSPs in an application server. When the intercept routine grabs the 5250 data stream, it first looks for a corresponding JSP that might have detailed customization. If there is no specific JSP for the screen image, the HATS runtime engine applies the default rules. (See the previous chart for examples of customization.)

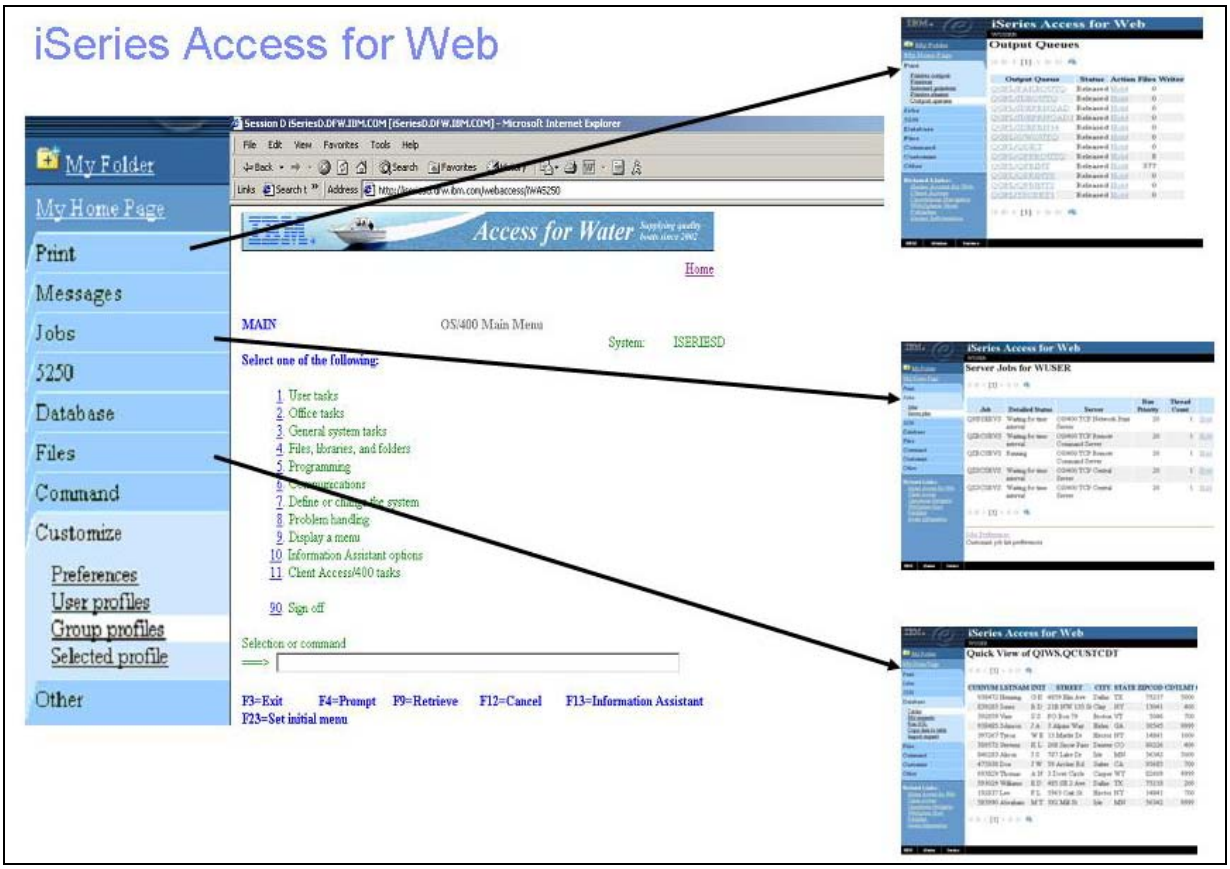
These JSPs are placed in an application server. When a browser makes a request to invoke the program, the application server initiates that request. The unchanged program runs and when the WRITE operation occurs, the program dumps the contents of the program I/O buffer to the operating system (just as in the IBM WebFacing Tool example). However, the operating system interrogates the status of a switch to determine if the request initiated from a browser or a green screen. If the request came from a browser, the I/O buffer is handed back to the application server and the servlet. This data is merged with the JSP and HTML is sent to the browser.

As with the IBM WebFacing Tool, HATS lets two clerks work while running the same application code: one clerk can work on a green screen and other can work from a browser.



WebFacing Deployment with HATS Technology

WebFacing Deployment with HATS Technology is the runtime component of the IBM Refacing solutions. The IBM WebFacing Server on the System i platform talks directly with the unchanged application program. The WebFacing Deployment with HATS Technology environment runs the necessary mapping or transformation to convert the 5250 images to a browser-based rendering.



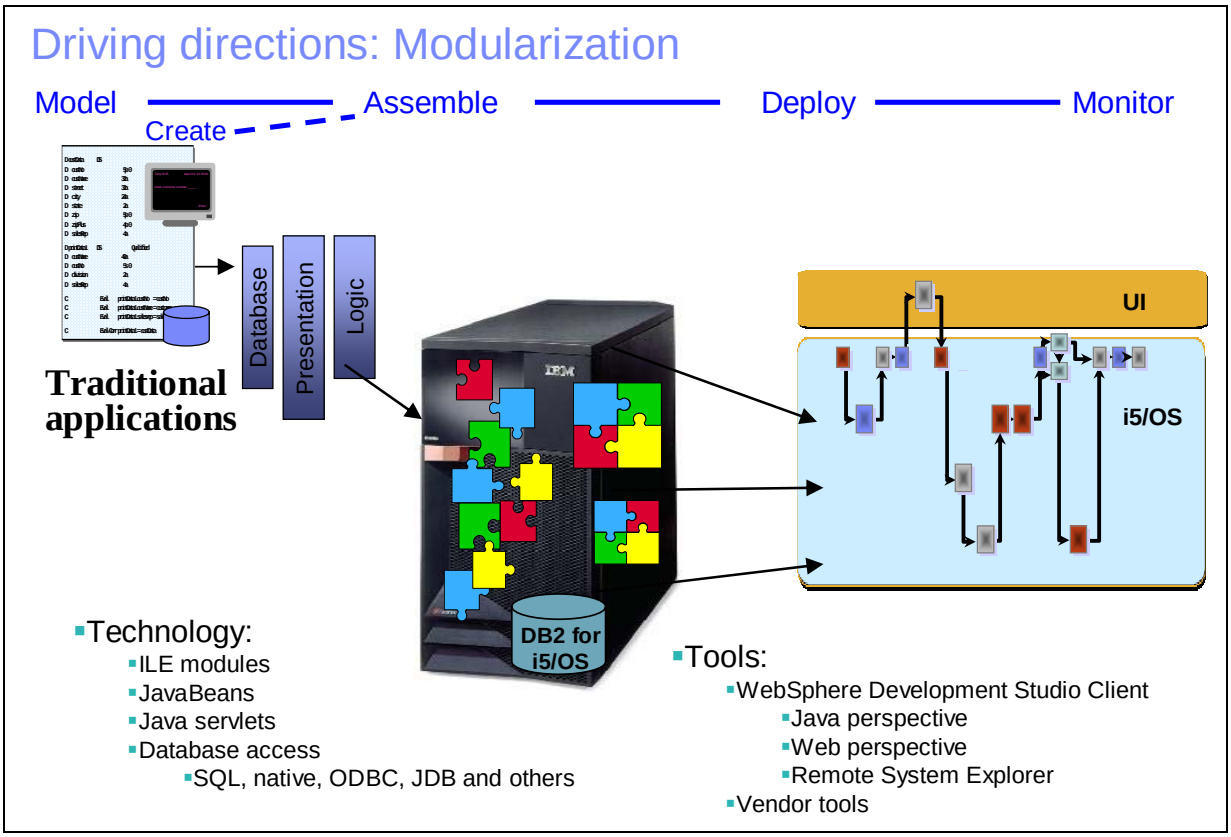
iSeries Access for Web

iSeries Access for Web transforms the 5250 data stream into HTML at run time.

The folder of available iSeries Access for Web functions is visible down the left side of this screen capture. The authorized options for an individual user are displayed here. There is a lot more function than simply performing 5250 data-stream-to-browser conversion. Other key functions include:

- Displaying and working with messages
- Displaying output queues and spooled files
- Dynamically creating database inquiries through SQL
- Running i5/OS commands

Although the 5250-to-browser conversion routines require interactive cycles, the other operational functions do not.



Driving directions: Modularization

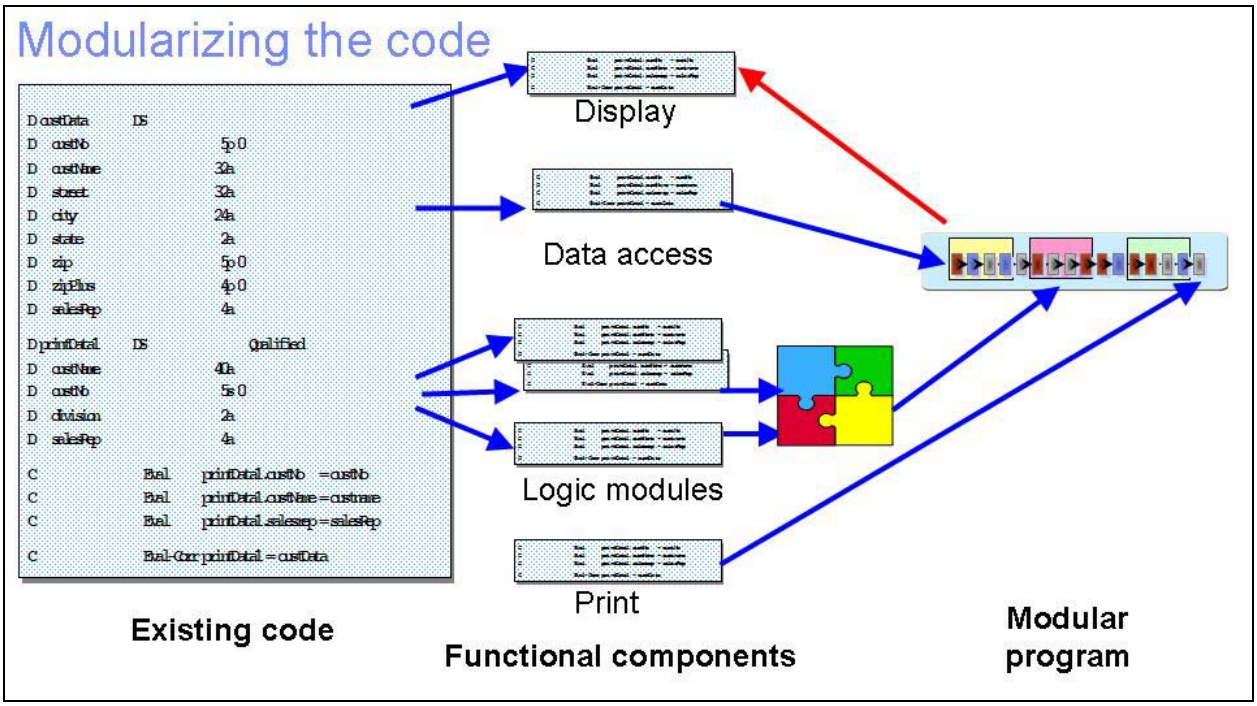
This pathway begins in the same place as the refacing pathways (with traditional applications). However, instead of simply adding a new user interface to the application, this process concentrates on improving the code itself. This approach is more aggressive, requiring the developer to crawl through the program and analyze which code can be isolated by function.

Step one of this phase is to move to the most current version of the compilers: ILE RPG (RPG IV) and ILE COBOL.

Step two involves analysis whose purpose is to separate the UI code, the data access code, the printing code and one or more business logic modules.

The IBM tools for this purpose are the traditional tools for doing code development: a Java editor, Web editor or RSE. A single command converts RPG/400 code to RPG IV (the i5/OS Convert RPG Program [CVTRPGPGM] command). This command provides only basic syntax conversion; it does not reengineer your code.

IBM vendor tools can assist in more extensive conversion and modularization work when migrating applications from RPG/400 to RPG IV.



Modularizing the code

This chart graphically shows the steps for creating modules from larger monolithic programs. The object of the exercise is to reduce the code into smaller functional components; that is, separating the logical functions (display, data access, one or more logic modules and printing) so that they become eligible for inclusion in new service-oriented applications.

Modular architecture

- ▪Reuse application code
- ▪Modularize traditional applications
  - •ILE modules (RPG, COBOL, C, CL)
  - •Separate functions (business logic, user interface, data access, printing)
  - •Remote System Explorer
- •Provide GUIs
  - •Handcrafted
    - •Web and Web-services tools
    - •JSF creation
    - •Java and Web tools for fat-client development
- •Run database functions with SQL
  - •Move database function to the database (referential integrity, triggers)
  - •SQL for data access
- •Support coexistence with Java where appropriate (for example, GUI code)

Modular architecture

By redesigning the application into a modular one, you allow for the well-designed replacement or addition of modern technologies (such as browser interfaces and distributed database activity). It might also become apparent that some business logic modules are better served by being exposed to trading partners as Web services; this is not feasible with monolithic programs.

