

# Faster Application Change and Reuse with WebSphere Studio Asset Analyzer

Gain intellectual control over your applications

Cut through application complexity

Make changes with more confidence



Daniel Moul  
Paul Hensler





International Technical Support Organization

**Faster Application Change and Reuse with WebSphere  
Studio Asset Analyzer**

February 2008

**Note:** Before using this information and the product it supports, read the information in “Notices” on page v.

**First Edition (February 2008)**

This edition applies to Version 4, Release 2, Modification 1 of WebSphere Studio Asset Analyzer (product number 5655-M22).

**© Copyright International Business Machines Corporation 2008. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

# Contents

<b>Notices</b> .....	v
Trademarks .....	vi
<b>Preface</b> .....	vii
The team that wrote this IBM Redpaper .....	vii
Become a published author .....	viii
Comments welcome .....	viii
<b>Chapter 1. The promise and challenges of reuse</b> .....	1
1.1 Size and complexity .....	2
1.1.1 Cost of change .....	3
1.1.2 Reluctance to change .....	3
1.1.3 Ever-changing market and regulatory conditions .....	4
1.1.4 Service-oriented architecture .....	5
1.1.5 SOA entry points .....	6
1.1.6 The “as is” and the “to be” .....	6
1.2 A brief roadmap for this paper .....	7
<b>Chapter 2. Using WebSphere Studio Asset Analyzer to cut through the complexity of application changes</b> .....	9
2.1 Introducing WebSphere Studio Asset Analyzer .....	10
2.2 The inventory process .....	11
2.3 Open architecture .....	12
2.4 Application understanding by exploring the inventory .....	12
2.5 Comparing TSO search capabilities .....	21
2.6 “Follow that data” with the impact analysis engine .....	22
2.7 Java application exploration and impact analysis scenario .....	30
2.8 Limitations of static analysis .....	33
2.9 Answering your own questions with custom queries .....	34
<b>Chapter 3. Approaches to discovering code for reuse</b> .....	35
3.1 Approaches to discovery .....	36
3.1.1 Starting with data .....	36
3.1.2 Starting with programs .....	37
3.2 Technical challenges to reuse: Change complexity .....	42
3.2.1 Intellectual control .....	42
3.2.2 Interfaces .....	42
3.2.3 Intermingled business logic .....	42
3.2.4 Application pattern .....	42
3.3 Non-technical challenges to reuse .....	43
<b>Chapter 4. Accelerate test case generation with WebSphere Studio Asset Analyzer</b> .....	45
4.1 Identifying test cases .....	46
4.1.1 Identifying the resources required to conduct the test .....	46
4.1.2 Feeding an automated test bed .....	46
<b>Chapter 5. Additional value through extension and integration with other tools and metadata repositories</b> .....	51
5.1 Benefits of an open architecture .....	52
5.1.1 Extending the metadata (tagging) .....	52

5.1.2 Using Web service . . . . .	52
5.1.3 Using the open metrics framework . . . . .	53
5.1.4 Using your favorite means of issuing SQL queries . . . . .	53
5.1.5 Potential drawbacks . . . . .	55
5.1.6 Using the URL application programming interface . . . . .	55
5.2 Related tools . . . . .	55
5.2.1 Asset Transformation Workbench and Rational Transformation Workbench . . . . .	55
5.2.2 Rational Developer for System z . . . . .	59
5.2.3 Rational Asset Manager . . . . .	59
5.2.4 CICS Interdependency Analyzer . . . . .	60
5.2.5 WebSphere Service Registry and Repository . . . . .	61
5.2.6 Data dictionaries . . . . .	61
5.2.7 Tivoli Change and Configuration Management Database . . . . .	61
5.2.8 IBM Information Server . . . . .	62
<b>Appendix A. Deployment options for WebSphere Studio Asset Analyzer . . . . .</b>	<b>63</b>
<b>Related publications . . . . .</b>	<b>67</b>
Online resources . . . . .	67
How to get IBM Redbooks publications . . . . .	68
Help from IBM . . . . .	68
<b>Index . . . . .</b>	<b>69</b>

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

Redbooks (logo) ®

z/OS®

zSeries®

z9™

AIX®

Component Business Model™

CICS®

DB2 Connect™

DB2®

DRDA®

IBM®

IMS™

QMFT™

Rational®

Redbooks®

REXX™

System z™

System z9®

Tivoli®

WebSphere®

The following terms are trademarks of other companies:

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

Excel, Windows, 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.



# Preface

This IBM® Redpaper is written for analysts, developers, architects, and technical managers with the goal of introducing ideas about mainframe application change and reuse using WebSphere® Studio Asset Analyzer and related tools from IBM.

This paper focuses on mainframe applications for the following reasons:

- ▶ There are a lot of these types of applications.
- ▶ It can be very challenging to understand these applications well enough to take action to change or reuse them.
- ▶ There has been less focus on the *how* of mainframe application reuse than, for example, the reuse of Java™ objects and applications.

For these reasons, even though WebSphere Studio Asset Analyzer does support Java application analysis, you find only occasional references to it in this paper.

In the spirit of “a picture is worth a thousand words” and because some people learn best through illustrations, some chapters in this book include a large number of figures to illustrate concepts and features. This is not an apology. If this is not your favorite learning style, feel free to skim to the next section.

This paper shows figures for WebSphere Studio Asset Analyzer V4.2. While we were writing this paper, the development team released WebSphere Studio Asset Analyzer V5.1. The ideas and presentation in this paper remain relevant with WebSphere Studio Asset Analyzer V5.1. These are still the early days of service-oriented architectures. Our tools and techniques will continue to evolve in ways that can help us be more productive in pursuit of the same goals. If you have stories to tell, questions, or suggestions, we invite you to contact us at:

- ▶ Daniel Moul: [dmoul@us.ibm.com](mailto:dmoul@us.ibm.com)
- ▶ Paul Hensler: [phensler@us.ibm.com](mailto:phensler@us.ibm.com)

## The team that wrote this IBM Redpaper

This paper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Raleigh Center.

**Daniel Moul** is the IBM product line manager who is responsible for WebSphere Studio Asset Analyzer and Asset Transformation Workbench. For the last decade, he has worked on and around IBM enterprise application servers and related tools in roles, including technical support manager, technical planner, development program manager, project manager, and product line manager. Before joining IBM, Daniel was the General Manager of South Seas Computing (Fiji) Ltd.

**Paul Hensler** has nearly 30 years of IT consulting experience. He is an IT Specialist in IBM Global Business Services currently specializing in documenting, analyzing, and improving existing application code. He has worked on several client implementations of WebSphere Studio Asset Analyzer.

Thanks to the following people for their contributions to this project:

Amy Silberbauer, IBM, Silicon Valley Laboratory

John DeIMonaco, IBM, Chicago

Leshek Fiedorowicz, IBM, Silicon Valley Laboratory

Michelle Cordes, IBM, Baton Rouge

Tony Llopis, IBM, Dallas

and

Joe DeCarlo, Manager, Special Projects, ITSO, Raleigh Center via San Jose

## Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbooks® publication dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our papers to be as helpful as possible. Send us your comments about this Redpaper or other Redbooks in one of the following ways:

- ▶ Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- ▶ Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400



# **The promise and challenges of reuse**

Today's enterprises could not exist without information technology. In fact, millions of lines of application code embody and enable business processes. These applications can be a source of competitive advantage as well as a brake on innovation.

Existing systems have evolved over decades into massive sets of interrelated, complex combinations of computers, network equipment, middleware servers, storage systems, custom, and off-the-shelf software applications, as well as the data that these systems consume and generate.

In this chapter, we examine the promise and the challenge of reuse. We also provide a brief roadmap for this paper.

# 1.1 Size and complexity

Within an organization’s many existing systems, each domain is in itself bewilderingly complex. One can develop a career-long specialty in sub-domains that are associated with any one of them.

For our purposes in this paper, we focus on the custom-written applications often found at the heart of this complexity. The combined labors of millions of person-years of effort are expressed in the source code and configurations of the custom-written applications. Much of it is expressed in the COBOL language, is running on mainframe computers, and is invested in the terms of many dollars of application development over the last four decades or more.

Within a single large enterprise, individual applications range in size from thousands of lines to many tens of millions of custom-developed lines of code (LOC). Together, these can run to over a hundred million.

For example, a global manufacturing company has over 177 000 000 LOC as shown in Table 1-1. This table represents only the traditional mainframe language source files currently inventoried in WebSphere Studio Asset Analyzer and does not include any WebSphere applications that are running in WebSphere Application Server for z/OS® or portions of the applications running on UNIX®, Linux®, personal computers, or hand-held devices

Table 1-1 Example application inventory summary of a global manufacturing company

Language	Type	File Count	Max LOC	Ave LOC	Total LOC	Standard Deviation (in lines)
ASM	INCL	430	4781	188	81 076	322
ASM	MAC	298	2149	127	38 092	263
ASM	MAIN	537	5222	359	193 232	632
ASM	MAP	7217	9134	418	3 018 904	536
COB	INCL	88 028	22 985	162	14 280 199	405
COB	MAIN	102 162	46 934	1387	141 796 522	1861
JCL	INCL	978	4462	46	45 853	157
JCL	MAIN	140 459	3476	56	7 923 408	75
JCL	PROC	71 552	5701	133	9 580 939	176
PLI	INCL	500	6894	132	66 007	431
PLI	MAIN	184	7435	638	117 537	1199
UNK?	UNK?	77	2690	237	18 274	451
Total		412 422			177 160 043	

To summarize in round numbers, Table 1-2 shows the totals for this enterprise.

Table 1-2 Enterprise totals

177 000 000	Lines of code (LOC) that consists of:
200 000	Program and copybooks
140 000	Batch jobs

For this enterprise and not shown in Table 1-2, the inventory also references or includes the totals shown in Table 1-3.

Table 1-3 Additional enterprise totals

126 000	DB2® columns
1 344 000	Data sets
2 400 000	Program literals
81 000 000	Data elements in the programs

There are enterprises that run more than 1000 separate applications, with the number of interfaces between these applications many times larger. These applications are typically a mix of custom-written and vendor-developed applications with some customizations. Both intra-application and cross-application dependencies must be considered when making changes. For example, one medium-sized financial institution was found to have 250 applications with 1700 dependencies between them, an average of about seven cross-application dependencies for each application.

### 1.1.1 Cost of change

Perhaps, then, given the size of application inventories, it is no wonder that change is slower and more costly than most enterprises would want. Concerns about speed to market and cost of change must be balanced with those of quality and avoidance of outages. Analysts and developers who implement a change must be thorough yet must avoid analysis paralysis. After the code is changed, it must be adequately validated in a Quality Assurance process before it is moved into production.

It is commonly noted that enterprises typically spend 60% to 80% of IT budgets on maintaining existing applications (including operations and application maintenance), leaving limited resources for new initiatives that matter to the boardroom—initiatives that provide strategic value and grow market share.

This cost of change in time and money acts as a *brake*, slowing change and limiting the enterprise's ability to respond to market threats and opportunities. Application changes can be the critical path in implementing new products and services, especially in financial services. Adding to the challenge there is little or no physical product to manufacture. Other industries, in which improving IT return on investment might have been expressed more in driving down costs than in improving productivity, leave IT organizations without the capacity to respond effectively to business requirements.

### 1.1.2 Reluctance to change

The size and complexity of these applications engender a reluctance to change them. Major re-engineering projects have generally fallen out of favor due to their low success rate in meeting commitments to schedule, quality, or cost. It can be challenging to gauge the complexity of a potential project before a large amount of money has been committed to it. These projects can be a career risk for those who sponsor and lead them.

Even minor application changes can be expensive and difficult to get right. Challenges to implementing changes in existing applications include:

- ▶ **Time consuming:** System documentation is rarely current and often not fully trustworthy, meaning that an analyst or developer must undertake a research project simply to estimate the size of the project.
- ▶ **Risk to application stability:** It can be difficult to identify the set of assets that are affected by a proposed change. This *bill of materials* is needed for project estimation, the developer's task list, and the identification of the test environment needed to validate the changes. Textual scans looking for variable names and so forth typically find less than 80% of the relevant assets (and, at the same time, produce many false-positives that must be weeded out). The remainder of the relevant assets are typically determined manually by inspection and “seeing what breaks” while in the process of rebuilding the programs or testing the changes.
- ▶ **Cross-application dependencies:** Nearly all enterprise applications share data with, directly call, or are called by other applications, which creates cross-application dependencies. Changes in one application, for example to the format of a data file or database table, can break another application. It can be difficult to determine which applications have dependencies on your application or its data, and even if you can determine the relevant programs, it can be non-trivial to determine who is responsible for that program or application within the enterprise.
- ▶ **Composite application dependencies:** It is increasingly common for application components to span multiple runtime systems. For example, the data access and business logic portions of an application might be expressed in COBOL running in a CICS® container, while the user interface portion might be running in WebSphere Application Server EJB™ and servlet containers. Similarly, applications composed of sets of Web services (for example, running in the WebSphere Process Server) are agnostic to the underlying runtime that is hosting the service implementation.
- ▶ **Application Entropy:** Over time, applications are modified by people who might be unaware of the original design or who might be addressing new requirements that do not fit easily into the original system. As system complexity rises, so does the likelihood that the analyst or developer will miss some of the less obvious issues that need to be addressed for a high-quality change. Applications can become more brittle, further increasing the risk and cost of making changes.

### 1.1.3 Ever-changing market and regulatory conditions

Market and regulatory conditions are not static, thus neither are businesses and the applications that enable the businesses to function. Application change is a certainty. Along with regular maintenance activities, more involved transformation projects cannot be avoided, such as:

- ▶ Changes to key numbers (for example, due to mergers and acquisitions), customer number, contract number, order number, depot number, part number, and product number
- ▶ Government and industry trade group laws, regulations, and standards, such as security number (ISIN), bank account number (IBAN), and euro currency migrations
- ▶ Mass corrections (for example Y2K date remediation)

Over time the process of fixing an application and tactically extending it to address new requirements can lead to an increasingly complex and convoluted application. Some organizations might eventually freeze development of a core application as “too difficult” and move to incremental changes around the periphery—a strategy that simply pushes out the big issues into the future.

In any case, a day of reckoning comes sooner or later, often in the form of an assessment as part of an Application Portfolio Management review. When an application provides significant value for the medium- and long-term, the best investment is to restructure and reduce the complexity of an application to improve its capacity for reuse and simplify on-going maintenance.

Recent years have seen a number of projects defined as *disentangling, restructuring*, or otherwise modernizing mainframe applications to reposition the application for the future. These projects can be in addition to (or part of) a large transformation project as discussed previously. For example, one financial services company has an application that consists of 40 000 000 LOC that embodies about 19 major business processes. The initial project goal is to identify which programs and data are used in each of the business processes. After that, the company intends to modernize and to enable the business processes to be reused in a service-oriented architecture.

### 1.1.4 Service-oriented architecture

The recent rise of service-oriented architecture (SOA) is a response to the common desire to align IT more closely with the business of the enterprise and, at the same time, increase the flexibility of IT in meeting the needs of the business. In an SOA, IT value is delivered in *chunks* that make sense to the business and that can be combined and recombined as needed. This concept is object orientation raised to the level of business tasks.

The IBM portal for SOA is a good starting point for learning more about the concepts underlying SOA and the products, services, and methodologies that IBM and IBM Business Partners provide:

<http://ibm.com/soa>

A majority of the repeatable business tasks that one would like to make available as services are embodied in the enterprise's existing application inventory. Reusing the existing application capability is typically far less expensive than rewriting, testing, and deploying this capability elsewhere and then keeping the parallel implementations of this business task in synch. While there is cost and effort associated with reusing existing application capabilities, it can be as little as one fifth the cost or less compared to re-implementing it. For more information, see the following online article:

<http://archive.bcs.org/BCS/Products/publishing/itnow/OnlineArchive/sep98/software reuse.htm>

You can also consult our list of other resources in "Related publications" on page 67 for more information.

## 1.1.5 SOA entry points

*Reuse* is one of the five common *entry points* to an SOA. The entry points include both business- and IT-oriented entry points.

**Note:** These entry points were identified after reviewing more than 1800 SOA-related client IBM engagements. For more information on the five common SOA entry points and how they were identified, see the following online article:

[http://www.ibm.com/innovation/us/pointofview/soa/apr03/getting\\_started.html](http://www.ibm.com/innovation/us/pointofview/soa/apr03/getting_started.html)

The business-oriented entry points include:

- ▶ **People:** Connect people to the information and processes that they need to achieve new levels of productivity and collaboration.
- ▶ **Process:** Express your business processes as sets of choreographed repeatable business tasks, thus simplifying reuse, change, and business process optimization.
- ▶ **Information:** Unlock and integrate information from various data silos in your organization and make it available as services.

The IT-oriented entry points include:

- ▶ **Reuse:** Make use of existing application assets in new ways as *services*.
- ▶ **Connectivity:** Solve application connectivity challenges through open standards-based Web services and an Enterprise Service Bus.

For information about the importance of reuse in an SOA strategy, see:

<http://www.ibmpressbooks.com/title/0131870025>

You can also consult our list of other resources in “Related publications” on page 67 for more information.

## 1.1.6 The “as is” and the “to be”

In a strategic approach to implementing SOA, it is common to undertake a business process functional decomposition along the lines of the Component Business Model™ methodology developed by IBM Global Business Services. After modeling business functions, they are mapped to existing IT systems. Typically, this map identifies IT systems that overlap in function and that are aligned imperfectly with the business functions that they support. Then, when drilling in to a specific business process, it is not uncommon to find that the programs within an application do not line up at the boundaries of the functional decomposition. For example, one batch job might cross six business processes.

The discovery and analysis phase usually involves documenting the “as is” and “to be” system architectures. The complexity and structure of existing applications can make documenting the “as is” system quite challenging. It typically involves a subject matter expert (SME) or, if an SME is not available, a lot of work. Even if an SME is involved, it can be difficult to be sure that you have the “as is” system documented accurately rather than “as the SME thinks it is” or “as it was last month.”

For more information about Global Business Services Component Business Modeling, see:

[http://www.ibm.com/services/us/bcs/html/bcs\\_componentmodeling.html](http://www.ibm.com/services/us/bcs/html/bcs_componentmodeling.html)



## 1.2 A brief roadmap for this paper

WebSphere Studio Asset Analyzer can help you address many of the challenges that we have described thus far. By inventorying your application source and organizing metadata about your heterogeneous software assets in an accessible way, WebSphere Studio Asset Analyzer helps to address the basic challenges of gaining intellectual control of your applications and accomplishing application change.

In the paper, we first introduce WebSphere Studio Asset Analyzer with these two goals in mind. Then, we look at two special cases in which WebSphere Studio Asset Analyzer accelerates projects relate to change:

- ▶ Identifying code for reuse
- ▶ Preparing test cases to validate changes

Finally, we look at integration with other tools and then end with brief summaries of deployment scenarios for WebSphere Studio Asset Analyzer.

**Note:** WebSphere Studio Asset Analyzer can help you when you are:

- ▶ Analyzing applications for reuse or other transformation projects
- ▶ Preparing to test the resulting changes
- ▶ Performing on-going maintenance after the changes





## Using WebSphere Studio Asset Analyzer to cut through the complexity of application changes

In this chapter, we introduce WebSphere Studio Asset Analyzer, its parts, and the information it collects and organizes that make it useful. We look at some scenarios that showcase its usefulness. This introduction provides a foundation for the next chapter, which applies the application insight in WebSphere Studio Asset Analyzer to the question of reuse.

## 2.1 Introducing WebSphere Studio Asset Analyzer

WebSphere Studio Asset Analyzer is an application metadata repository that is derived from a static analysis of source code and the configurations of the database and transactional systems in which the applications run. It consists of the following components:

- ▶ Source scanners running on z/OS and, optionally, distributed scanners running on AIX® or Windows®.
- ▶ Metadata repository in DB2 on z/OS.
- ▶ WebSphere applications running on z/OS, AIX, and Windows systems that provide the user interface, impact analysis engine, and optional distributed scanners.

See Appendix A, “Deployment options for WebSphere Studio Asset Analyzer” on page 63 for more information about deployment configurations and support of other platforms including Linux and Solaris™.

Figure 2-1 shows the major components of WebSphere Studio Asset Analyzer that take part in the process of collecting inventory, doing analysis, and presenting metadata about software artifacts.

The first step is to inventory your applications:

- ▶ Analyze source and Java bytecode to capture relevant information about the objects in the application (program names, data stores, program tokens, data elements, control transfers, relationships between programs and data, program metrics, and so forth).
- ▶ Capture some system configuration information to provide context.
- ▶ Store this metadata in a DB2 database.
- ▶ As part of the process, figure out what is missing or inconsistent in your inventory (missing programs, bad JCL, program name clashes, and so forth).

After inventorying an application, its metadata becomes the source of application insight, which can be used for, among other things, rapid application understanding and change impact analysis. The application insight is accessible in multiple ways:

- ▶ People access this application insight through a browser interface.
- ▶ Other tools and programs use a Web service interface or make direct DB2 SQL calls.

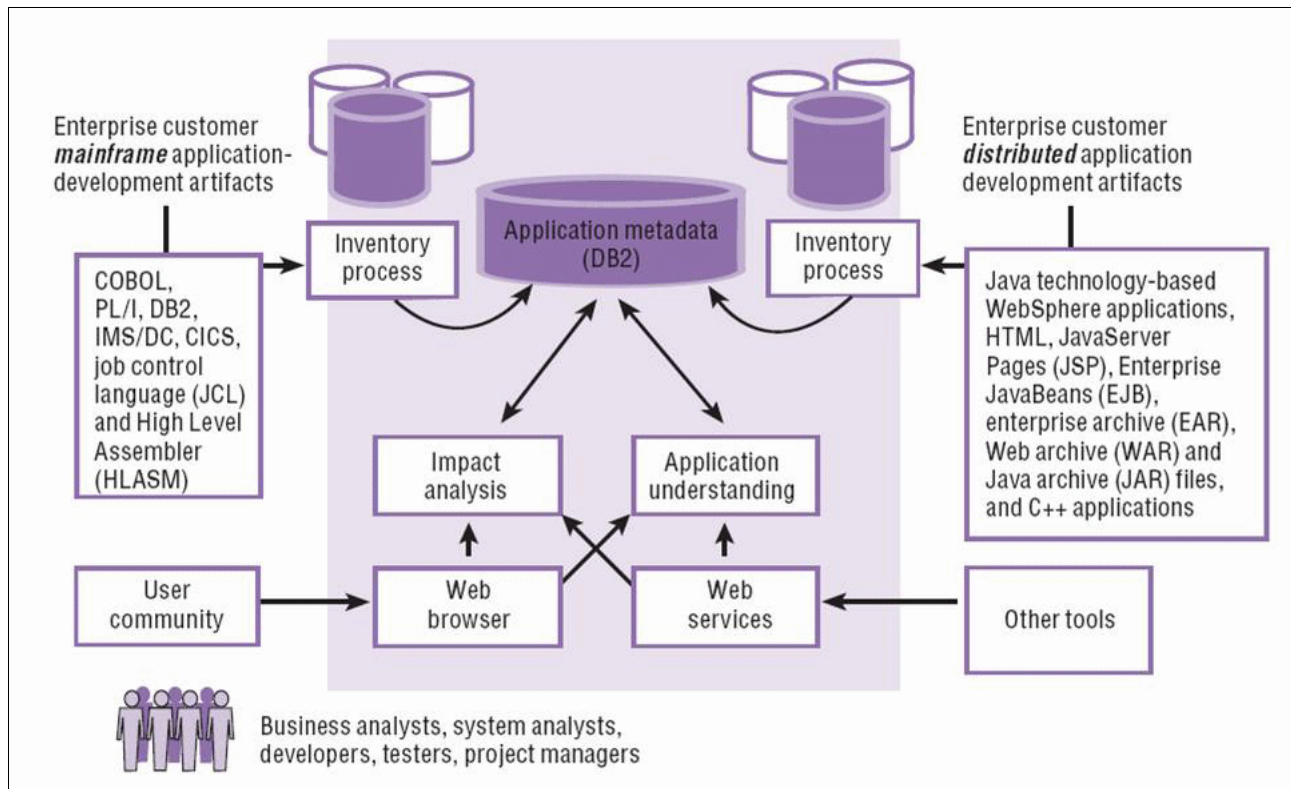


Figure 2-1 Major functions and components of WebSphere Studio Asset Analyzer

## 2.2 The inventory process

WebSphere Studio Asset Analyzer scans mainframe and distributed software artifacts, storing related information in a DB2 repository that resides on the mainframe. You do not have to download your mainframe source code to a workstation, nor do you have to upload your distributed application inventory to a mainframe. WebSphere Studio Asset Analyzer scans source where it lives, whether that is in PDS/PDSE data sets on the mainframe, directories on distributed servers, or one of a number of software configuration management systems (SCMs). WebSphere Studio Asset Analyzer also scans mainframe CICS and IMS™ subsystems, the DB2 catalog, and WebSphere Application Server domains. After the metadata is collected and post-processed, you can ask various interesting questions about your application.

The forty-plus distributed scanners read .war and .ear files from a local file system as defined in one or more scan roots. They also scan WebSphere Application Server configurations to find database connections (JDBC™), Java Connector Architecture (JCA) connectors, and other interesting attributes of the application server environment in which applications are deployed.

The repository created by WebSphere Studio Asset Analyzer corresponds to your applications at the time each asset was inventoried. You can think of it as a snapshot of your application. It typically corresponds to the code that is running in production or currently in test. To keep the repository up to date, it is common practice to tie WebSphere Studio Asset Analyzer into the change management process, for example by alerting WebSphere Studio Asset Analyzer when a new configuration level is created in the library management system or moved to production. Then the changed programs are inventoried again.

## 2.3 Open architecture

The open architecture of WebSphere Studio Asset Analyzer offers:

- ▶ Interactive access to the WebSphere Studio Asset Analyzer application insight through a Web browser. The simple user interface minimizes training requirements, and as a server-based product licensed by mainframe value unit, the zero installation client makes it easy to roll out to the developer or analyst community.
- ▶ Programmatic access through either direct DB2 SQL queries or a Web services API.
- ▶ An open, documented database schema, which makes it possible to enhance and customize the metadata in WebSphere Studio Asset Analyzer to meet the unique requirements of your organization. WebSphere Studio Asset Analyzer includes a custom query facility, group bookmarks, and configuration of displayable metrics and statistics.
- ▶ Hyperlink access through a well defined URL syntax (the *UrlApi*) for finding information about asset types and individual asset. For programmatic access or for imbedding into documents or other electronic media, for instance.

## 2.4 Application understanding by exploring the inventory

The user interface of WebSphere Studio Asset Analyzer is organized around the idea of collecting and exploring the application inventory. By searching to limit the scope to result sets of interest and then following various hyper links, you can drill down through summary and detail pages to find answers to many questions commonly asked in the analysis process.