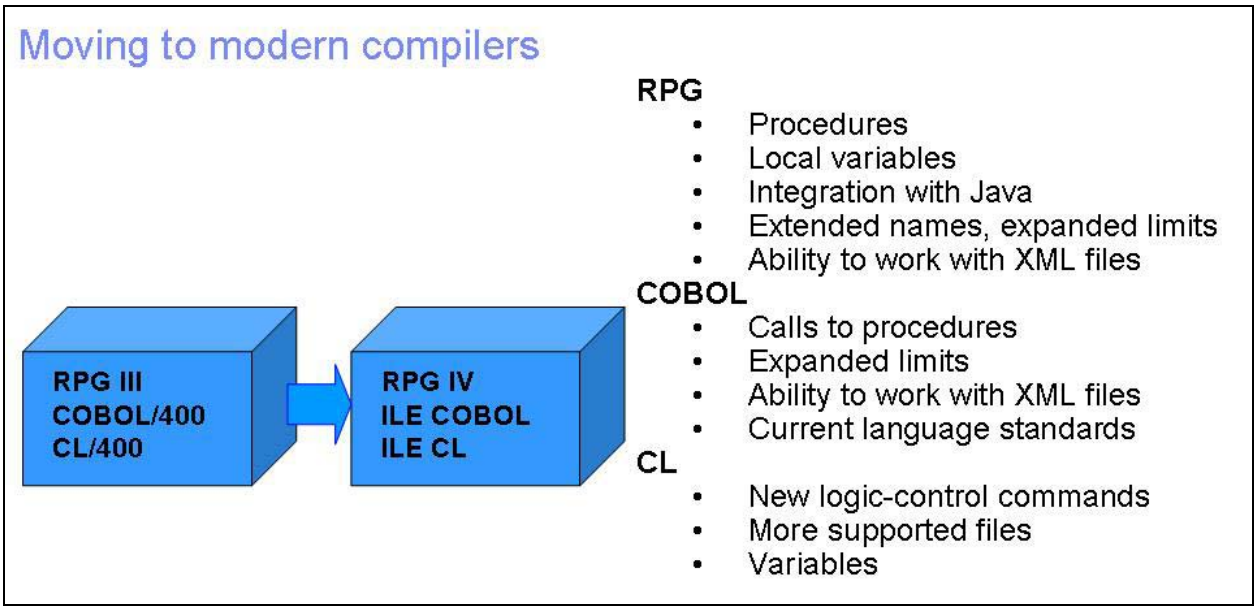
Modularizing the code can still be done with RSE. As you might remember from the first step on the Roadmap, this tool is a component of WebSphere Development Studio Client for i5/OS. Building the new browser-based UI is done with the Web Tool component of the same toolset. The Web Tool component includes the necessary wizards and editors to help develop, expose and publish modules of business logic, thereby creating Web services.

Note: A Web service makes functions (in the form of a remote-function call) available to others to run from another system, without knowing the language in which the service is written or the operating system on which it runs. Web services can be accessed within a company to reuse important functions or shared between companies for business-to-business (B2B) purposes.

The System i Web tools also contain a Web Interaction wizard to generate input and output Web pages for the input and output parameters of a program or procedure. It also generates the Web-application connectors that are needed for binding functions. The Web pages are functional and ready for Web designers to improve visually, if preferred.

To help System i programmers move from SDA to Page Designer for building Web pages, the System i Web tools also come with a number of Web controls, known as *visual custom tags*, that leverage SDA skills. For example, the label and entry-field controls provide support for edit code and edit words; entry fields allow support for validity checking and database references. The necessary HTML and JavaScript is generated for you.

Another key feature of this new modular application is its ability to incorporate modules that are written in different languages, such as Java, COBOL and RPG. As the application requirements are identified, it might become obvious that some key pieces of the application need to be written in the language that is most appropriate to do the task. In the future, applications will no longer be written in a single language; the language will be chosen based on the task's requirements.



Moving to modern compilers

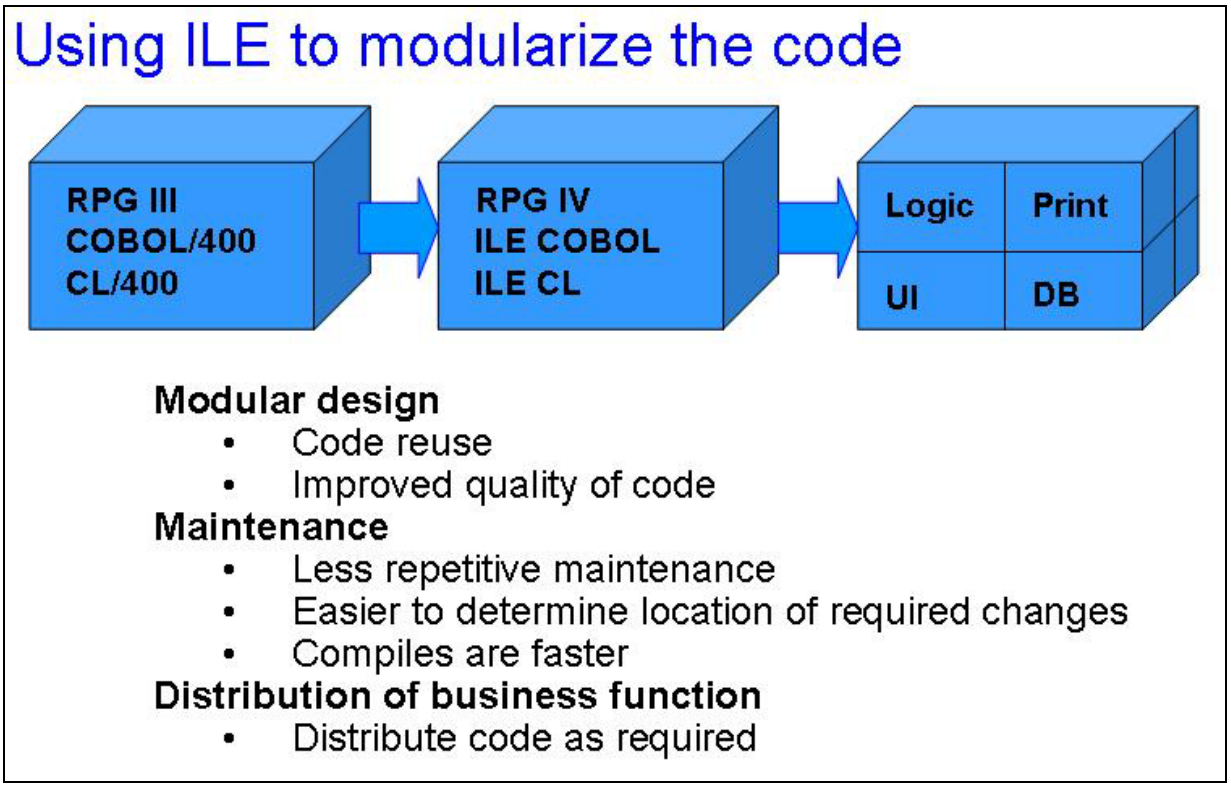
The move to modularization first requires a move to the most-current compilers. The amount of work is significantly outweighed by the advantages that are inherent in the new technologies. However, as with every piece of the Road Atlas, even the move to current compilers must be driven by business requirements.

Some of the more-significant technical enhancements to be gained by using the modern compilers are listed on this chart. It is not the intent of this course to teach language constructs, but here are a few highlights.

RPG IV provides greater support for procedures. It allows local variables and integration with Java code. It permits extended file names and other expanded limits. It also lets you work with XML files, which is particularly important when interacting in B2B environments.

ILE COBOL allows calls to procedures and also supports expanded limits and XML. Additionally, ILE COBOL supports the current industry standards for COBOL.

ILE CL provides new logic-control commands and supports more files and variables.

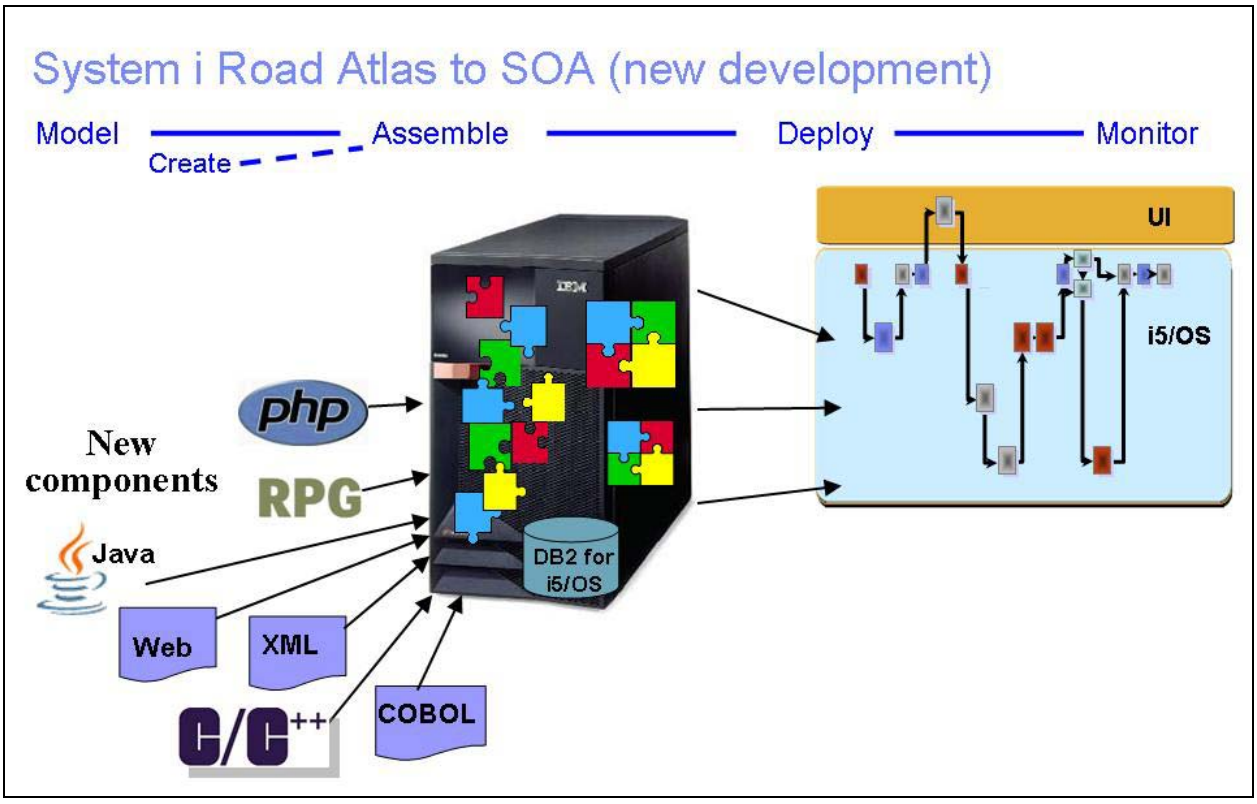


Using ILE to modularize the code

This chart shows a high-level view of how you can modularize RPG and COBOL applications. After the code is in a modern compiler environment, it is possible to separate the code into smaller functional components that can then be exposed for reuse.

This is also the time that database functions can be moved in the database itself where these rules can be enforced without relying on application programmers to code them into every application.

RSE can help a programmer move through the code to look for potentially reusable components.