The following list shows examples of searches for *Applications*, *Programs*, and *Run Units*:

- ▶ List the batch jobs in an application
- ▶ List the programs in an application
- ▶ List the transactions in an application
- ▶ View the diagram for an application
- ▶ View the transaction flow for transactions in an application
- ▶ See the size and complexity of your programs
- ▶ List the batch jobs that use a program
- ▶ List the data elements used by a program
- ▶ List the data sets used by a program
- ▶ List the DB2 columns used by a program
- ▶ List the DB2 stored procedures used by a program
- ▶ List the DB2 tables used by a program
- ▶ List the programs called by a particular program
- ▶ List the programs that call another program
- ▶ List the run units that include a program
- ▶ List the transactions that include a program
- ▶ List the programs, transactions, and batch jobs that use an included source file
- ▶ View the control flow for a program
- ▶ Browse the source code for a program
- ▶ List the batch jobs and transactions that use a run unit
- ▶ List the entry points in a run unit
- ▶ List the programs that make up a run unit
- ▶ List the source needed to build a run unit
- ▶ View the calling hierarchy for a run unit

The following list shows examples of searches for *Data Elements*, *Data Sets*, and *DB2*:

- ▶ List the programs that use a data element with a given name
- ▶ List the data elements with certain data characteristics (type, length, and so forth)
- ▶ List the batch jobs and steps that use a data set
- ▶ List the programs that access a data set
- ▶ List the transactions that access a data set
- ▶ List the programs that use a DB2 column
- ▶ List the programs that use a DB2 stored procedure
- ▶ List the programs that use a DB2 table
- ▶ List the columns in a DB2 subsystem
- ▶ List the run units that use the DB2 subsystem
- ▶ List the stored procedures in a DB2 subsystem
- ▶ List the tables in a DB2 subsystem
- ▶ List the views in a DB2 subsystem
- ▶ List the views that use a DB2 column
- ▶ List the views that use a DB2 table

The following list shows examples of searches for *IMS*:

- ▶ List the transactions, DBDs, PSBs, online data stores, containers (ACB, DBD, PSB libraries) associated with an IMS subsystem
- ▶ List the segments, PSBs and data sets associated with a DBD
- ▶ List the run units, PSBs, entry points and programs associated with an IMS transaction
- ▶ List the PCBs, Batch Jobs, Transactions associated with a PSB
- ▶ List the fields in an IMS segment as well as the run units using a PSB that references the segment
- ▶ List the batch jobs that use a PSB
- ▶ List the DBDs for an IMS subsystem
- ▶ List the IMS programs
- ▶ List the IMS subsystems for a site
- ▶ List the IMS subsystems that use a DBD
- ▶ List the IMS subsystems that use a PSB
- ▶ List the PCBs defined for a PSB
- ▶ List the PSBs for an IMS subsystem
- ▶ List the PSBs that use a DBD
- ▶ List the segments in a DBD
- ▶ List the transactions that use a PSB
- ▶ View the transaction flow for an IMS transaction

The following list shows examples of searches for *CICS*:

- ▶ List the CICS files and groups for an online region
- ▶ List the CICS programs
- ▶ List the CICS regions for a site
- ▶ List the files that define a BMS map or map set
- ▶ List the maps in a BMS map set
- ▶ List the programs that use a BMS map or map set
- ▶ List the transactions in a CICS group
- ▶ View the flow of transactions in a CICS group
- ▶ View the transaction flow for a CICS transaction

The following list shows examples of searches for *batch jobs*:

- ▶ Browsing the source for a batch job
- ▶ List the INCLUDEs and PROCs used by a batch job
- ▶ List the run units invoked by a batch job
- ▶ View a diagram of a batch job

The following list shows examples of searches for *Java* applications:

- ▶ Find interrelationships among components
- ▶ Find where a component is referenced in the code
- ▶ Find the “seams” in an application when partitioning code
- ▶ List the most recently modified archive files
- ▶ View a count breakdown of archive files found in a scan root
- ▶ View a count breakdown of EJBs by type
- ▶ View the count breakdown of EJBs in each EJB-JAR
- ▶ View the EJBs defined in an EJB-JAR file
- ▶ Determine the primary developer of a component
- ▶ Count Java bytecode classes by type
- ▶ View the classes in a Java package
- ▶ View the incoming and outgoing references to any classes, methods, and fields in a Java package
- ▶ View JSP™ files ordered by file size
- ▶ View the custom tags used in a JSP
- ▶ View any EJB references that are defined in a WAR file
- ▶ View any incoming or outgoing references to a particular WAR file or any assets in the WAR file
- ▶ View the servlets defined in a WAR file

There are various ways to scope the result set. Figure 2-2 shows a basic search that uses asset names.

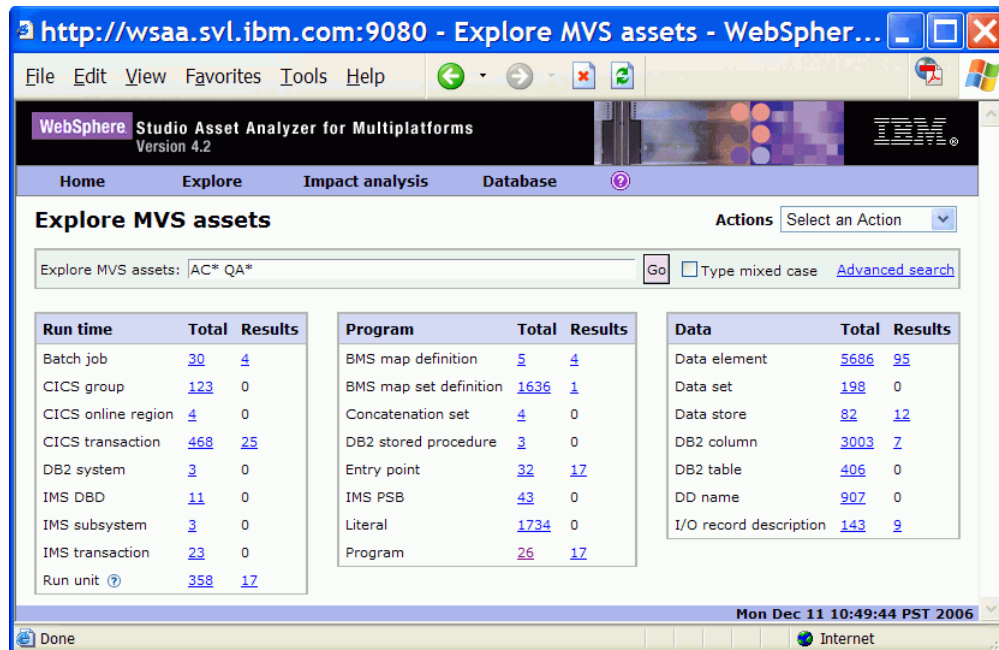


Figure 2-2 Basic search showing mainframe assets types with names starting with AC\* or QA\*



Advanced search options on the various asset summary pages include relevant attribute types (for example program attributes including “CICS program” or “IMS program” for data element attributes, data type and length, and so forth). In this way, the advanced search taps into the richness of the application metadata that is stored in the repository. For example, in Figure 2-3, the advanced search limits metrics results to CICS COBOL programs.

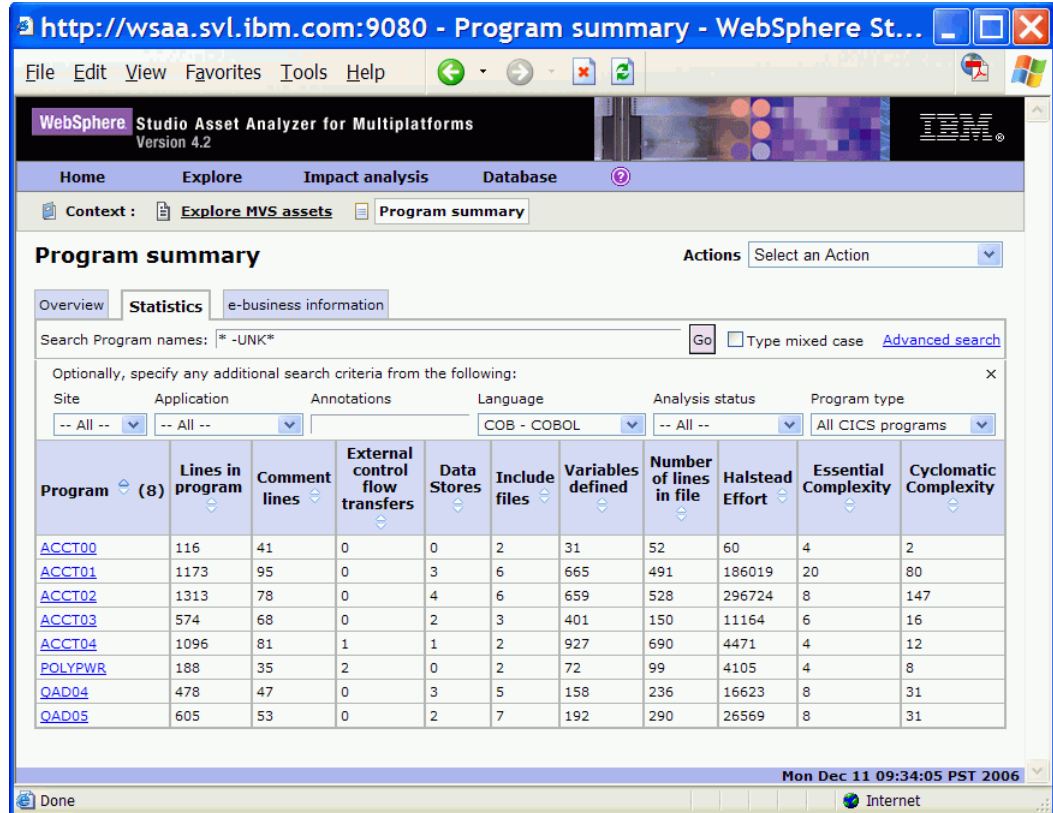


Figure 2-3 Program metric information with the result set limited to CICS COBOL programs

In Figure 2-4, note that *ACCT02* has the most cyclomatic complexity in the set. Thus, we can drill in to the program details to learn more about it, as shown in Figure 2-4.

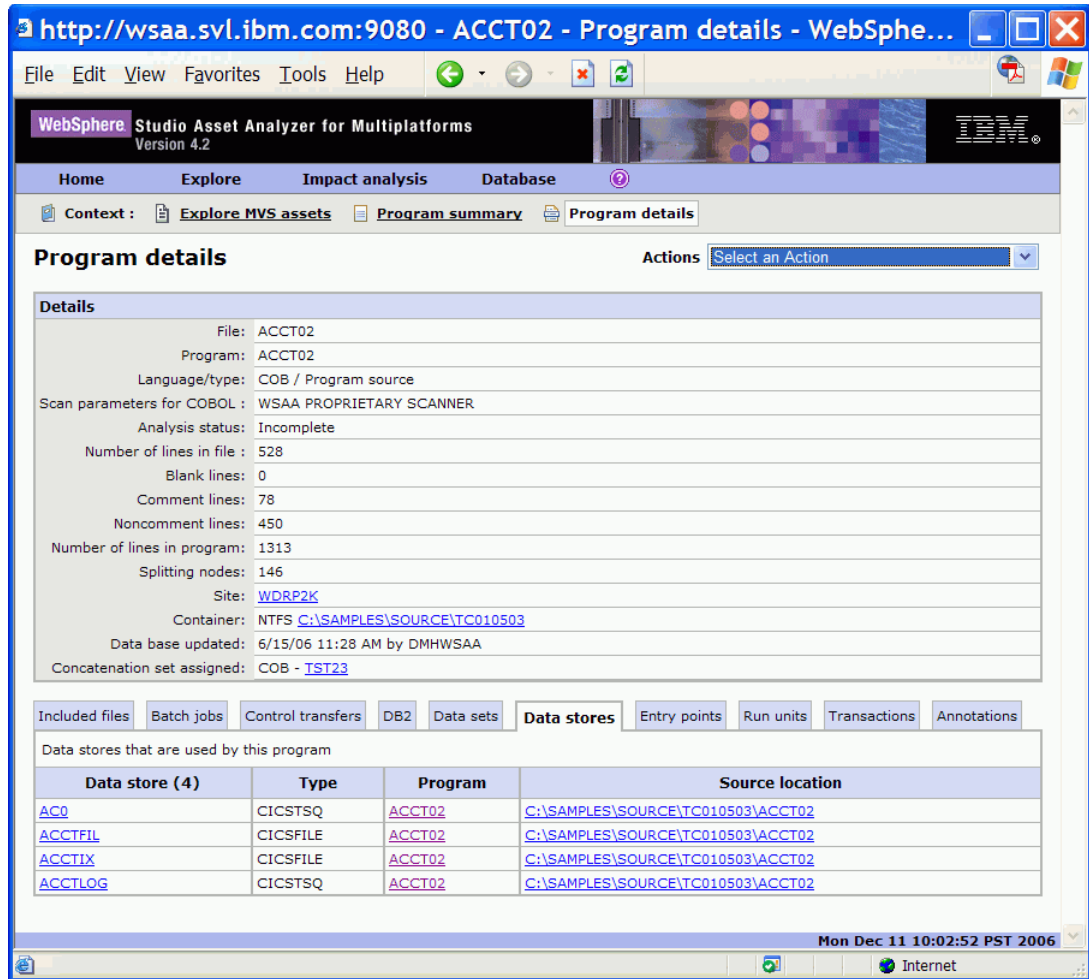


Figure 2-4 Program Details with Data stores tab active

Figure 2-5 shows visualizations of the program. Figure 2-6 on page 18 and Figure 2-7 on page 19 show the objects that it relates to, and Figure 2-8 on page 20 shows the source files that are needed to build a run unit.

**Note:** The underlined paragraph names shown in the figure are URL links that take you to a read-only copy of the source program in another browser window, positioned at that paragraph.

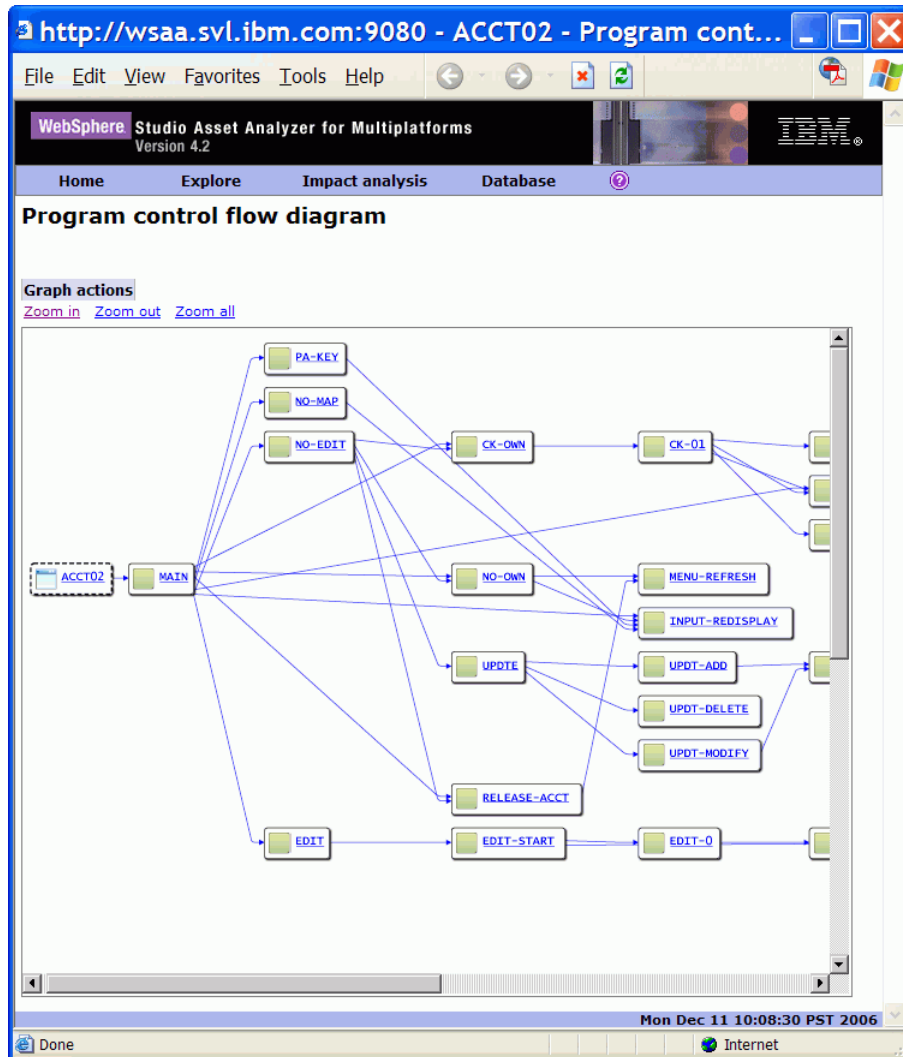


Figure 2-5 Paragraph flow in program ACCT02

In Figure 2-6, the entry point ACCT02 in program ACCT02 calls program ACCT04 and uses two files, two BMS map sets, and two CICS TSQs. Note that the lines are directional.