System i Road Atlas to SOA (new development)

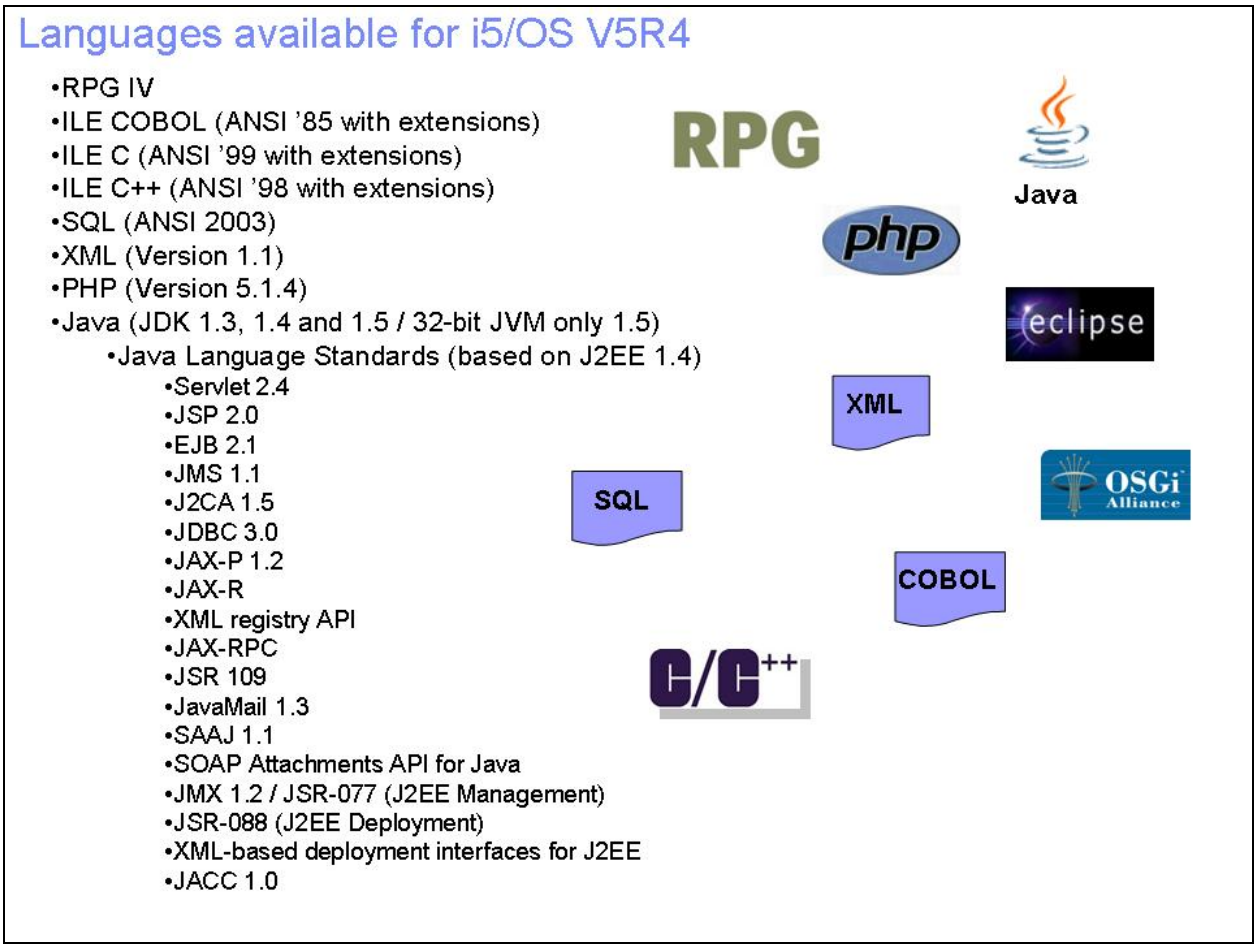
Road Atlas V4 has been expanded to include the development of new components. Many options are available to a System i developer, everything from hypertext preprocessor (PHP), RPG, COBOL, Web, Java, XML and others.

The emphasis is on creating new components that follow the modern paradigms of coding:

- Isolation of function into modules
- Separation of the UI
- A clear definition of the interfaces between functions and interfaces

The languages and tools chosen to create the new components are irrelevant in terms of the resulting application. The picture represents many of the technologies that are currently available for the System i platform; any or all can be used to create new modules.

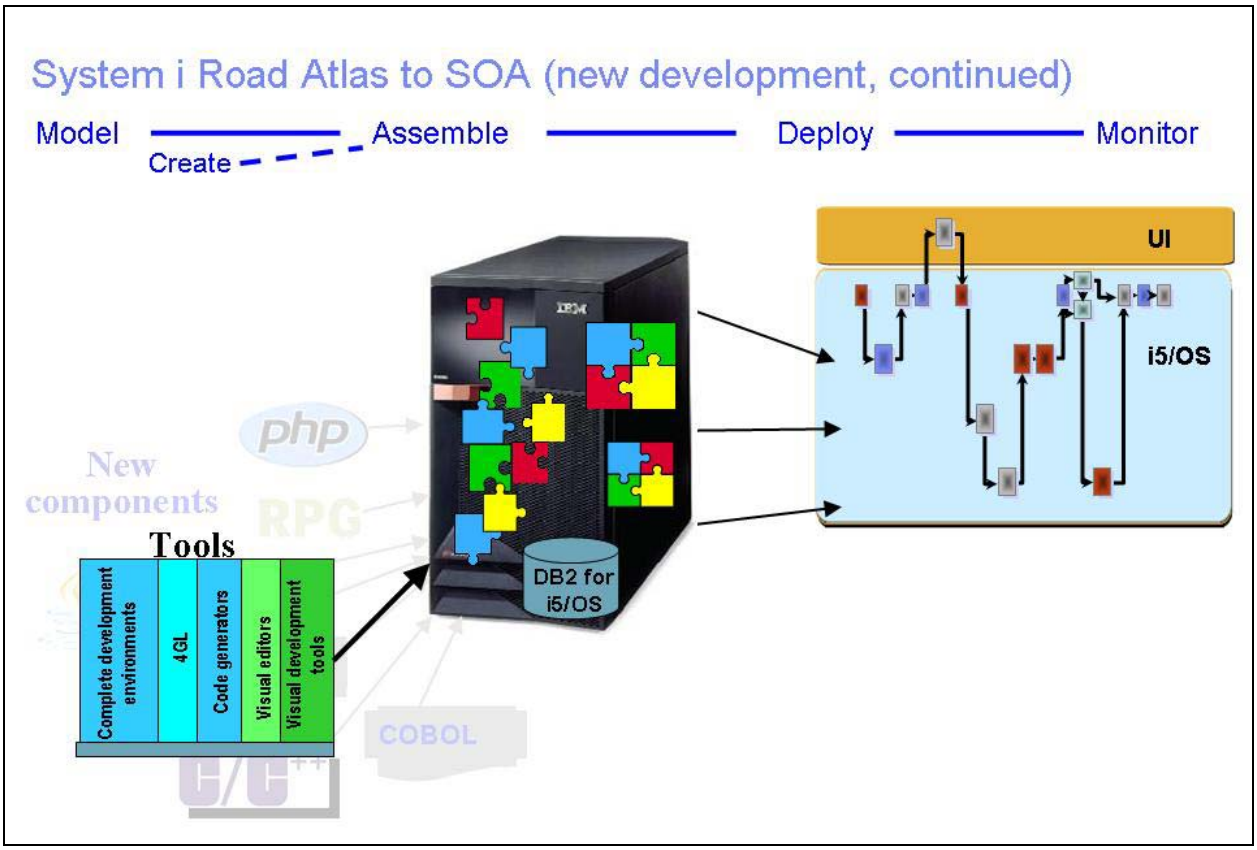
A few modern tools help in the creation of these new modules. Although older development tools can also be used, the later tools provide additional wizards and smart guides to assist developers in their various tasks. Modern tools allow for the creation of modules in new technologies that cannot be coded in the traditional green-screen tools, such as XML and PHP.



Languages available for i5/OS V5R4

As you can see on this chart, i5/OS V5R4 supports at least eight languages for developing new application components. In many instances, the mission of the component can be accomplished in more than one of these languages. Therefore, programmers are free to choose the language in which they are most proficient.

However, some of these languages are inherently better for certain tasks. For example, XML is excellent for providing data that is shared over the Internet, Similarly, Java is becoming the language of choice when building Web browser-based UIs.

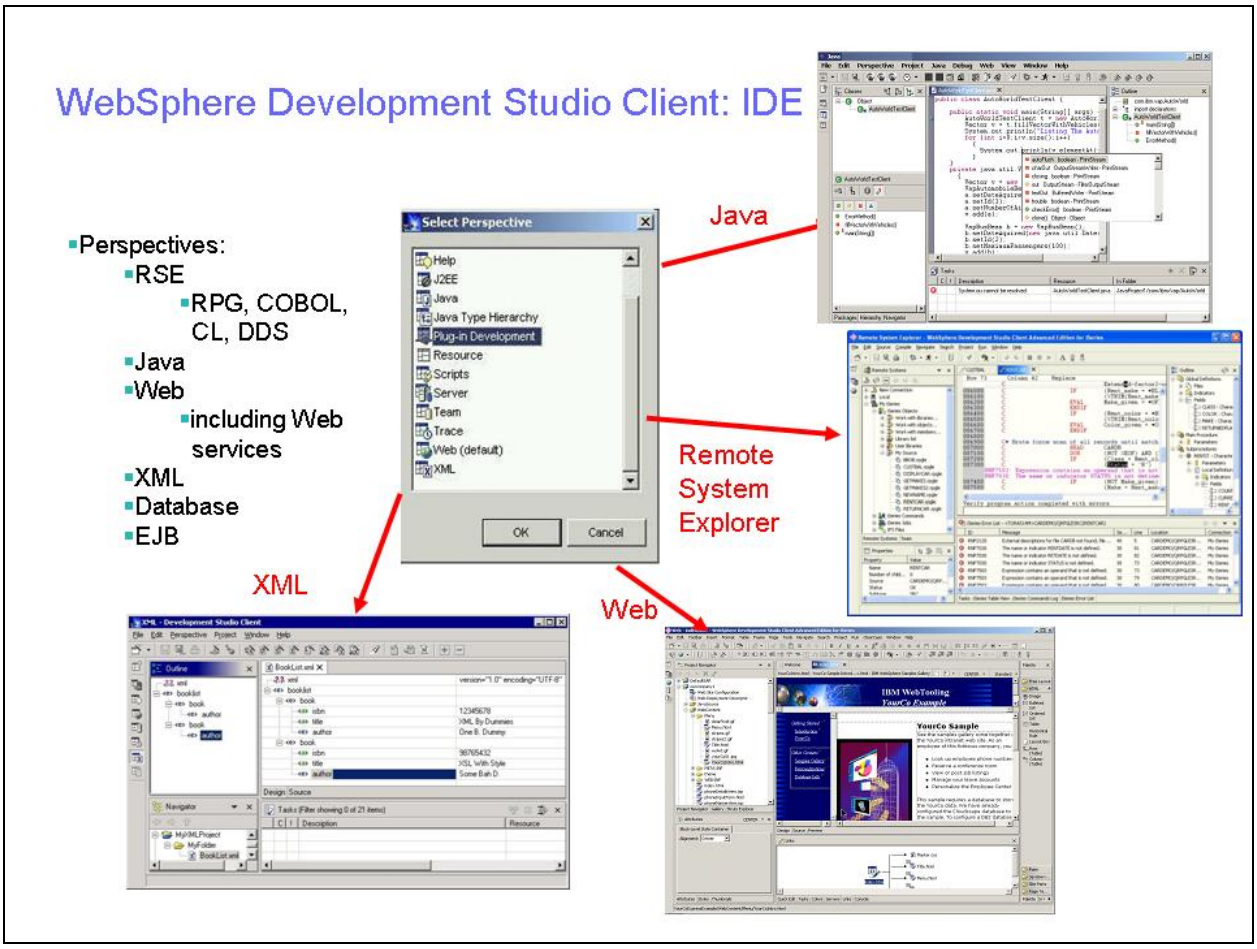


System i Road Atlas to SOA (new development, continued)

The languages and technologies available for development of business components can be very helpful. Now that you understand some of the languages that can be used for the development of new business functions, you need to understand the tools that are available to assist in the development of these new business-function components.

A variety of tools can be used to generate components. The key to any of the state-of-the-art tools is that they are workstation-based; they are graphical and highly integrated.

This next section of this course explores some of these tools in detail.