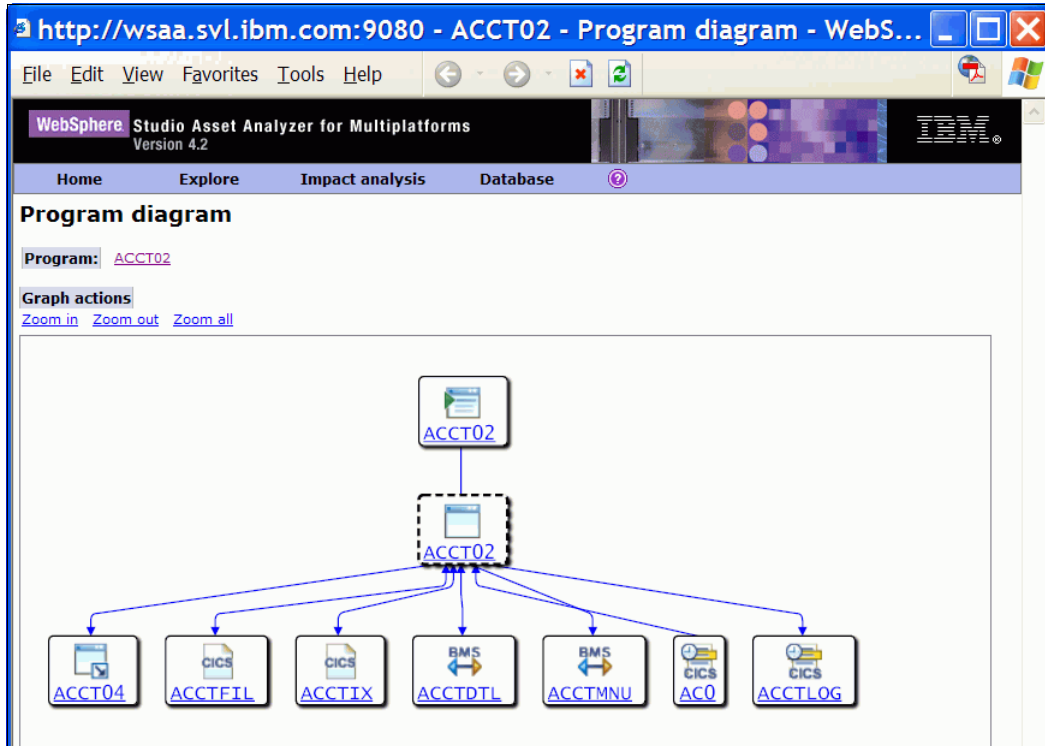


Figure 2-6 Program Diagram for ACCT02 shows the objects it relates to

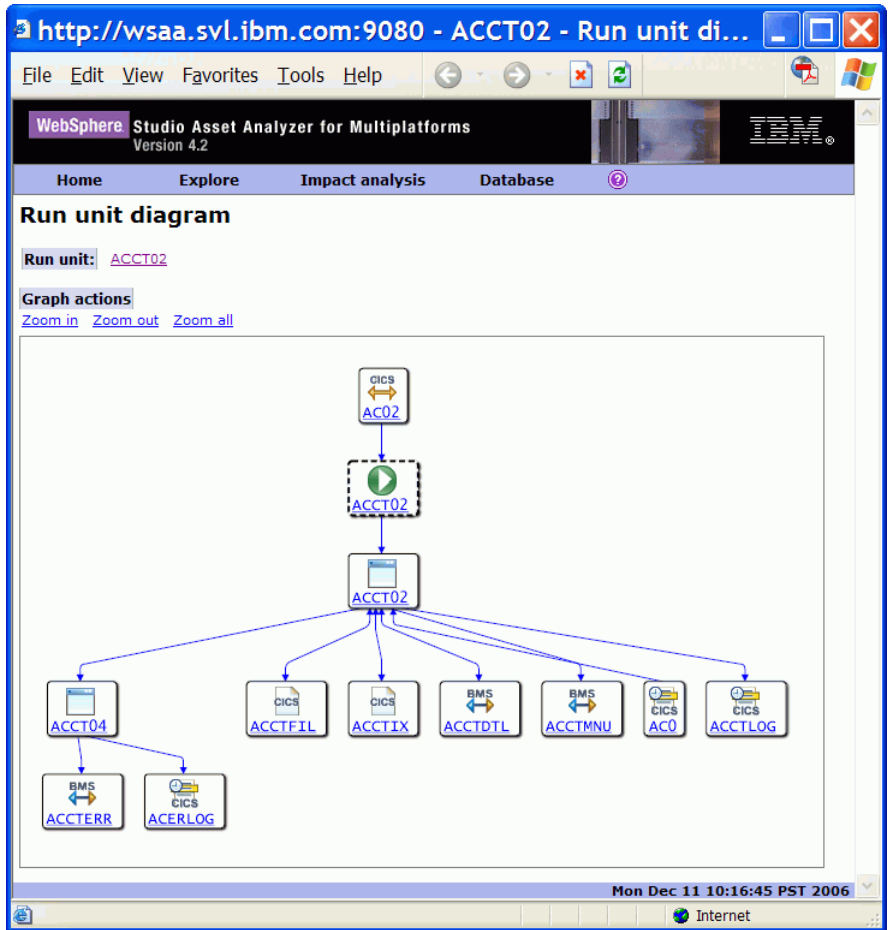


Figure 2-7 The JCL, programs, screens, and data store that are related in one program call hierarchy

In this simplistic case, shown in Figure 2-8, the Run unit diagram differs from the Program diagram only in the addition of program ACCT04 and its related objects. In most real-life scenarios, the difference would be more significant.

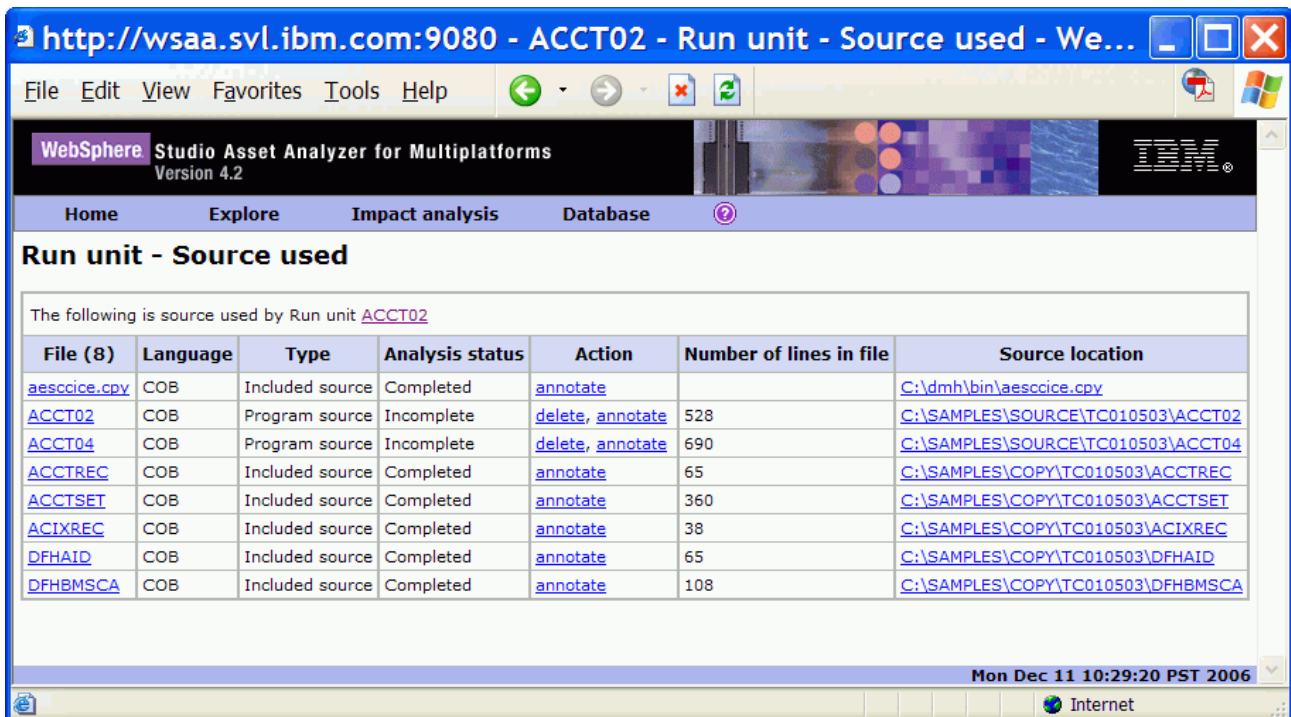


Figure 2-8 WebSphere Studio Asset Analyzer also identifies the source files need to build a particular run unit

There are also visualizations available for batch jobs. Figure 2-9 shows one step in a batch job. This example illustrates one of the strengths of WebSphere Studio Asset Analyzer. If you were to analyze this batch job manually to find all the data sources it uses, you would need to identify all the programs and review each of them. With WebSphere Studio Asset Analyzer, at a glance you can see that two DB2 tables are accessed. If you look only at the JCL, you would not locate these two tables.

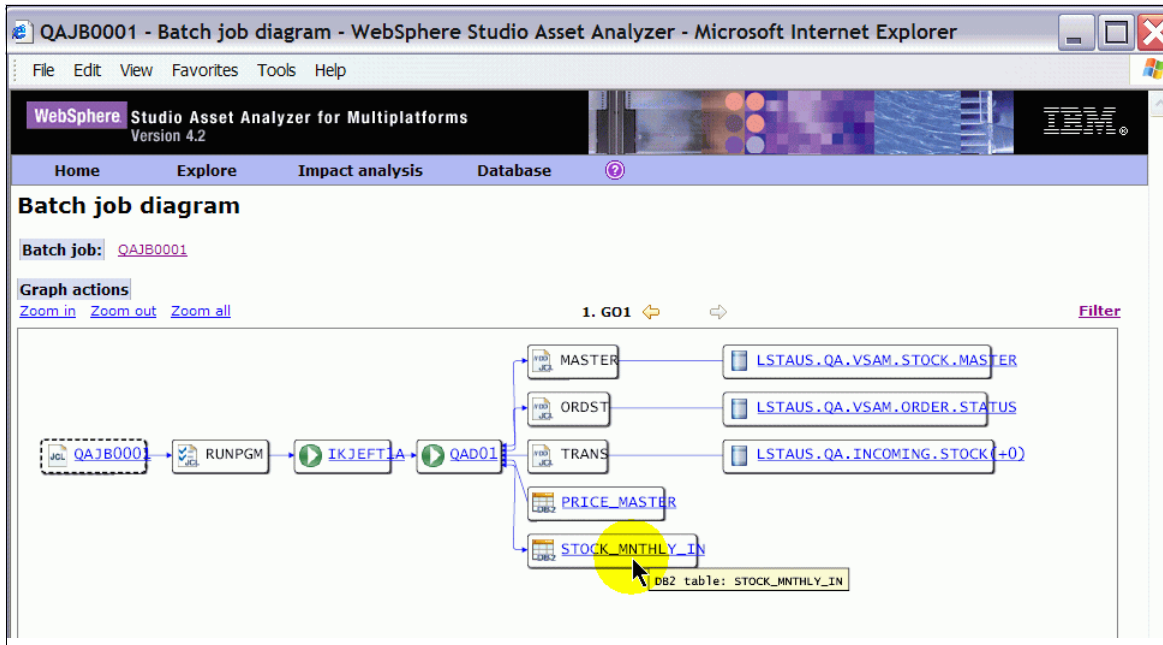


Figure 2-9 Batch job QAJB0001 shows the JCL, programs, and data sources that make up one step of the job

## 2.5 Comparing TSO search capabilities

Most large installations already have a set of tools to help them understand the relationships between the components of their applications. Using WebSphere Studio Asset Analyzer can be more effective and efficient than performing the same analysis under TSO.

For example, a simple question as to where a particular variable or copybook is used can be determined through WebSphere Studio Asset Analyzer retrieval or using the scan facilities under TSO ISPF option 3.1.4. Some key differences include:

- ▶ **Ease of use:** Under TSO, the user might have to run multiple scans. If four different libraries house programs that can imbed a copybook, all four need to be scanned. A variable search has to include copylibs as well as program libraries.
- ▶ **Existing knowledge:** The user needs to know which libraries to scan. Someone already familiar with the application structure and its libraries might know this information and can initiate the correct scans. For someone who is somewhat new to the application, this process is often a mystery, which can lead to incomplete results or unnecessary scans.
- ▶ **Formatting:** Use of REPLACE with a copybook imbed can make understanding source impacts more difficult. For example, a copybook is imbedded with a prefix of OUTREC in five programs and other prefixes in 20 others. Someone scanning for variables named OUTREC-ACCT-SUM could miss the copybook. More likely, they have to scan for everything with ACCT-SUM and then spend time weeding out the false positives. WebSphere Studio Asset Analyzer resolves these substitutions when the source is scanned so that the user can retrieve only instances of OUTREC-ACCT-SUM. Similarly,

PROCs often include symbolic substitutions in data set names. A PROC can be used multiple times, passing different data set high-level qualifiers. Again, an analyst might miss the reference or might have to cast a much wider net and find the actual uses manually. WebSphere Studio Asset Analyzer allows an analyst to find all the uses of a data set because the symbolic substitutions are resolved when the invoking JCL is scanned.

- ▶ **Resource usage:** WebSphere Studio Asset Analyzer scans the source once and logs the relationship in the repository. A “where used” might require a couple of clicks and up to a minute before the results display. Running multiple 3.1.4 scans can take much longer and consume more resource. Some installations require large scans be done in batch for that reason. For a large installation, the resource consumed with those multiple scans is considerable when compared to that consumed by a few DB2 queries. The online nature of most usage of WebSphere Studio Asset Analyzer means that analysts get their answers more quickly and in the context of their current work flow or task.
- ▶ **Scanning inactive components:** A TSO search scans every program, PROC, or copybook, including unused ones. The WebSphere Studio Asset Analyzer repository can be trimmed to reflect what is actually in use. Unused copybooks and PROCs, for example, can be identified and deleted from the WebSphere Studio Asset Analyzer repository. Typically, as much as 15% to 25% of an enterprise’s copybooks are unused. A scan that includes these inactive components creates much more work—wasted work—for the analysts.
- ▶ **Searching additional attributes:** The metadata that WebSphere Studio Asset Analyzer creates includes attributes that are associated with each asset, for example data element attributes include data type and length and program attributes include whether they run in CICS or IMS. These attributes can be used when constructing searches to find or exclude specific sets of assets.

There are also functional differences including:

- ▶ TSO scans pick up text strings in comments while WebSphere Studio Asset Analyzer will not.
- ▶ WebSphere Studio Asset Analyzer finds nested copybooks that ISPF option 3.1.4 would miss.
- ▶ WebSphere Studio Asset Analyzer searches can be restricted to where a variable is actually used rather than everywhere it is declared.
- ▶ TSO scans identify usages in components that have not been loaded into the WebSphere Studio Asset Analyzer repository.

## 2.6 “Follow that data” with the impact analysis engine

As discussed in the last section, many application dependencies are made visible through the “explore” menu options in the WebSphere Studio Asset Analyzer user interface. Additionally, sometimes it is desirable to “follow the data” to discover a set of related programs and data. This process is called *impact analysis*. Common reasons for running impact analysis include:

- ▶ Adding columns to a DB2 view or DB2 table
- ▶ Changing an interface
- ▶ Changing a transaction
- ▶ Changing the characteristics of a data set
- ▶ Changing the characteristics of a data store
- ▶ Changing the content of a Java package or .jar file
- ▶ Deleting a Java class or JSP file
- ▶ Changing the size of a data element field



- ▶ Removing methods from an EJB
- ▶ Renaming a DB2 column or table
- ▶ Renaming a servlet

Analysts might rightly consider themselves software archeologists. Researching a change requires digging down through layers of an application, typically building a graph of relevant objects in the application as they go. It is difficult to know how many levels down to pursue dependencies. Manual code inspection supplemented with textual scans through ISPF or other tools does not follow data flows and yields a high ratio of false negatives and false positives. Given the tradeoff between time requirements and completeness, developers and analysts can pursue a “good enough” approach and hope to catch anything they missed when attempting to build or test the changes.

WebSphere Studio Asset Analyzer helps to automate the discovery of these relationships. The impact analysis engine follows the movement of data and provides reports and diagrams of the affected assets.

Possible starting points for impact analysis include:

- ▶ Mainframe
  - Data element
  - Data set name
  - Data store
  - DB2 column
  - DB2 table
  - DB2 view
  - Entry point
  - Statement range
- ▶ Java
  - Archive file
  - EJB JAR
  - EJB
  - HTML file
  - Java EE Client
  - Java bytecode class
  - Java package
  - JSP file
  - Servlet
  - JSP tag library
  - JSP tag

For example, when changing data element sizes, you also need to change record I/O definitions and the related data stores (VSAM files, DB2 columns, and so forth) in your own application and also in any other applications that use the same data stores. WebSphere Studio Asset Analyzer helps you to find both the direct impacts (programmatic data movements and control transfers) and indirect impacts (other programs, JCL, and so forth) affected because they use the same data stores.

Imagine that you have a change request to expand the length of the stock part number in an application. You can search for data element naming patterns or start with a program, file, or database table that you know includes the stock number. In this case, we search for I/O record descriptions that have data elements with \*STOCK\* or \*STK\* in their names to find an entry point into the analysis. See Figure 2-10.

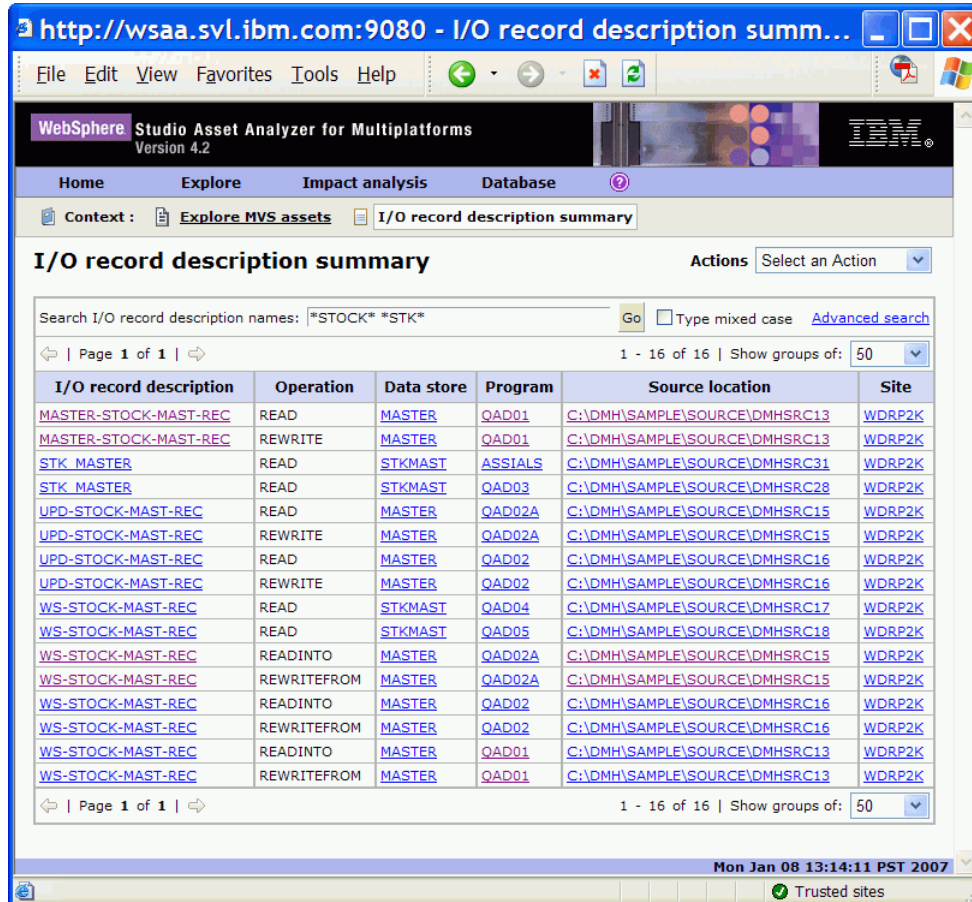


Figure 2-10 Record descriptor result set

Chances are that the record descriptions in the result set include files and you want to do an impact analysis on the relevant part number field or fields in each one. To illustrate the process, we do this only for MASTER-STOCK-MASTER-REC. Click one of its links to learn more about this record format and to confirm that it includes the stock part number, as shown in Figure 2-11.

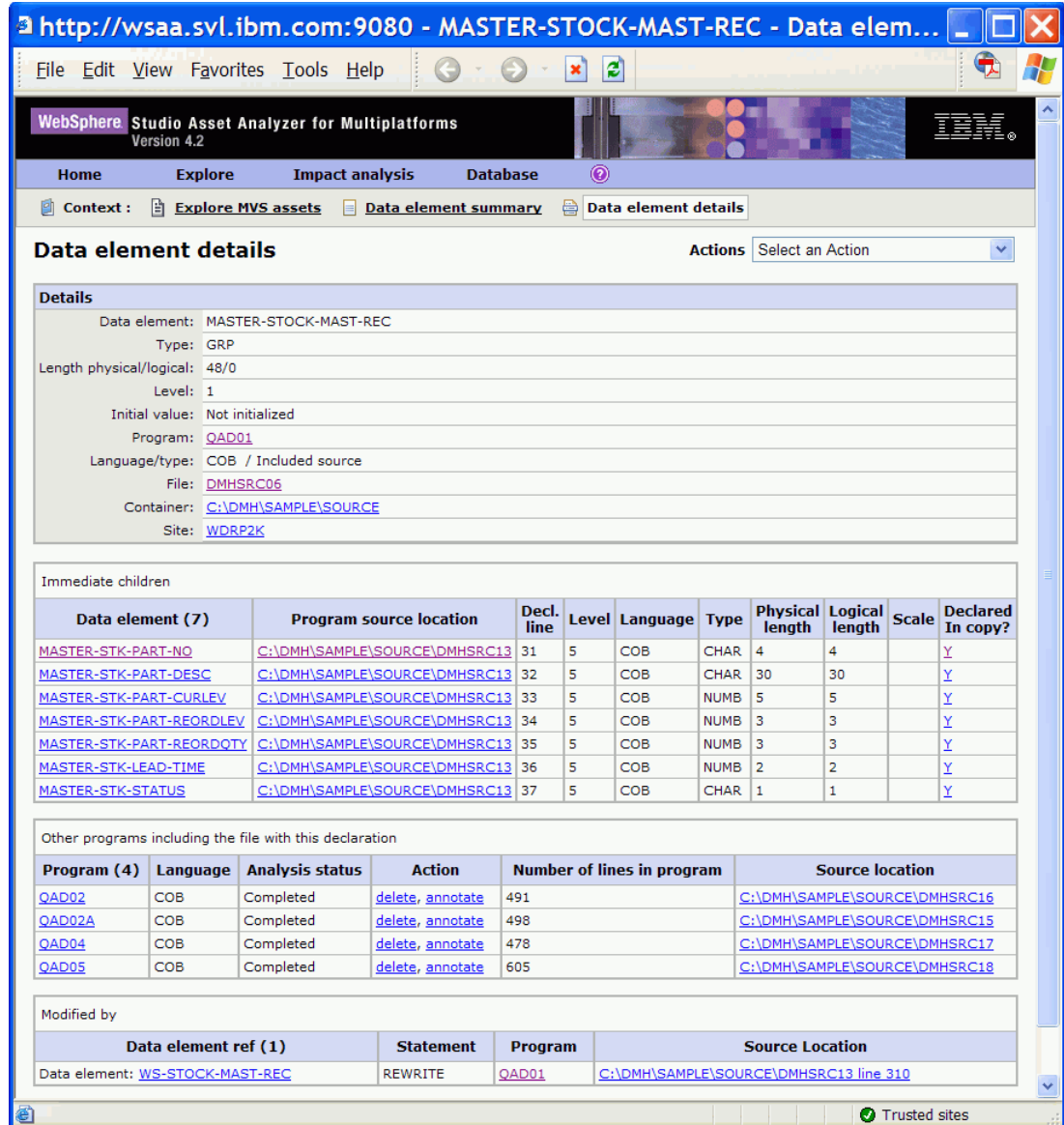


Figure 2-11 Data element details page

The record includes MASTER-STK-PART-NO, which is an included source file, for example a copybook that is defined in file DMHSRC06. Click the file name to get more information as shown in Figure 2-12.

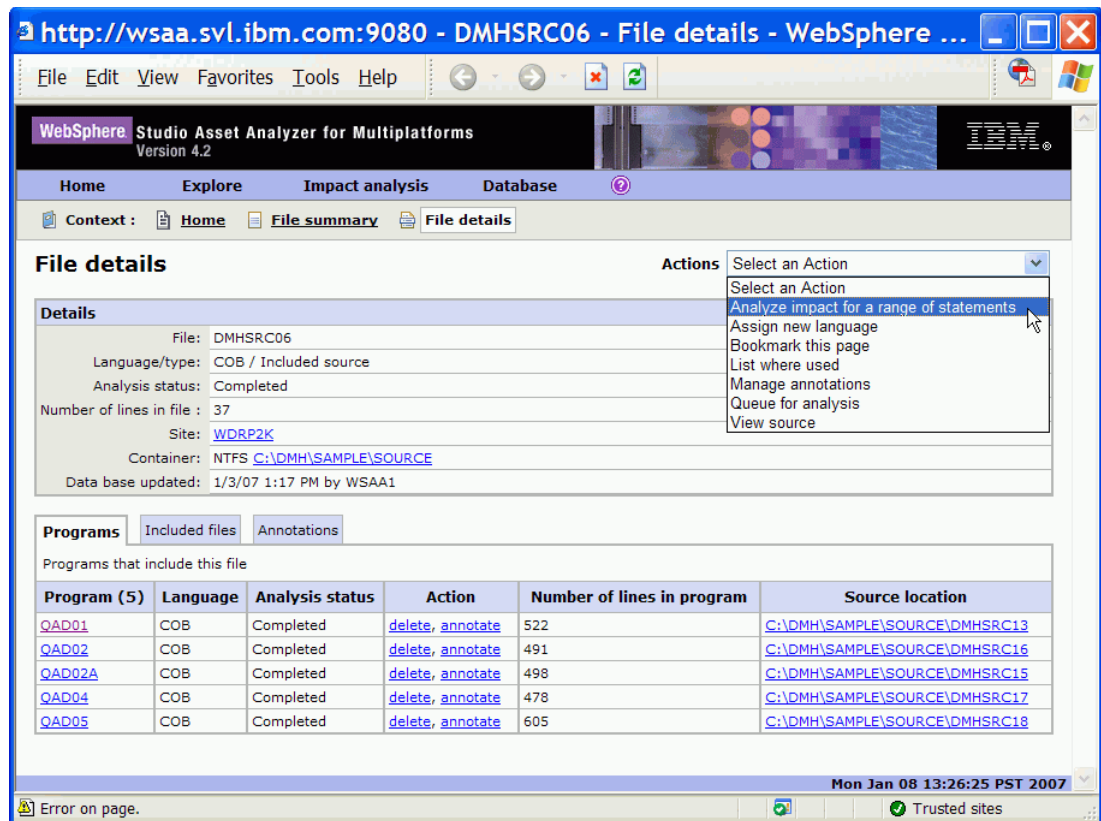


Figure 2-12 Source of copybook defining MASTER-STK-PART-NO

This copybook is used in five programs. Is the analysis now complete? No. You still do not know what transactions or JCL are associated with these program, nor do you know whether these five programs transfer control to other programs with the part number value that is transferred in the parameter list. You also do not know if there are other programs (online or batch) in other applications that use the stock part number and, even more challenging, perhaps calling it something else or using a different record layout. Now, you run an impact analysis.