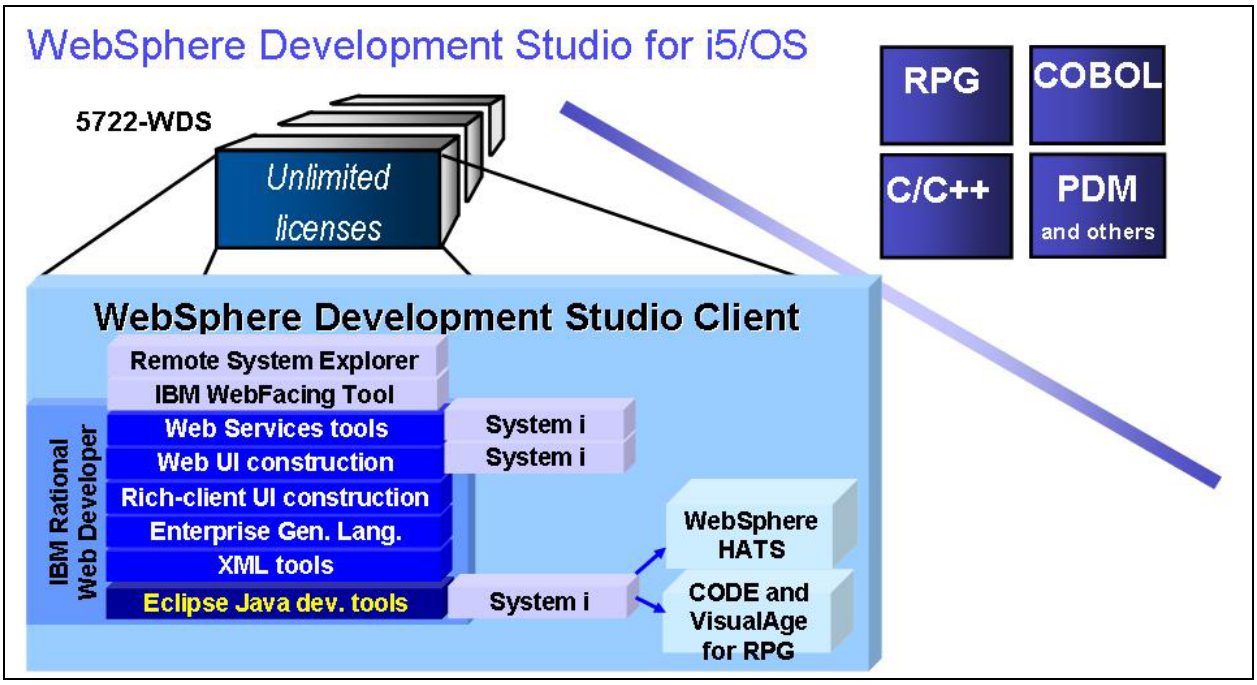
WebSphere Development Studio Client: IDE

The Eclipse-based IBM WebSphere Development Studio Client desktop IDE is the most robust IBM tool for building new components. As you can see from this set of screen captures, it provides multiple perspectives, depending on the language used to create the component. There are Java, Web, XML, RPG, COBOL, Enterprise JavaBeans™ (EJB™) and database perspectives.

Additionally, many tool vendors offer plug-ins to enhance these development environments and provide other perspectives, wizards and smart guides.

After selecting **Perspective** from the menu bar on the main studio page, the development environment opens pop-up window that lists all the available perspectives. From this pop-up window, you can select your preferred perspective.

Every perspective can be managed and customized (for example, window placement, size, and which ones are open at any one time). Changing tasks is as easy as clicking **Perspectives** and selecting an alternative perspective. When multiple perspectives are open, they appear as tabs that allow you to jump between tasks.



WebSphere Development Studio for i5/OS

This is a diagram of the tools in the WebSphere Development Studio Client environment, which has five server components (compilers and tools) and provides entitlement to unlimited licenses of WebSphere Development Studio Client (standard). All these tools are built on top of the Eclipse platform, (which contains a rich set of Java-development tools) and are also built on top of the Rational Web Developer tool suite.

The Eclipse IDE is in the bottom tier of this chart; Rational Web Developer is in the middle tiers; the top boxes show extensions for the System i platform, such as RSE and the IBM WebFacing Tool. Some System i extensions are for use with the Java and Web tools and for the construction of Web UIs. The light-green boxes (on the right) represent the two tool new components in this IDE as of i5/OS V5R4:

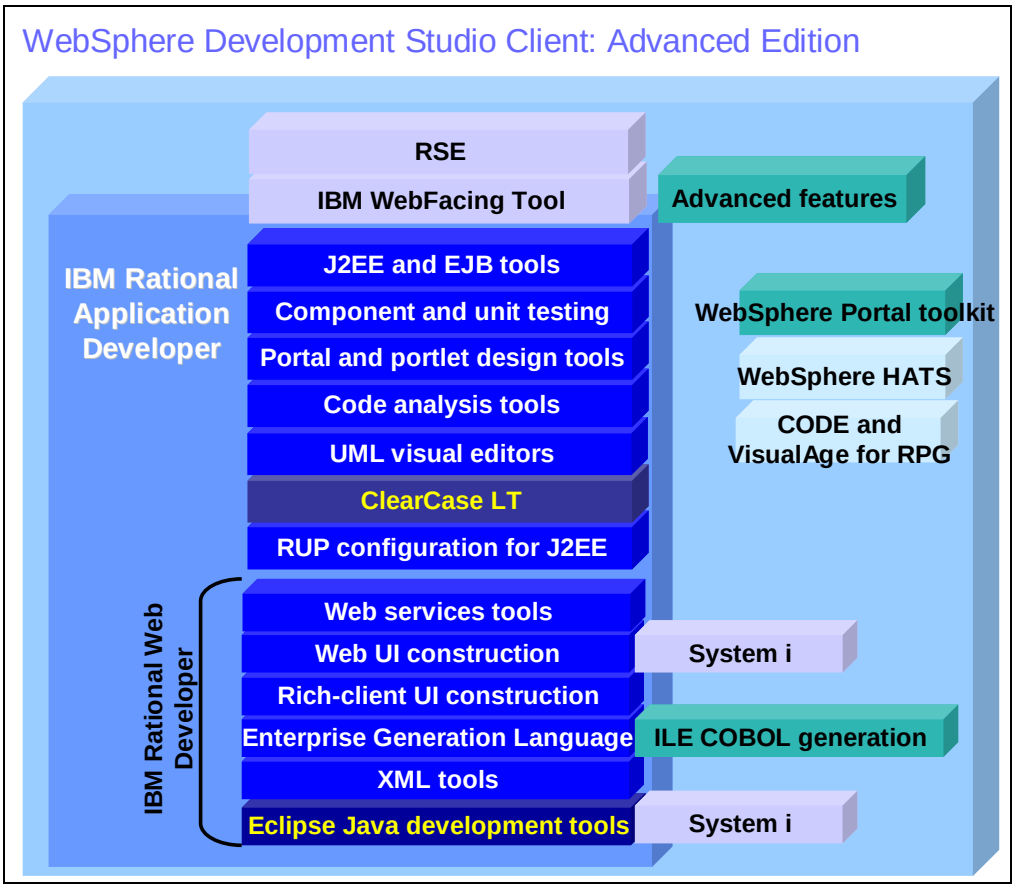
- If refacing is necessary, you can plug WebSphere HATS into the IDE.
- CODE Editor and IBM VisualAge® for RPG are included, but, neither plugs into WebSphere Development Studio Client. From within the IDE, you can manage projects that require the use of these tools. Right-click the component on which you are working, and the IDE prompts you to open this piece of code with CODE Editor or VisualAge for RPG.

WebSphere Development Studio Client is a superset of IBM WebSphere Studio Site Developer because of its unique System i extensions:

- Extensions to the Java tools let you access, export to and run code on the System i platform.
- Extensions to the debugger let you troubleshoot RPG, COBOL, C, C++ and CL code.
- Extensions allow you to create Struts actions with RPG and COBOL business logic and construct Web pages by leveraging your SDA skills.
- Extensions allow you to build Web services from RPG or COBOL business logic.
- The IBM WebFacing Tool for i5/OS interprets 5250 DDS to create Web UIs.

RSE lets you manage, edit, compile and debug RPG, COBOL, DDS, CL, C and C++ code. But after you begin working in Java application environments, you will no longer use your legacy application-coding methods, at least from the perspective of new application development.

There is also support for offline development (not on the System i platform) and team-based projects.



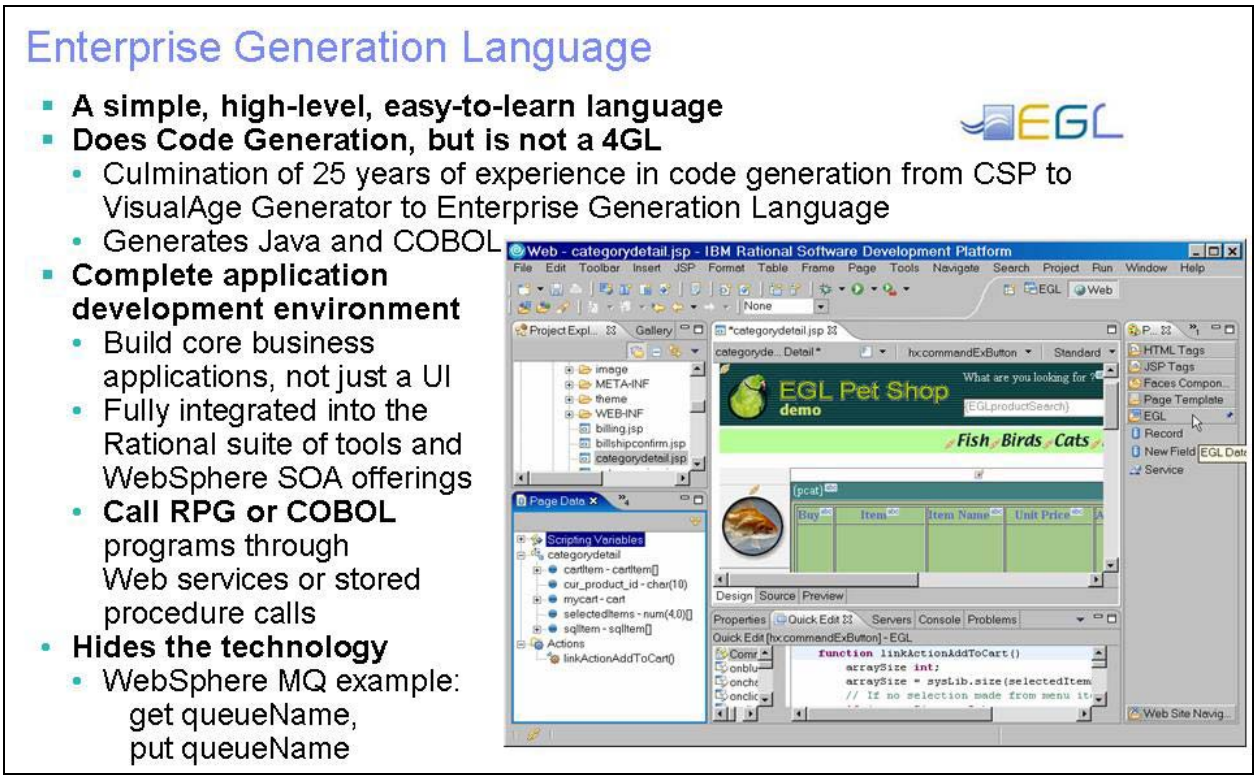
WebSphere Development Studio Client: Advanced Edition

WebSphere Development Studio Client Advanced Edition is sold as a per-seat license and is ordered through the IBM Passport Advantage program.

This enhanced toolset is also built on top of Eclipse and Rational Notice (from the graphic) that Rational Application Developer includes all the features and functions of IBM WebSphere Development Studio Client, as well as many other plug-ins for advanced Java and Web development purposes. WebSphere Development Studio Client Advanced Edition also includes various plug-ins and extensions that customize Rational Application Developer for the System i platform.

The specific, additional features included in WebSphere Development Studio Client Advanced Edition are as follows:

- All features and functions of Rational Application Developer
- IBM Rational ClearCase® Light, which is a team-development product
- Enterprise Generation Language (EGL), which generates COBOL for the System i platform
- IBM WebSphere Portal ToolKit, which is an optional plug-in
- Significant advanced options for the IBM WebFacing Tool (this includes the ability to create portlet-enabled, Webfaced applications and single sign-on for security, and there are also extra enhancements for manipulation of buttons and system commands)



Enterprise Generation Language

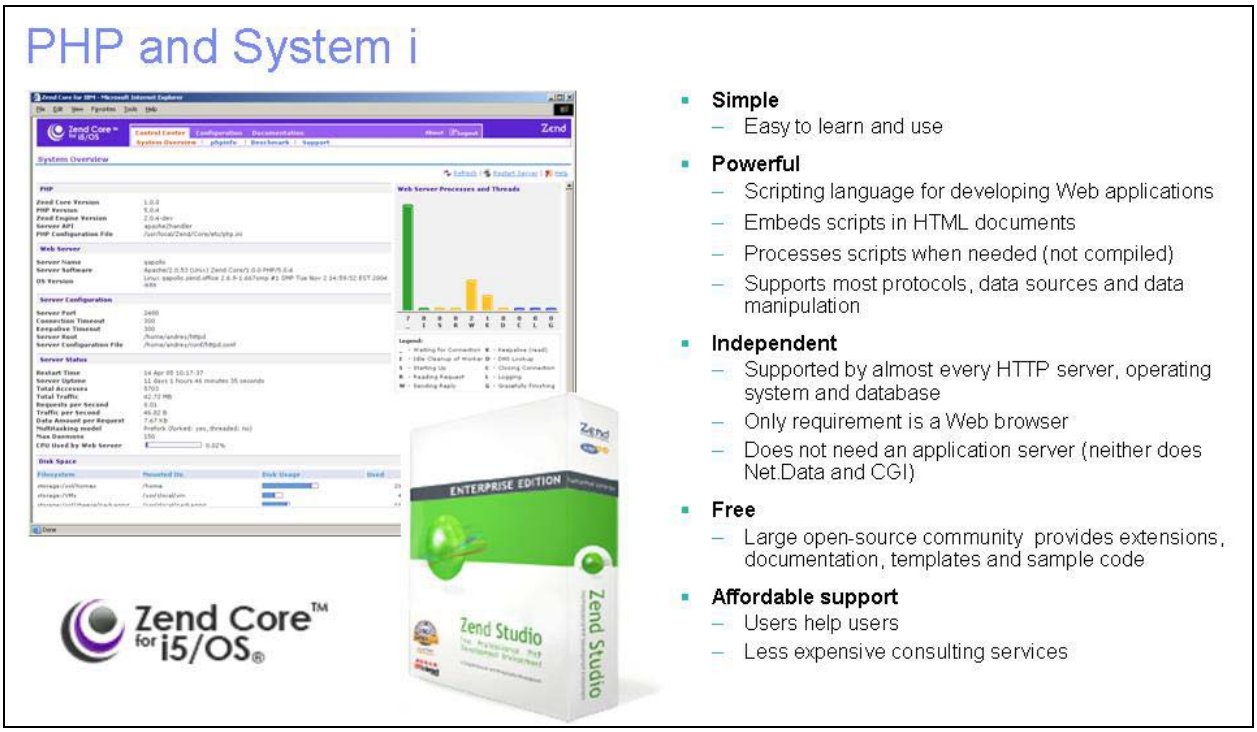
Enterprise Generation Language (EGL) is a modern programming language that allows business-oriented developers to apply a procedural style of programming while exploiting the power of underlying object-oriented technology.