You can view the source of this copybook to find the statements that you want to include in the impact analysis, in this case only line 31. as shown in Figure 2-13.

```

30.      01  :XXX:-STOCK-MAST-REC .
31.      05  :XXX:-STK-PART-NO          PIC X(04) .
32.      05  :XXX:-STK-PART-DESC        PIC X(30) .
33.      05  :XXX:-STK-PART-CURLEV      PIC 9(05) .
34.      05  :XXX:-STK-PART-REORDLEV    PIC 9(03) .
35.      05  :XXX:-STK-PART-REORDQTY    PIC 9(03) .
36.      05  :XXX:-STK-LEAD-TIME        PIC 9(02) .
37.      05  :XXX:-STK-STATUS           PIC X(01) .

```

Figure 2-13 Extract from DMHSCR06, the file containing the copybook

As shown in Figure 2-12, select **Action** → **Analyze impact of a change**. The impact analysis wizard prompts you to choose the depth of analysis and to select which sites and applications to include in the scope of the analysis.

Figure 2-14 shows an overview of the results. Impact analysis on STK-PART-NO reveals two batch jobs and two CICS transactions directly impacted through five impacted programs, indirectly an additional three CICS transactions, and seven data elements in four batch jobs are affected. However, no additional programs are impacted, because the programs that manipulate the affected seven data elements are defined to WebSphere Studio Asset Analyzer as utilities, such as SORT. Indirect impacts are the result of programs using the same physical storage but without programmatic invocation in the chain from the starting asset.

You now have the list of assets that you need to review to consider changing the length of STK-PART-NO.

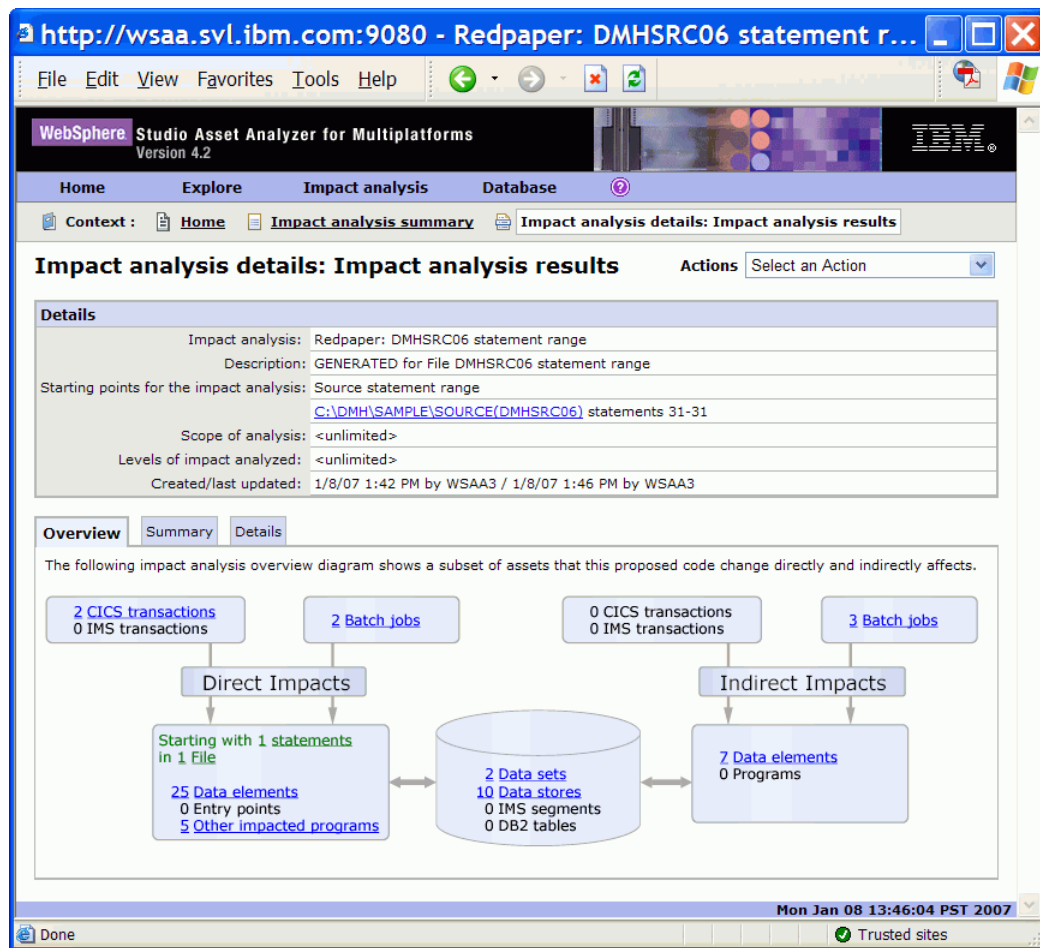


Figure 2-14 Overview of the impact analysis for STK-PART-NO in the copybook

Figure 2-15, Figure 2-16, and Figure 2-17 shows the impact analysis details page. It lists all applications, sites, programs, DB2 tables, VSAM files, data stores, programs, and so forth that are potentially affected by the proposed change.

This report can be used to generate an analysis task list for inclusion in a project plan or to help with project estimation. As with other tables that WebSphere Studio Asset Analyzer generates, it can be exported to Excel® or copied into planning or report documents.

The screenshot shows the 'Impact analysis details: Impact analysis results' page in WebSphere Studio Asset Analyzer. The page includes a 'Details' section with the following information:

- Impact analysis: Redpaper: DMHSRC06 statement range
- Description: GENERATED for File DMHSRC06 statement range
- Starting points for the impact analysis: Source statement range
- Scope of analysis: <unlimited>
- Levels of impact analyzed: <unlimited>
- Created/last updated: 1/8/07 1:42 PM by WSAA3 / 1/8/07 1:46 PM by WSAA3

Below the details is a table listing affected assets:

Application short name (1)	Name	Description
<a href="#">MVSIVP</a>		Created by the Inventory page

Container (1)	Type	File count	Site
<a href="#">C:\DMH\SAMPLE\SOURCE</a>	NTFS	72	<a href="#">WDRP2K</a>

File (11)	Language	Type	Analysis status	Number of lines in file	Source location	Site
<a href="#">DMHJCL17</a>	JCL	Batch job source	Completed	312	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL17</a>	<a href="#">WDRP2K</a>
<a href="#">DMHJCL31</a>	JCL	Batch job source	Completed	52	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL31</a>	<a href="#">WDRP2K</a>
<a href="#">DMHJCL32</a>	JCL	Batch job source	Completed	41	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL32</a>	<a href="#">WDRP2K</a>
<a href="#">DMHJCL37</a>	JCL	Batch job source	Completed	26	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL37</a>	<a href="#">WDRP2K</a>
<a href="#">DMHJCL40</a>	JCL	Batch job source	Completed	243	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL40</a>	<a href="#">WDRP2K</a>
<a href="#">DMHSRC06</a>	COB	Included source	Completed	37	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC06</a>	<a href="#">WDRP2K</a>
<a href="#">DMHSRC13</a>	COB	Program source	Completed	400	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC13</a>	<a href="#">WDRP2K</a>
<a href="#">DMHSRC15</a>	COB	Program source	Completed	405	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC15</a>	<a href="#">WDRP2K</a>
<a href="#">DMHSRC16</a>	COB	Program source	Completed	398	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC16</a>	<a href="#">WDRP2K</a>
<a href="#">DMHSRC17</a>	COB	Program source	Completed	236	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC17</a>	<a href="#">WDRP2K</a>
<a href="#">DMHSRC18</a>	COB	Program source	Completed	290	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC18</a>	<a href="#">WDRP2K</a>

Figure 2-15 Impact analysis details page (part 1)

http://wsaa.svl.ibm.com:9080 - Redpaper: DMHSRC06 statement range - Impact a...

File Edit View Favorites Tools Help

Site (1)	Description	HTTP IP address	FFS IP address
<a href="#">WDRP2K</a>	Current Site		

Batch job (5)	Analysis status	Source location	Site
<a href="#">JPHILDNP</a>	Completed	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL17</a>	<a href="#">WDRP2K</a>
<a href="#">QAJB0001</a>	Completed	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL31</a>	<a href="#">WDRP2K</a>
<a href="#">QAJB0002</a>	Completed	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL32</a>	<a href="#">WDRP2K</a>
<a href="#">Y2KMVSP</a>	Completed	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL40</a>	<a href="#">WDRP2K</a>
<a href="#">Y2KMVSR</a>	Completed	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHJCL37</a>	<a href="#">WDRP2K</a>

CICS online region (1)	Site
<a href="#">ATC3</a>	<a href="#">WDRP2K</a>

CICS transaction (2)	Run unit	CICS online region	CICS group
<a href="#">QAMD</a>	<a href="#">QAD04</a>	<a href="#">ATC3</a>	<a href="#">ATCGROUP</a>
<a href="#">QAMV</a>	<a href="#">QAD05</a>	<a href="#">ATC3</a>	<a href="#">ATCGROUP</a>

Run unit (7)	Analysis status	References	Site
<a href="#">IDCAMS</a>	Completed	36	<a href="#">WDRP2K</a>
<a href="#">QAD01</a>	Completed	1	<a href="#">WDRP2K</a>
<a href="#">QAD02</a>	Error	1	<a href="#">WDRP2K</a>
<a href="#">QAD02A</a>	Completed	0	<a href="#">WDRP2K</a>
<a href="#">QAD03</a>	Completed	2	<a href="#">WDRP2K</a>
<a href="#">QAD04</a>	Completed	1	<a href="#">WDRP2K</a>
<a href="#">QAD05</a>	Completed	1	<a href="#">WDRP2K</a>

Program (5)	Language	Analysis status	Number of lines in program	Source location	Site
<a href="#">QAD01</a>	COB	Completed	522	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC13</a>	<a href="#">WDRP2K</a>
<a href="#">QAD02</a>	COB	Completed	491	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC16</a>	<a href="#">WDRP2K</a>
<a href="#">QAD02A</a>	COB	Completed	498	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC15</a>	<a href="#">WDRP2K</a>
<a href="#">QAD04</a>	COB	Completed	478	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC17</a>	<a href="#">WDRP2K</a>
<a href="#">QAD05</a>	COB	Completed	605	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC18</a>	<a href="#">WDRP2K</a>

Data element (32)	Site	Program	Program source location	Decl. line	Level	Language	Type	Physical length	Logical length	Scale	Declared in copy?
<a href="#">ERROR-DATA</a>	<a href="#">WDRP2K</a>	<a href="#">QAD01</a>	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC13</a>	87	5	COB	NUMB	5	5		N
<a href="#">ERROR-REC</a>	<a href="#">WDRP2K</a>	<a href="#">QAD01</a>	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC13</a>	85	1	COB	GRP	35			N
<a href="#">INPUT-PART-NO</a>	<a href="#">WDRP2K</a>	<a href="#">QAD01</a>	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC13</a>	79	5	COB	NUMB	4	4		N
<a href="#">INPUT-REC</a>	<a href="#">WDRP2K</a>	<a href="#">QAD01</a>	<a href="#">C:\DMH\SAMPLE\SOURCE\DMHSRC13</a>	78	1	COB	GRP	7			N

Trusted sites

Figure 2-16 Impact analysis details page (part two)

http://wsaa.svl.ibm.com:9080 - Redpaper: DMHSRC06 statement range - Impact a...

File Edit View Favorites Tools Help

Component	Code	QAD	Source	Line	Lang	Type	Len	Pos	Impact	Site
MASTER-STK-PART-NO	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	31	5	COB	CHAR	4	4	Y
MASTER-STOCK-MAST-REC	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	30	1	COB	GRP	48		Y
UPD-STK-PART-NO	WDRP2K	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	31	5	COB	CHAR	4	4	Y
UPD-STK-PART-NO	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	31	5	COB	CHAR	4	4	Y
UPD-STOCK-MAST-REC	WDRP2K	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	30	1	COB	GRP	48		Y
UPD-STOCK-MAST-REC	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	30	1	COB	GRP	48		Y
WS-ERROR-DATA	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	148	5	COB	NUMB	5	5	N
WS-ERROR-REC	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	146	1	COB	GRP	35		N
WS-INPUT-PNO	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	111	5	COB	NUMB	4	4	N
WS-INPUT-REC	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	110	1	COB	GRP	7		N
WS-LOG-DATA	WDRP2K	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	108	5	COB	NUMB	5	5	N
WS-LOG-DATA	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	113	5	COB	NUMB	5	5	N
WS-LOG-REC	WDRP2K	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	106	1	COB	GRP	35		N
WS-LOG-REC	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	111	1	COB	GRP	35		N
WS-STK-PART-NO	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	31	5	COB	CHAR	4	4	Y
WS-STK-PART-NO	WDRP2K	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	31	5	COB	CHAR	4	4	Y
WS-STK-PART-NO	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	31	5	COB	CHAR	4	4	Y
WS-STK-PART-NO	WDRP2K	QAD04	C:\DMH\SAMPLE\SOURCE\DMHSRC17	31	5	COB	CHAR	4	4	Y
WS-STK-PART-NO	WDRP2K	QAD05	C:\DMH\SAMPLE\SOURCE\DMHSRC18	31	5	COB	CHAR	4	4	Y
WS-STOCK-MAST-REC	WDRP2K	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	30	1	COB	GRP	48		Y
WS-STOCK-MAST-REC	WDRP2K	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	30	1	COB	GRP	48		Y
WS-STOCK-MAST-REC	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	30	1	COB	GRP	48		Y
WS-STOCK-MAST-REC	WDRP2K	QAD04	C:\DMH\SAMPLE\SOURCE\DMHSRC17	30	1	COB	GRP	48		Y
WS-STOCK-MAST-REC	WDRP2K	QAD05	C:\DMH\SAMPLE\SOURCE\DMHSRC18	30	1	COB	GRP	48		Y
WS-TRANSACT-PNO	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	108	5	COB	NUMB	4	4	N
WS-TRANSACT-REC	WDRP2K	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	107	1	COB	GRP	7		N

Data set (2)		Site
LSTAU.SQA.INCOMING.STOCK(+0)		WDRP2K
LSTAU.SQA.VSAM.STOCK.MASTER		WDRP2K

Data store (10)	Type	Program	Source location	Site
ERR	FILE	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	WDRP2K
LOGFILE	FILE	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	WDRP2K
LOGFILE	FILE	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	WDRP2K
MASTER	FILE	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	WDRP2K
MASTER	FILE	QAD02	C:\DMH\SAMPLE\SOURCE\DMHSRC16	WDRP2K
MASTER	FILE	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	WDRP2K
STKMAST	CICSFILE	QAD04	C:\DMH\SAMPLE\SOURCE\DMHSRC17	WDRP2K
STKMAST	CICSFILE	QAD05	C:\DMH\SAMPLE\SOURCE\DMHSRC18	WDRP2K
TRANS	FILE	QAD01	C:\DMH\SAMPLE\SOURCE\DMHSRC13	WDRP2K
TRANSACT	FILE	QAD02A	C:\DMH\SAMPLE\SOURCE\DMHSRC15	WDRP2K

Mon Jan 08 13:52:32 PST 2007

Trusted sites

Figure 2-17 Impact analysis details page (part three)

Impact analysis automates the iterative investigative process that makes up much of an analyst's work, saving significant amounts of time while helping to improve the completeness of the analysis effort. Compared to a manual process, the automated approach with WebSphere Studio Asset Analyzer can deliver higher productivity for analysts, less rework in development and test, and fewer outages due to incomplete analysis.