EGL is part of the Rational Software Development platform, which allows developers direct access to a broad range of development services. These services include: modeling, design, construction, test and other development tools. The Rational Software Development platform helps IT organizations to create Web, Web-service, batch and character- and text-based applications quickly and easily.

EGL is shipped as a component of WebSphere Development Studio Client; it is the tool that produces Java code. In WebSphere Development Studio Client Advanced Edition, EGL provides a COBOL-generation capability that requires the COBOL compiler to be installed on the System i model.

Key features of EGL allow you to do the following:

- Develop service-oriented applications without extensive knowledge of SOA.
- Build Java and J2EE applications without a deep knowledge of the underlying technology.
- Deliver applications based on industry standards that interoperate with existing systems.
- Call RPG or COBOL programs on the System i platform with Web services or stored procedure calls.
- Reduce training costs and leverage existing business developers' skill sets.
- Achieve high levels of productivity by leveraging the latest platforms and technologies. (A developer can build and deploy EGL applications to any operating system that is supported on the System i platform.)



PHP and System i

In 2006, IBM announced that the System i platform supports Zend products, allowing PHP to run inside the IBM Portable Applications Solutions Environment (PASE) environment in i5/OS. This was a significant environment enhancement as it allows the many PHP developers to target the System i platform with their PHP applications. PHP has a large following in the open-source community and many components are already available in this environment.

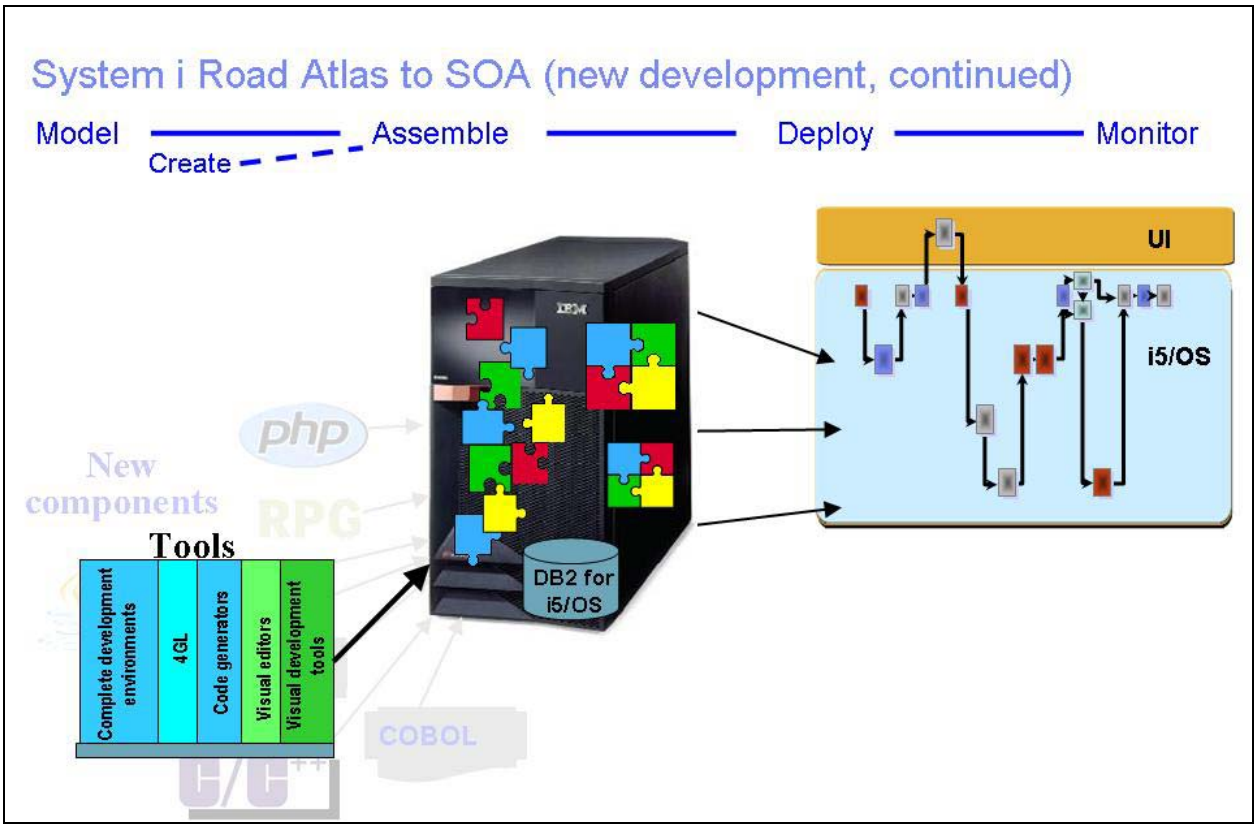
Although the Zend development environment is separate from WebSphere Development Studio Client, there is a goal to create a plug-in to the Rational toolset (and, therefore, to WebSphere Development Studio Client) as soon as possible.

PHP makes it possible for applications to gain the benefits of a Web interface without the need for a Web application server. Instead, a Web browser is the only system requirement. (**Note:** Net.Data and CGI also do not require a Web application server.)

PHP applications are gaining in popularity. Therefore, for System i IT shops and solution providers that want to develop or implement PHP applications, there is an enormous convenience in being able to do so on the platform they already have and understand.

A large community of PHP users have developed high-quality products, tools, code samples, applications and documentation. It can also be easier for IT shops to recruit new, skilled programmers with the availability of PHP on the System i platform.

Because of PHP's popularity as an open-source development and runtime environment, support can potentially be quite affordable. It is common for users to help other users; consulting services can be less expensive, too.



System i Road Atlas to SOA: Locating components elsewhere

The final pathway through the Road Atlas involves the creation, migration and use of application components from other sources. This is the key area of focus for environments that run service-oriented applications.

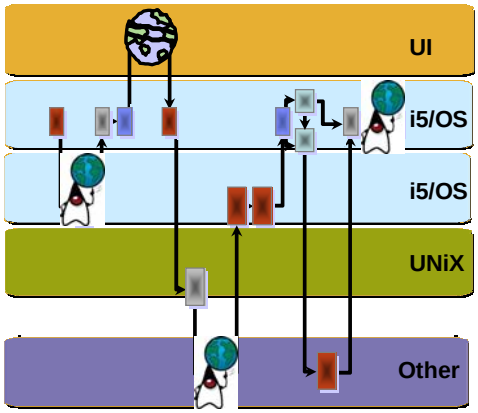
Applications are becoming an assembly of components; business-function components can run anywhere and can be written in any language. Therefore, the idea of locating components into (or from) other sources makes perfect sense, especially when components can be found from many sources.

For example, you can download components to i5/OS. HTML and Java widgets can also be found at various sites on the Web. This means that developers (including Webfacing and HATS developers) can use the precoded scripts to enhance their applications without the need to build these snippets themselves. Additionally, precoded JavaBeans can be found on the Web to do specific business function. Many of these can be downloaded to run on i5/OS.

Many tools build components to run on i5/OS, including some open-source tools. Developers can use Web-development tools that produce standard UI code (such as HTML, JSPs or JSFs) to create components that can then be loaded to i5/OS.

Locating components

- Components from other sources
  - Already exist
  - On System i or elsewhere
- Locate and move
  - Download to System i
    - i5/OS
    - Partitions (Linux, UNIX, Windows)
  - Technology
    - JavaBeans, Web objects, RPG procedures
- Locate and use
  - Access components
    - On other servers
  - Technology
    - Web services
    - XML
    - Connectors and adapters



Locating components

Business-function components can be found in a variety of places. These components are used to populate the application architecture that is created during in the Model process. The components might exist in any language and run on any platform. It is up to the application architect to decide how best to exploit these functional components.

You can find functional components on the Web or from other applications running on the System i platform (such as IBM Lotus® Domino® applications). You can also find functional components from applications created by other development environments (4GLs) that are running on i5/OS. You can even find functional components that are running on other server hardware.

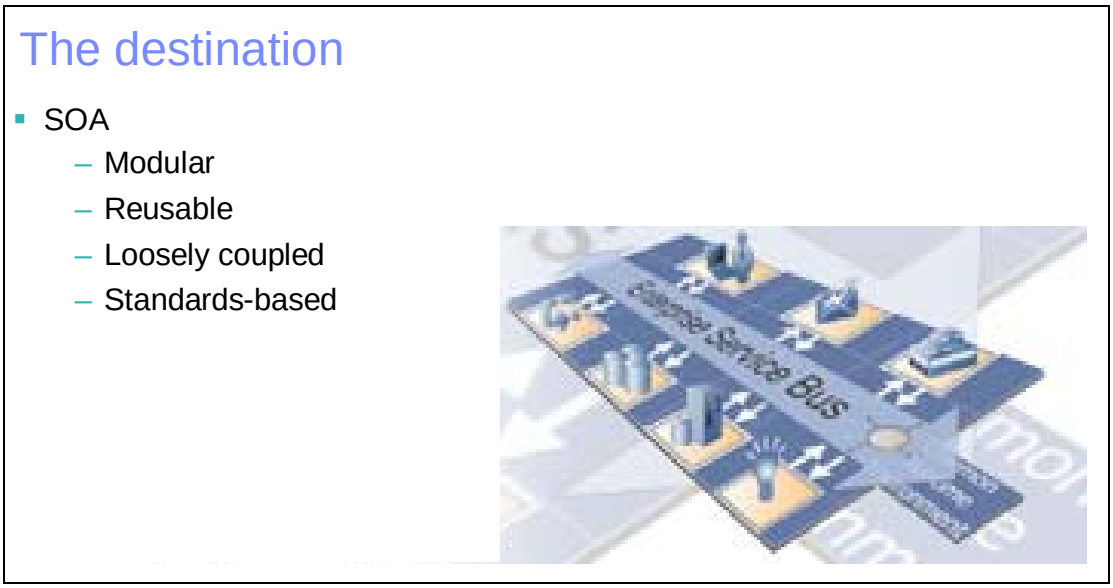
You can use functional components by downloading them to run on i5/OS or you can move the components to a System i partition and run them in that environment. Alternatively, you can simply evoke these components from within your application that is running on i5/OS and let the functional code run in the language and on the platform where it has been found.

After the components are located, depending on the language and the platform, you might need to integrate the components into your application. If the components are written in the same language and run on the same platform, there are more integration options than when you must connect modules that are written in different languages and run on different platforms. In the simplest form, the ILE allows for the binding of functional components together. However, in more-complex environments, connectors and adapters can be required so that the components interact with each other.

This graphic shows the various types of technologies that might be used as connectors.