## 2.7 Java application exploration and impact analysis scenario

Imagine that you find defect or a security problem in a particular .jar file, and you need to determine which applications use that .jar file. WebSphere Studio Asset Analyzer can help you do to this based on an analysis of the bytecode of your deployed applications. In this



scenario, let us assume that someone in your organization implemented BasicCalculator, and you need to find where it is used. First, you search for any known assets with a name starting with *BasicCalculator* as shown in Figure 2-18.

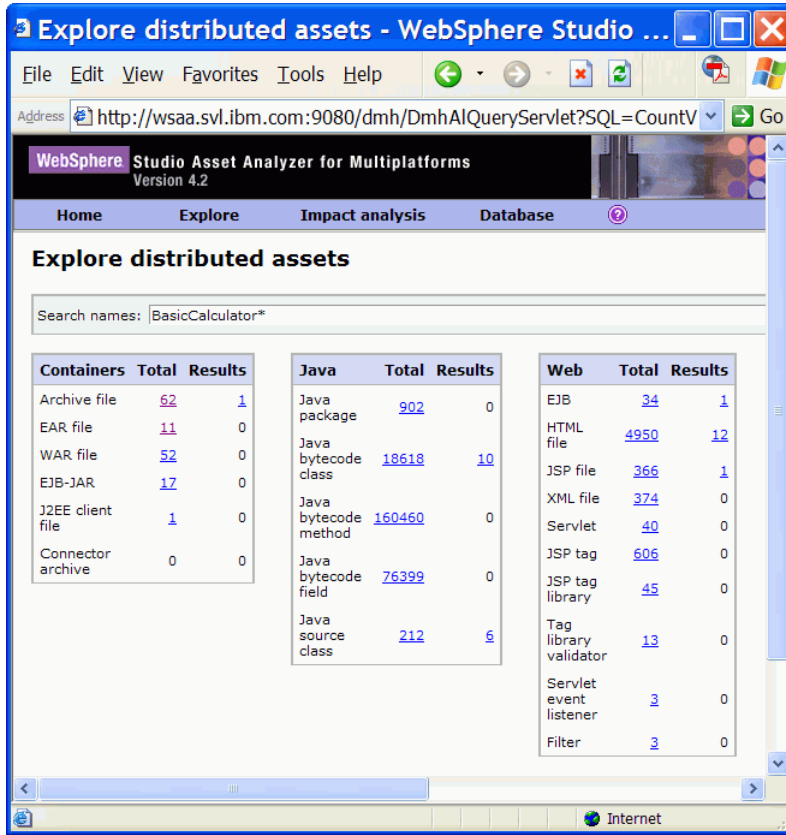


Figure 2-18 Only one EJB has a name matching this pattern

The EJB details page is helpful, as shown in Figure 2-19 and Figure 2-20. You see that there is a .jar file of the same name and that it is used in a client and a Web archive.

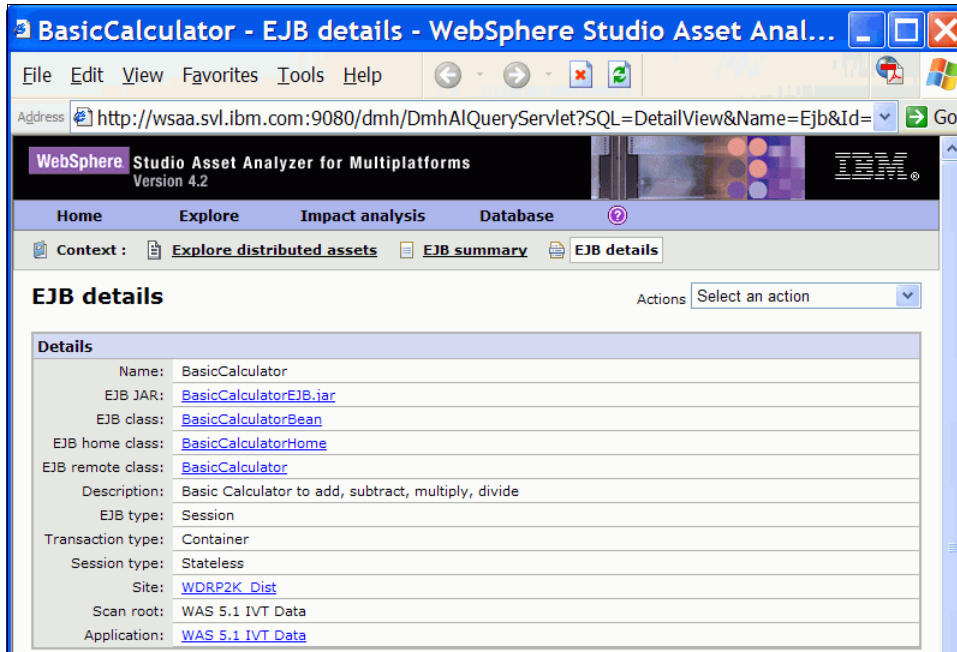


Figure 2-19 Some of the EJB details available

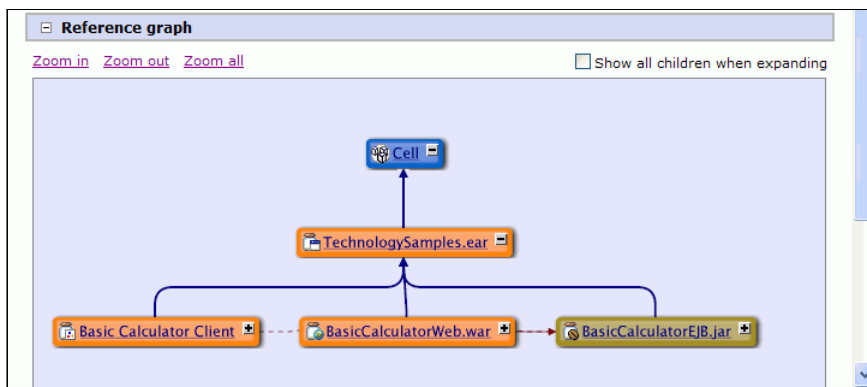


Figure 2-20 The .jar file that includes the BasicCalculator EJB is used in two places

Running an impact analysis on BasicCalculator EJB, as shown in Figure 2-21, provides more detail on the dependencies, and it does not find any indirect impacts.

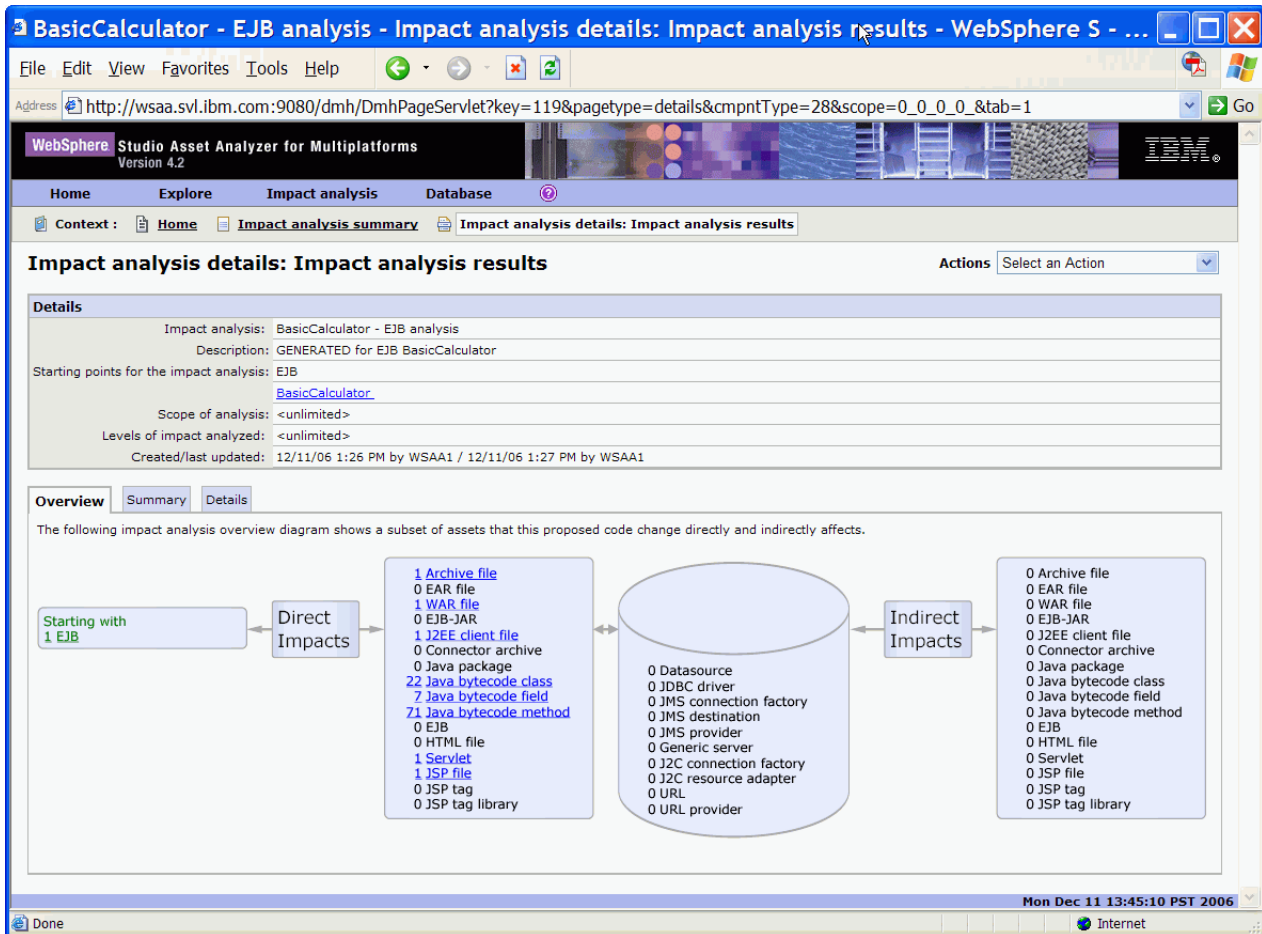


Figure 2-21 Impact analysis results for an EJB

## 2.8 Limitations of static analysis

WebSphere Studio Asset Analyzer metadata is limited to what can be discovered by *static analysis* of the program source and runtime subsystem configuration (and for Java applications, the bytecode). So if any application behavior is dynamic (for example determined at runtime), WebSphere Studio Asset Analyzer will not be able to discover the related program control transfers and data accesses—a limitation of all static analysis tools.

Examples of dynamic application behavior include:

- ▶ The name of a program target of a dynamic call is obtained by the calling program by reading it from a file record.
- ▶ CICS user exits (GLUEs or TRUEs) or User Replaceable Modules determine which CICS region, program, data set, or queue to use.
- ▶ A Java application uses reflection or runtime binding to determine what data source to use.

Dynamically-determined dependencies do need to be considered when making and testing application changes. Using WebSphere Studio Asset Analyzer, you can assert these

relationships manually between software assets. These are called *user-defined relationships*. Additionally, you can use tools such CICS Interdependency Analyzer to capture a record of application behavior. We discuss this tool in Chapter 5, “Additional value through extension and integration with other tools and metadata repositories” on page 51.

## 2.9 Answering your own questions with custom queries

The user interface to WebSphere Studio Asset Analyzer provides a framework to ask many of the common types of questions one might ask about applications. However, there are many other ways that the metadata in WebSphere Studio Asset Analyzer can be combined to answer interesting questions.

To simplify that task, WebSphere Studio Asset Analyzer includes a feature called *Custom Queries*. These Custom Queries are sharable SQL queries that produce output in the form of tables directly in the Web browser. You define, use, and share Custom Queries right in the WebSphere Studio Asset Analyzer user interface. Then, you can import or export the Custom Queries as XML files, and they can be attached to the Action drop-down menu on the summary and details pages. (Figure 3-4 on page 40 shows an example of a Custom Query.)



## Approaches to discovering code for reuse

As mentioned in Chapter 1, “The promise and challenges of reuse” on page 1, in this paper we focus on the *Reuse* entry point to SOA. This chapter discusses different approaches to discover the code for reuse. It assumes that you know what kinds of services you want to create.

## 3.1 Approaches to discovery

Perhaps your organization has already completed a business function decomposition analysis similar to the Component Business Modeling methodology developed by IBM Global Business Services. You might have started with People, Process, or Information as your SOA entry point. You have requirements at hand for specific services, and now it is time to determine which of these entry point can be provided by your current application, as shown in Figure 3-1. Alternatively, you might be working in a tactical mode and have identified a valuable *infrastructure service* such as creating the information about a new customer.