Lotus

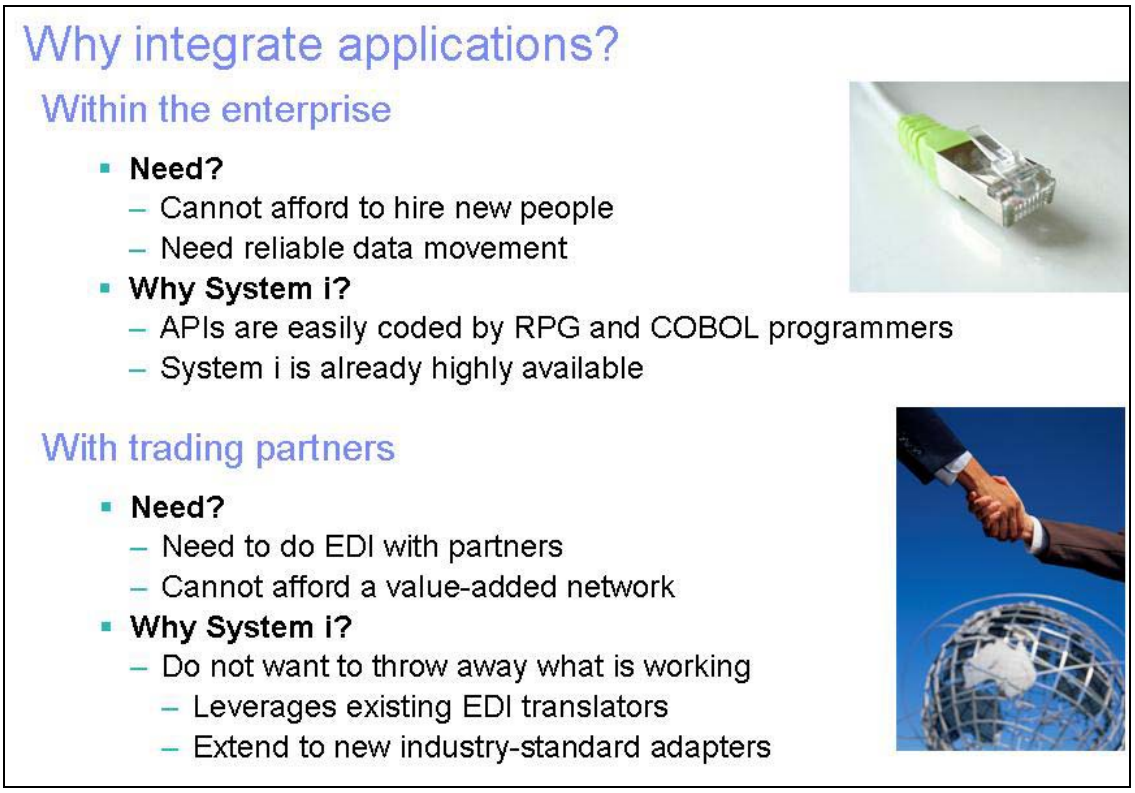
Lotus Domino creates applications that can be used as components when assembling applications in a service-oriented environment. Java is the integration language for evoking a Domino application from a traditional application. Java is also used to call from a Domino application to a traditional application.



The destination

Determining the ultimate destination for your modernized application is an important early decision. The ultimate goal is to achieve an SOA structured application: one that is modular, reusable, loosely coupled and based on proven industry standards. SOA has been embraced by many vendors. SOA is not a whimsical IT strategy; it is one that application-development organizations and enterprises around the world view as the best strategy for going forward with development in today's multiplatform environment.

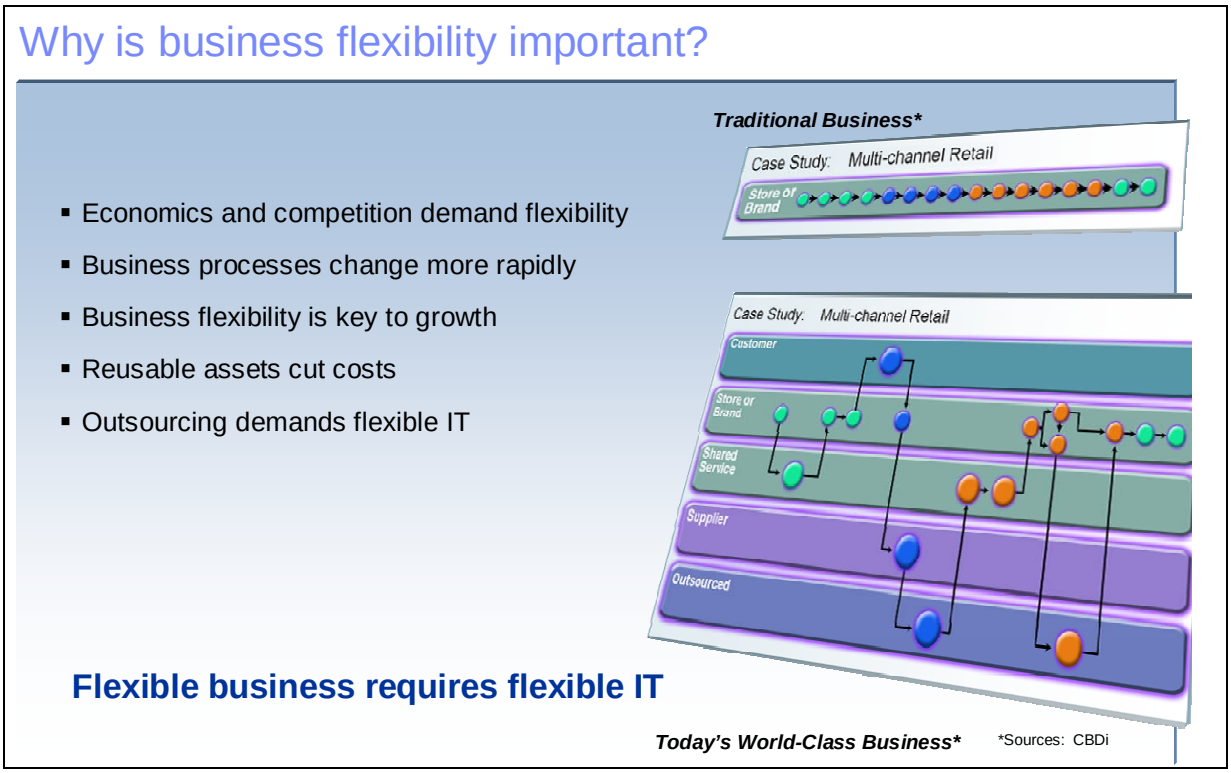
Note: Microsoft has acknowledged the value of implementing a service-oriented environment (SOE).



Why integrate applications?

Within the enterprise: Integration of components can help provide significant value to your business. It allows applications to reuse business functions from your own enterprise and from other businesses. Shared components help to mitigate the risk of assembling new applications. Applications comprised of already in-use components need to indicate that they are well tested. System i is an excellent platform in the world of integration. From traditional applications, APIs are easily coded to evoke other application components.

With partners: Information-sharing between trusted trading partners is not a new concept. The notion of sharing through integration of components is implemented more easily when using the new concept of an SOE. Using Web services, XML transfer or a variety of other connector mechanisms, information can be passed from component to component and application code can be more easily distributed among the various components.



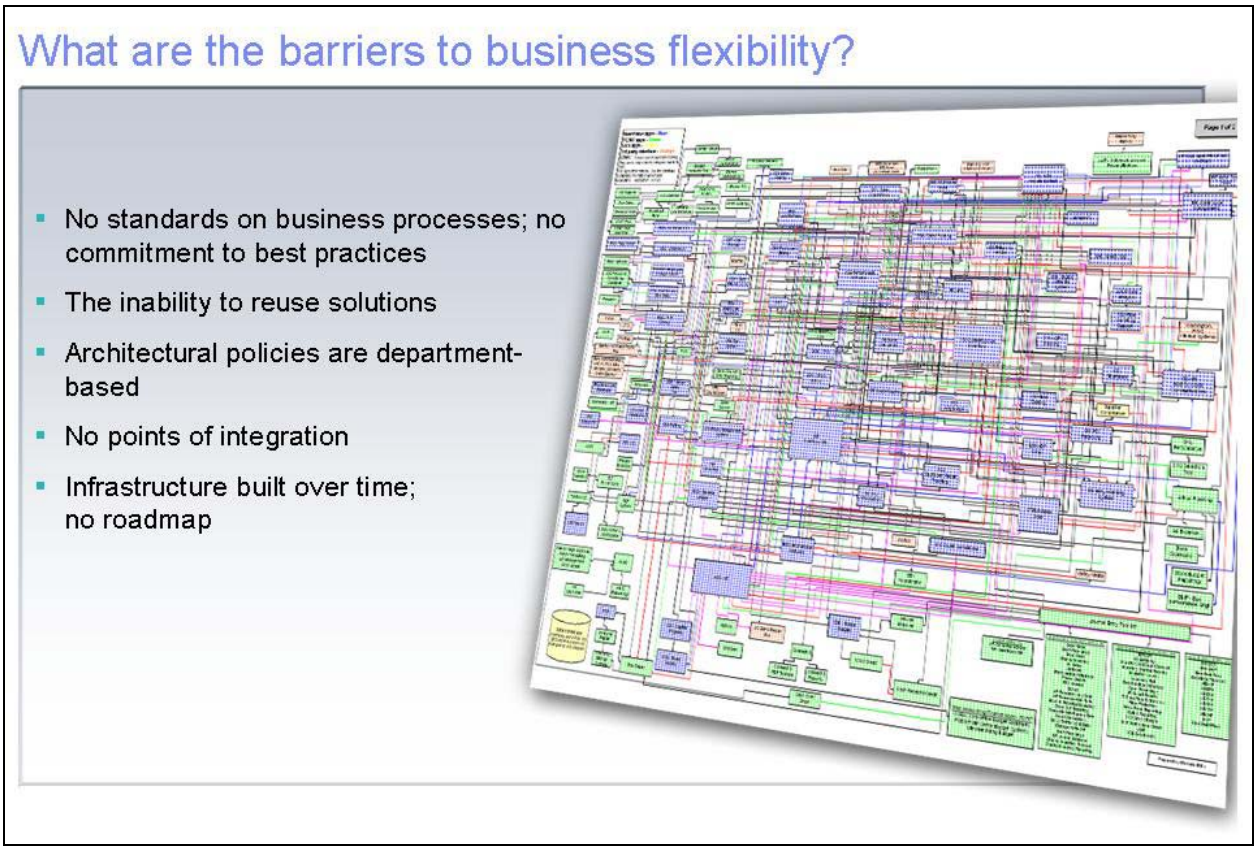
Why is business flexibility important?

Why is SOA so important? There are several reasons:

- **Economics:** Globalization has made the world marketplace more competitive. Therefore, companies need to be more flexible.
- **Rapidly changing environments:** As the world changes ever faster, business processes need to change simply to keep up.
- **Growth:** Flexibility is at the top of most CEO agendas (to keep their companies growing).
- **Cost savings:** Reusable assets can cut costs by up to 20 percent.
- **Outsourcing:** Larger numbers of companies are looking for more effective ways to outsource noncore business services. SOA makes it much easier to interconnect business services across different organizations.

In other words, flexible business requires a flexible IT environment.

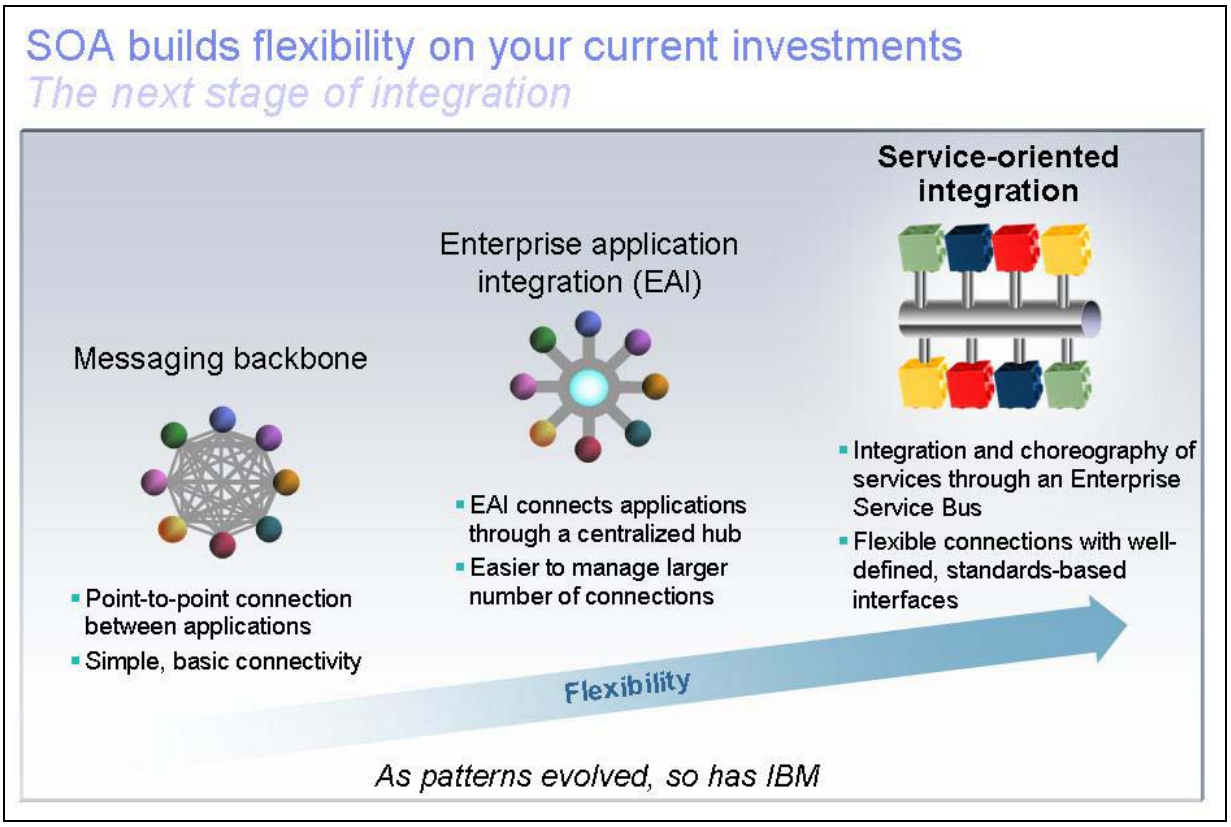
SOA can help you reuse software assets and, therefore, save time and money in IT costs. For instance, the article "The Business Case for Software Reuse" in the *ComputerWorld* February 2, 2004 issue included the following statement from Jeffrey Poulin and Brent Carlson, who wrote, "We estimate that a reasonable savings because of reuse is approximately 80 percent of the cost needed to develop the same software development asset (SDA) for one-time use."



What are the barriers to business flexibility?

In most enterprises today, there are barriers to making the business model flexible and to nurturing the ability to respond to instant requests:

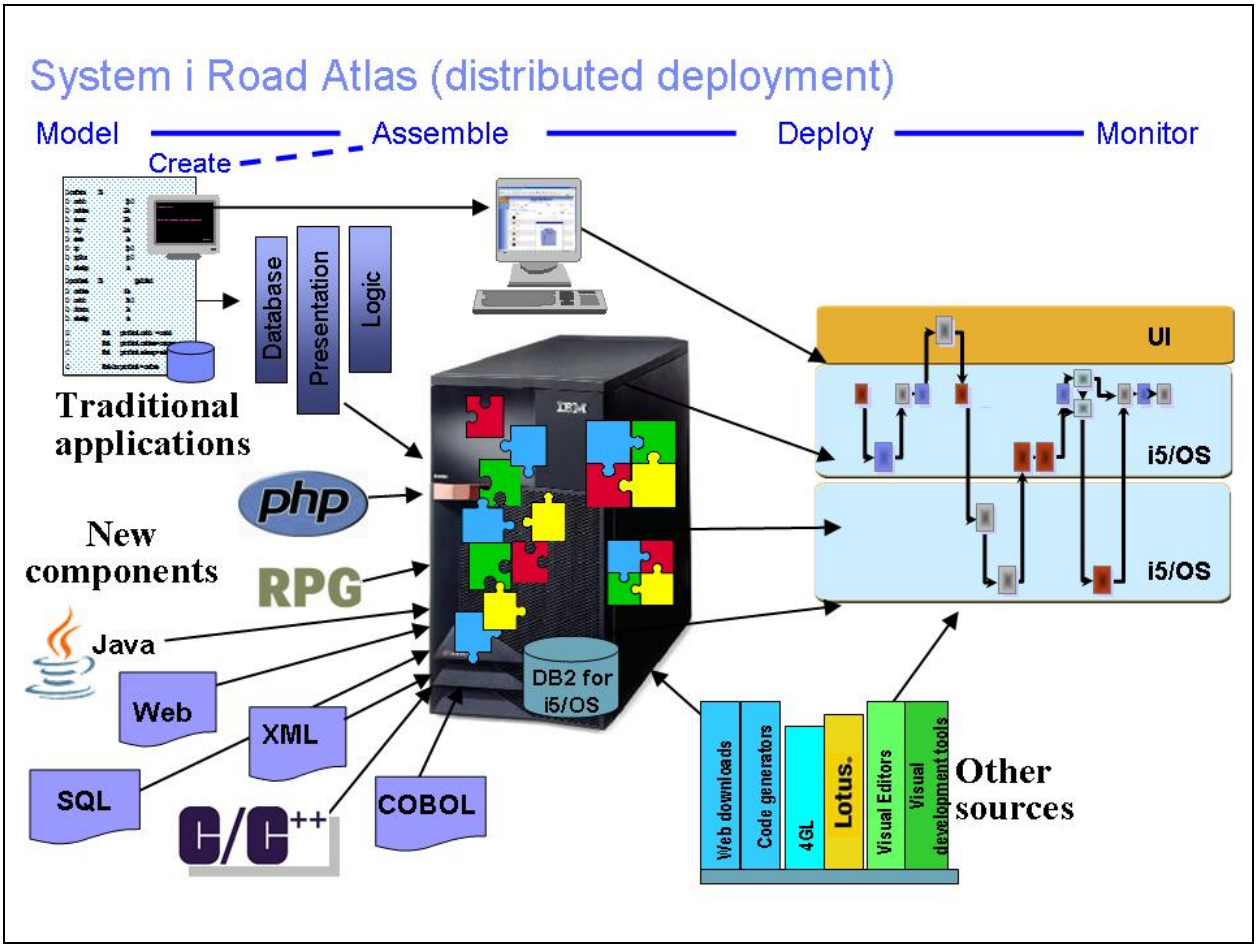
- It seems that there are seldom any process standards within a business environment. Each business unit solves its business requirements in its own unique way, regardless of the other departments. This means that there is a conglomeration of solutions, products, tools and platforms, each solving a unique business requirement. This means that when there is a request to enhance or change a process, it is difficult to make the change happen.
- If every solution is unique and each department solves problems with different tools and platforms, it is almost impossible to reuse existing technology to improve service levels.
- If there is an existing solution architecture, it is often done in isolation from other solution architectures. Therefore, no integration points are built in and no integration levels are considered.
- The infrastructure builds over time. This means that, although one solution might be brought in to solve a business need, this might mean that, within a short time, the solution will grow and grow. Add-on functions are continually developed and often do not integrate well with other solutions.



SOA builds flexibility on your current investments: The next stage of integration

The desire to make IT more flexible is not new. Indeed, it is as old as the IT industry itself. Each of these integration techniques has its place and is appropriate for handling certain situations.

SOA blends the best of all these concepts into one new architecture. But it is important to recognize that SOA is not the end of the road either, it is the next step in the evolution of flexible infrastructures.

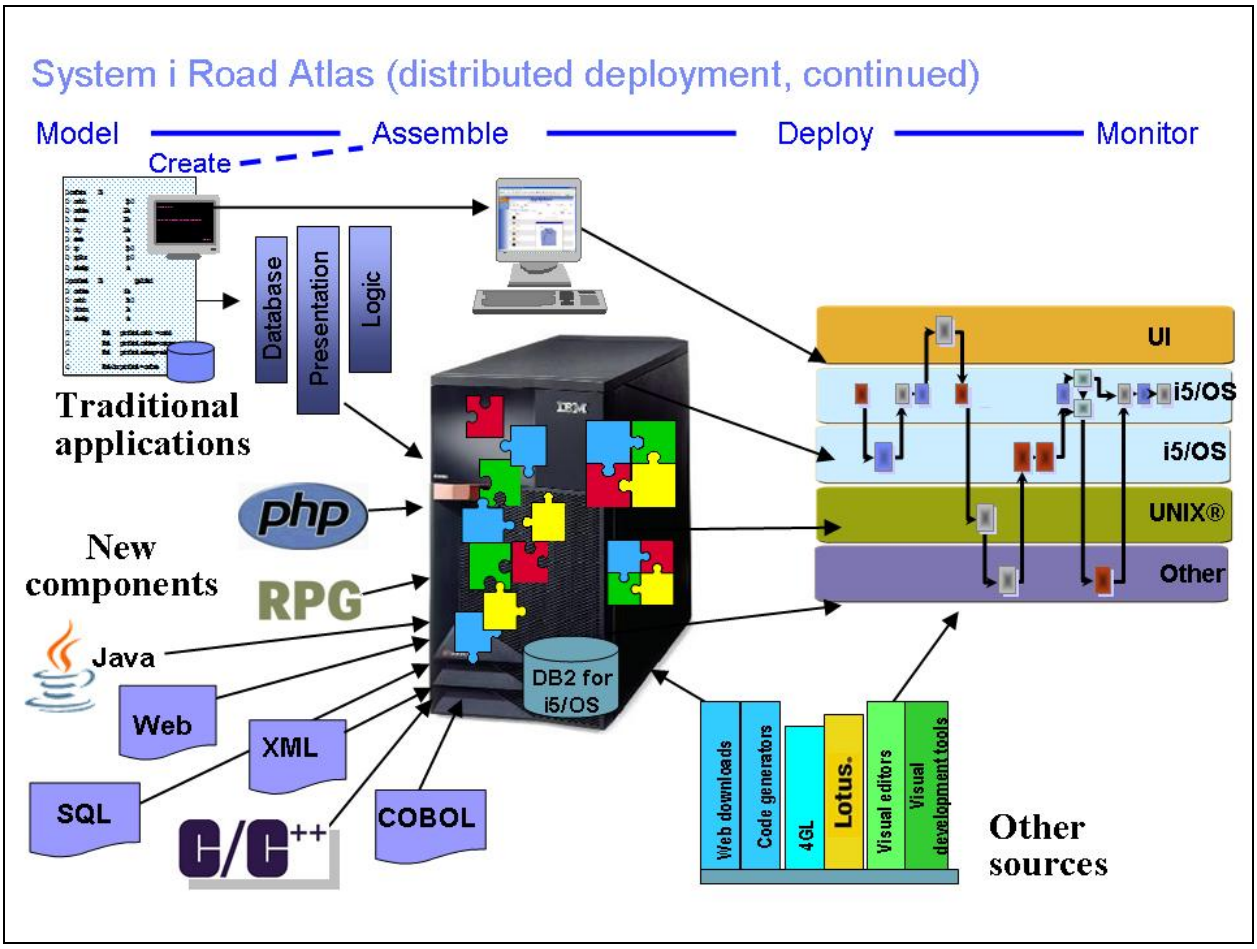


System i Road Atlas (distributed deployment)

When an application is no longer a monolithic collection of code, but has become an architected solution of business processes, then it is possible to distribute the application across multiple partitions.

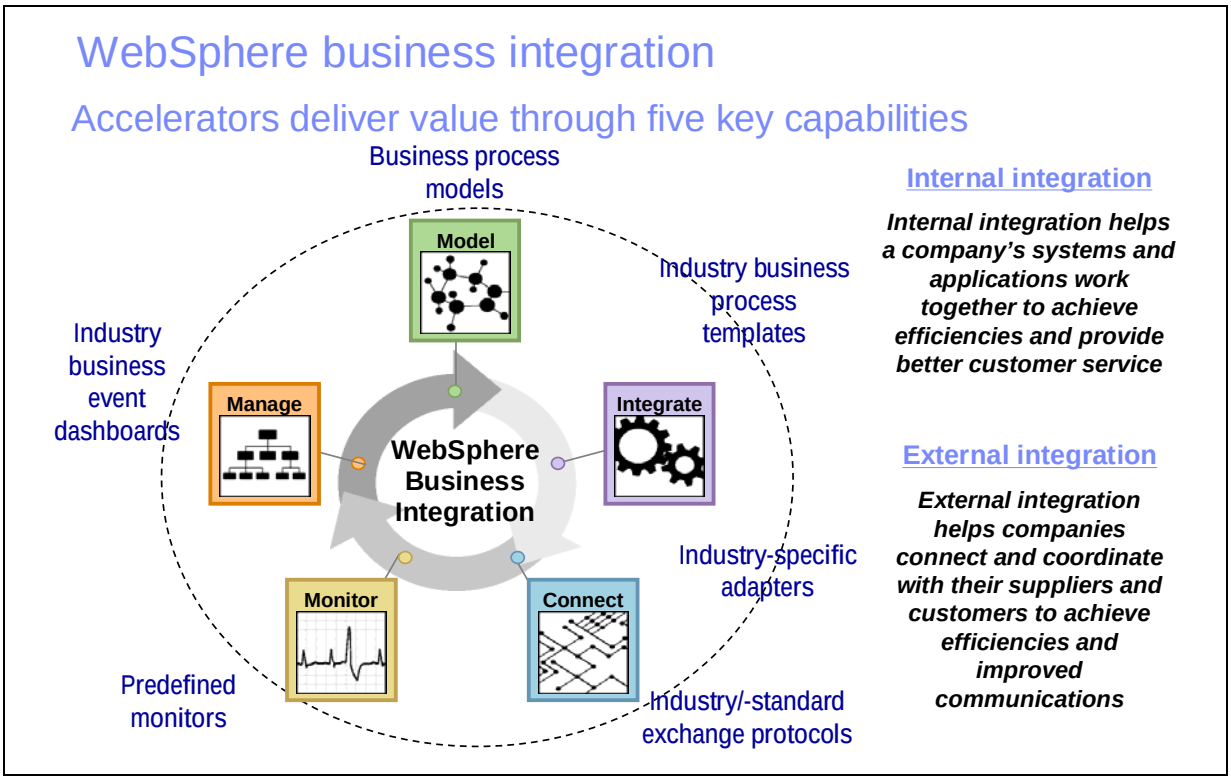
The right side of the diagram shows a modularized application with business functions that are distributed. The UI layer, regardless of how it is implemented, is shown as a separate layer, with two i5/OS partitions. These might be multiple partitions on one box or they can be on different physical systems. The application can be split across multiple locations.

This is a good time to discuss the advantages of an SOA application; it lets you put business function closer to the organization that owns it. As an example of the need to locate function in a distributed manner, consider the allocation of inventory after an order has been entered into an order-entry system. As the order is processed, one of the key components is that inventory must be looked up and allocated. The stocking of inventory occurs and is controlled at individual warehouse locations, not from a central site. In order for the inventory to be allocated, it makes sense to have that component of the application run at the warehouse.



System i Road Atlas (distributed deployment, continued)

Just to reiterate, when an application is no longer a monolithic collection of code, but has become an architected solution of business processes, then it is possible to distribute the application across multiple partitions or platforms. This picture shows the additional platforms where components of the application might run.



WebSphere business integration

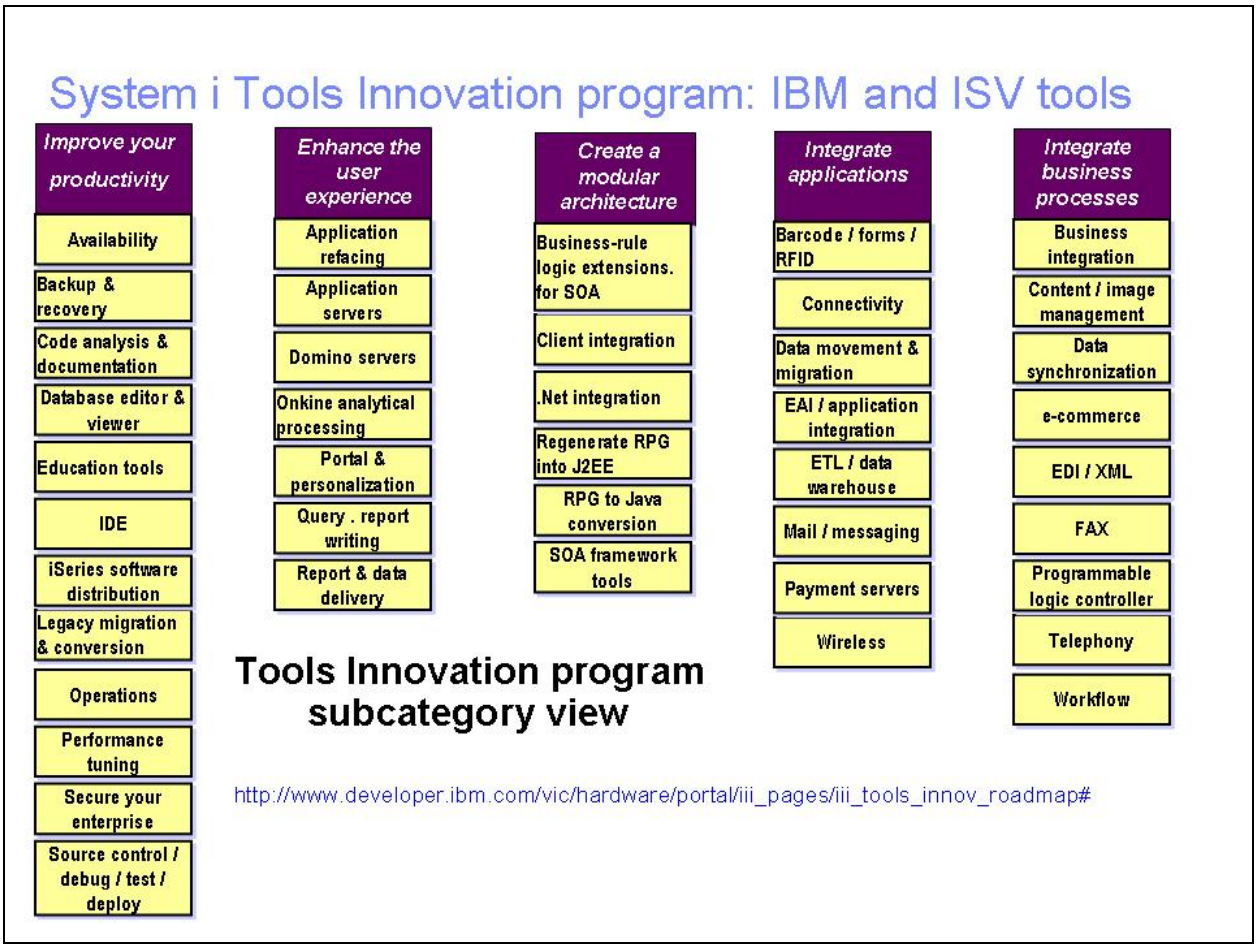
This chart illustrates a summary of the various business-function components that can be integrated. WebSphere Business Integration products provide a simple and easy way to assemble and monitor the distribution of business function.

Business process management is important for many reasons. It helps solve the need for enterprises to keep up with their rapidly changing environment. It also supports the enforcement of standards in coding processes, deployment methods and maintenance tasks.

The WebSphere Business Integration products listed (listed below) help enterprises manage their business processes.

- IBM WebSphere Business Server Express Plus
- IBM WebSphere Host Access Transformation Services Studio with WebFacing Deployment with HATS Technology
- The IBM SOA Foundation:
    - IBM WebSphere Business Modeler
    - IBM WebSphere Integration Developer
    - IBM WebSphere Enterprise Server Bus
    - IBM WebSphere Process Server
    - IBM WebSphere Business Monitor

And the System i platform delivers all these tools to help you start with the right foundation and deployment vehicle for business-process management.

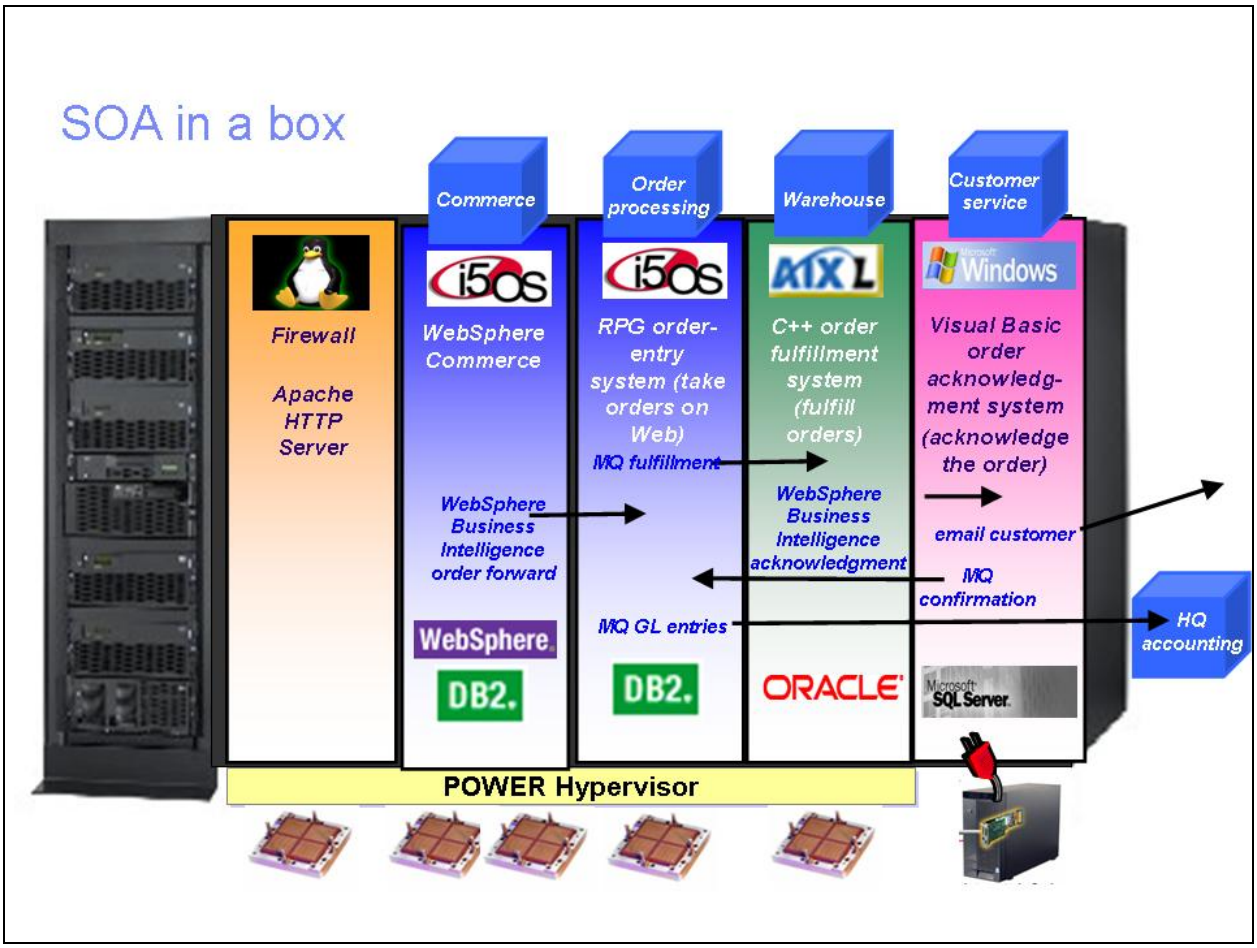


System i Tools Innovation Program: IBM and ISV tools

To supplement the IBM suite of development tools for the System i platform, IBM has partnered with many key tool vendors to provide additional options for development.

The System i Tools Innovation Program was announced in 2005 and now has 72 IBM tools registered, as well as 266 additional tools from various vendors (as of December 2006).

The tools are registered in various subcategories to define exactly what function they accomplish.



SOA in a box

This picture shows the various partitions of a System i model. (**Note:** Although this chart shows partitions with several operating system, these partitions can, instead, be operating systems that reside on separate platforms.)

This shows how an application can be made up of various business functions (listed in the boxes across the top of the chart, above each partition) that are strung together through the use of good SOA techniques. These business processes, which make up an entire order-entry application, are written in different languages, yet as each component is appropriate to accomplish the required task, this application can choose to use them.

Summary: The System i Developers Road Atlas

- Many development technologies for System i
- Many possible paths to a service-oriented environment
  - Refreshing applications
    - GUI, modular code, better structure
  - Development of new business functions
  - Inclusion of prepared components
- Accommodate the merging of new technology with traditional code
  - Integration of applications and business processes

Summary: The System i Developers Road Atlas

Version 4 of the Roadmap (now called the *Road Atlas*) has been updated to reflect numerous requests from the development community that relate to modernization. However, it also includes new development tools and encompasses a methodology regarding how the service oriented architecture (SOA) fits into the System i development strategy.

As with each, the modernization strategy is a step-by-step approach to build skills and to rework existing applications. These steps allow a gradual evolution rather than an all-at-once approach. Depending on business requirements, some steps can be skipped or accelerated to reach the ultimate goal. However, whether modernizing, developing new or locating components elsewhere, all are steps toward implementing SOA. This presentation discussed these steps.

The System i Developers Road Atlas has three main goals:

- Moving to graphical interfaces
- Moving to modern compilers that support additional functions, integration, modularization and code reuse
- Moving to SOA, where appropriate (This type of architecture allows applications to talk to each other, both within one enterprise or across enterprises.)

By restructuring application code into smaller modules, shops position their code for future integration with emerging technologies. An example of this is the use of Web services. This function is impossible if the application code is still in a large, multifunctioned program structure.

The Road Atlas accommodates applications at all stages of development. Based on existing skills and technology, development shops might determine where it is most appropriate to enter the steps of the Road Atlas. And of course, based on longer-term objectives, they can decide where to leave the roadmap. This allows shops and developers to build skills gradually as required to implement the business objectives.

Note: This is a living document. It will continue to be enhanced and clarified based on feedback and experience from System i clients, Business Partners and solution providers.

Resources

- IBM eServer iSeries Information Center
  http://publib.boulder.ibm.com/infocenter/iseries/v5r4/index.jsp

- IBM Publications Center
  www.elink.ibmlink.ibm.com/public/applications/publications/cgibin/pbi.cgi?CTY=US

- IBM Redbooks™
  www.redbooks.ibm.com/

- “Ready for Rational” validation Web site
  www-304.ibm.com/jct09002c/isv/rational/readyfor.html

- Eclipse Plug-in Central Web site
  www.eclipseplugincentral.com

Trademarks and special notices

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

AS/400, ClearCase, Domino, i5/OS, IBM, the IBM logo, Integrated Language Environment, iSeries, Lotus, Net.Data, OS/400, Passport Advantage, Rational, Redbooks, System i, VisualAge and WebSphere are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.