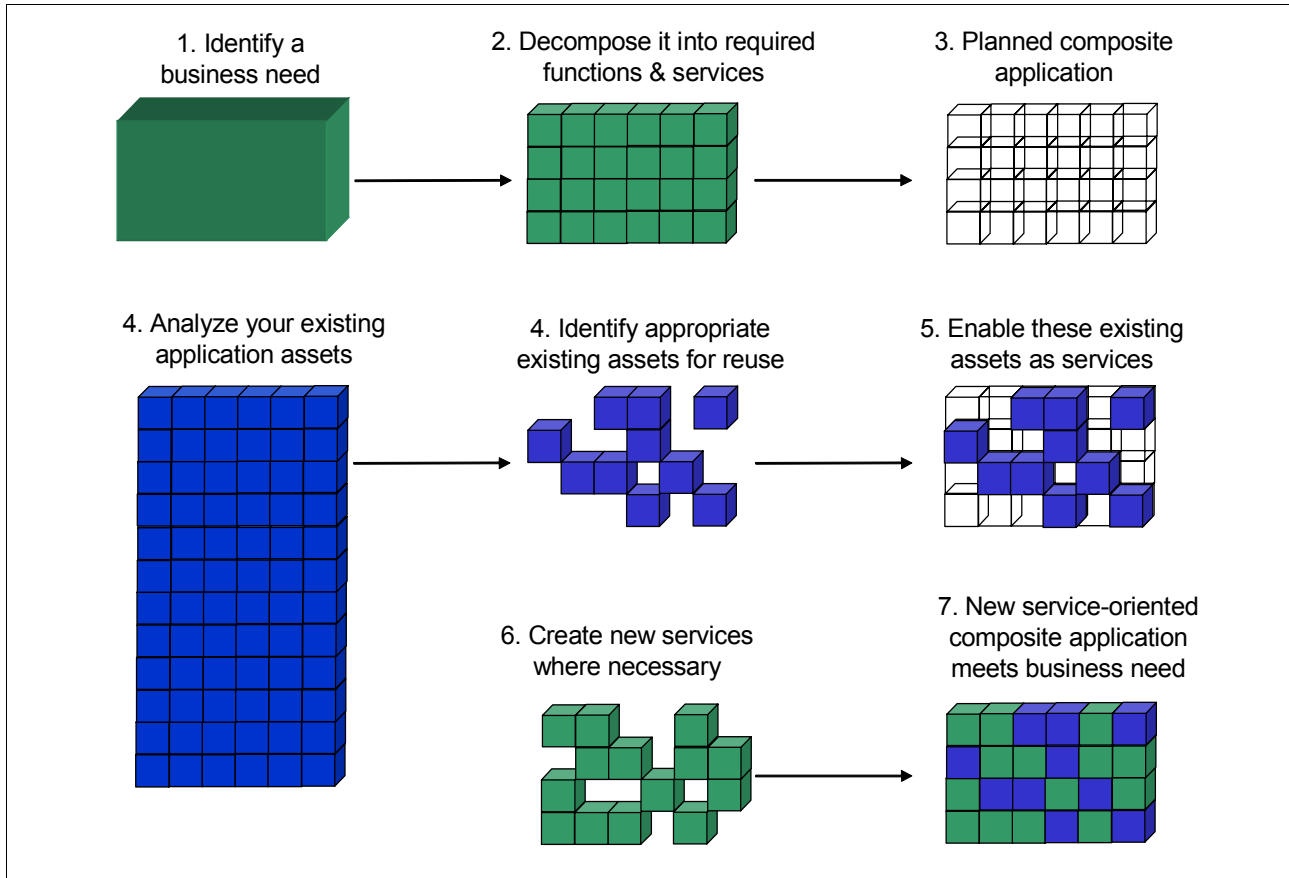


Figure 3-1 A composite application consisting of a combination of reused application assets and new services

Imagine that your department has just completed the implementation of WebSphere Studio Asset Analyzer, and you are ready to use it to help you with this task. Where do you begin?

### 3.1.1 Starting with data

You could start with interesting data. One way to think of a business process is to consider it a set of states of the associated data together with the transformations of this data. If you know some of the data that is created by or consumed in the business process, then you can use WebSphere Studio Asset Analyzer to help you find the related set of programs and other data stores.

Perhaps you know the name of a data set or DB2 table that contains the data that your Web service needs to read or update. In this case, you can start by finding the programs that access this data source, as shown in Figure 3-2. You can then run an impact analysis on the data set or DB2 table, the record or a related data element to find other programs, JCL, and other data elements and data stores through which this data flows. (We used the copybook that defines the record layout in the scenario that we looked at in Chapter 2, “Using WebSphere Studio Asset Analyzer to cut through the complexity of application changes” on page 9.)

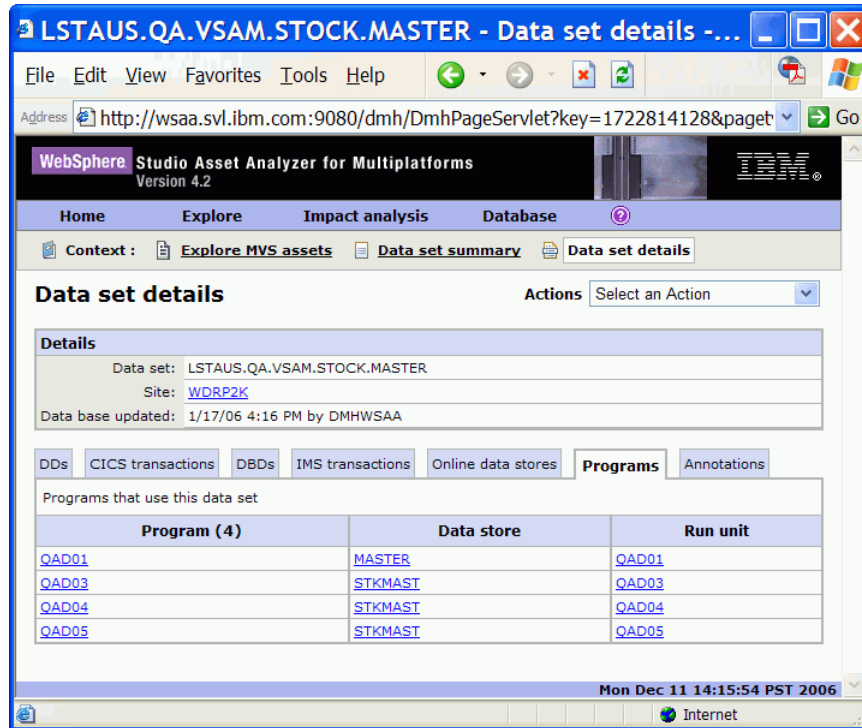


Figure 3-2 Programs accessing a particular data set

Another alternative scenario is that perhaps you know only that your Web service needs to deal with employee number, part number, order number, and so forth. You can construct searches of the data elements that are defined in your programs using the various likely permutations of the name. You can then run an impact analysis on one or more of these elements and find other data elements into which values are moved and the programs that contain these data elements and other data stores.

### 3.1.2 Starting with programs

You can start with programs or screens that you know are part of a business function that you want to reuse in your Web service.

#### Screens

*Screens* are a natural starting point, and user application’s manuals are good places to find mappings of screen-based activities and the business processes that they fulfill. You can think of a screen—a basic mapping support (BMS) map for Customer Information Control System (CICS), a Memory File System (MFS) screen for Information Management System (IMS)—as a user’s view of some point in a business process. Behind the screen are a set of programs, data movements, and state changes that embody the business function.

In general, screen-based mainframe applications might be easier to reuse than the highly object-oriented applications that have been developed over the last decade or two. For more information, see the online article at:

<http://esj.com/enterprise/article.aspx?EditorialsID=1457>

When you have identified the screens of interest, you can decide to wrap the existing screen flows in a Web service. This approach has the advantage of requiring no changes to the existing application, so it is fast to implement and test. However, it does have disadvantages, including:

- ▶ A new interface that you must maintain. Changes to the screen require changes to the wrapper.
- ▶ A longer code path, which does not optimize performance.

The IBM Host Access Transformation Services (HATS) does real-time transformations of 3270 and 5250 screens and screen flows into Web pages, Web page fragments, and Web services. The macro facility in HATS makes it possible to record a set of screen flows as illustrated in Figure 3-3. For more information HATS, see:

<http://www.ibm.com/software/webservers/hats/index.html>

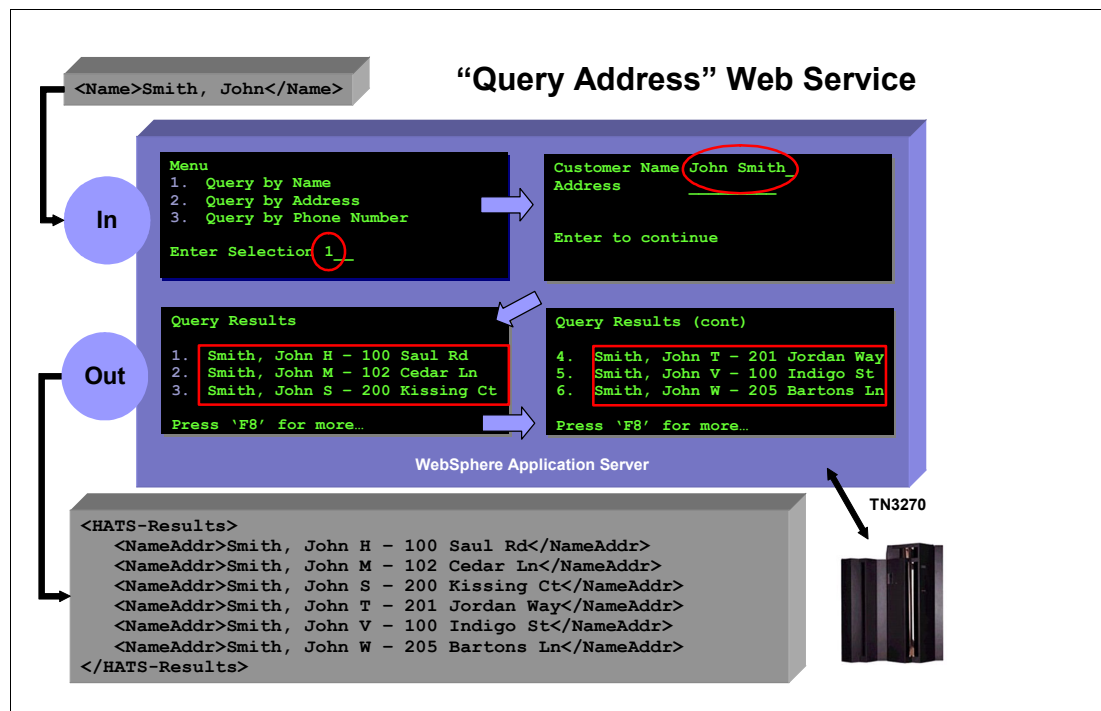


Figure 3-3 A HATS macro

### Micro flows to solve interface mismatch

One of the big challenges in creating business-level services out of existing application assets is that the designers of the original online applications made the completely rational assumption that a person would be present to navigate through the screens to accomplish the business task. Multiple user interactions are often required to accomplish the business task.

However, with a service, you typically want a single request and response, so your client calling the service does not need to know how to navigate through the internal implementation of your service. How can you map the fine-grain interfaces of the existing application to the coarse-grain interface that is required for your service? HATS addresses this issue by providing a macro recording and composition engine. A HATS macro can drive



various screens and can follow various paths through the screens based on the content of the screens.

A related and relatively new approach is to compose micro flows between screen-based programs visually. IBM offers the *Service Flow Modeler* as part of WebSphere Developer for System z™, in which you can visually compose micro flows (and longer flows) and run them as a HATS macro in the WebSphere Application Server. Alternatively, if you are deploying to CICS, you can combine screen-based and COMMAREA-based programs in the same micro flow and benefit from CICS qualities of service within and between CICS containers. For more information about Service Flow Modeler, see:

<http://ibm.com/software/awdtools/devzseries/>

## Programs

You might want to avoid screen I/O and call in to a program directly (for example, using a Java Connector provided in the CICS Transaction Gateway to issue Java method calls within a WebSphere application, where the presentation layer of the application exists). The Java connector translates the Java call into a native call in the Enterprise Information System (EIS), CICS in this case. The existing CICS programs are invoked through native CICS services, typically Distributed Program Link (DPL), with data passed in a communication area (COMMAREA). Alternatively, you could use the native Web service facilities in CICS Transaction Server V3 and make CICS the Web service provider, which many CICS shops are doing. (We think this is driving a big portion of the fast take-up of CICS TS V3 compared to prior CICS releases.)

Advantages of this approach include:

- ▶ Avoiding the de facto creation of a new programming interface (screen definitions) as is the case when using HATS or similar tools.
- ▶ Better scalability, because there is a shorter code path for the response.

Potential disadvantages include:

- ▶ Possible need for programmatic remediations to ensure proper data validation.
- ▶ Possible need to disentangle business logic from screen flows.

## Run units as expressions of business processes

Programs participate in various call sequences. These are called *run units* in WebSphere Studio Asset Analyzer. Examples of run units include:

- ▶ A CICS transaction consisting of a sequence of CICS programs
- ▶ A batch job consisting of various job steps

Software assets that are associated with run units (and that are visible in the run unit diagram) include JCL, screens, transactions, programs, and their data stores.

Run units often embody and enable one or more business functions. For example, a run unit might include CICS screens for order entry, which return data to programs that validate that data, which call other programs that embody the business logic and interact with the data stores.

This characteristic makes the run unit a useful construct when you are considering reuse of a business task or process. When you find any candidate element in the run unit, you can quickly get a look at that element in context of its related elements. (Figure 2-7 on page 19 shows a run unit diagram.)

WebSphere Studio Asset Analyzer also identifies assets that do not appear to have a means of invocation. These assets can be sets of JCL, programs, screens, and data that are no longer used in your organization. If, after further examination, you determine that these assets are obsolete, you can remove them from your inventory and, thereby, reduce future non-productive analysis efforts.

## Application architecture

Just as a woodworker considers the grain of a piece of wood before deciding what to do with the wood, analysts and architects can look at the “grain” of an application, that is the structure, grouping, and so forth of the programs and data, to get an idea of what might be easy to reuse and what might be more difficult.

Modern application design best practices separate the presentation (view) from the business logic (controller) and the structure of the data (model). This *model-view-controller* (MVC) approach facilitates reuse at any of the integration points between these concerns. However, what if your enterprise application was developed before MVC approaches were common? It is not unusual to find old COBOL programs that mix two or all three types of access, particularly programs that are tens of thousands of lines long. It is a challenge to identify and separate these concerns after the fact within practical cost and time parameters.

WebSphere Studio Asset Analyzer can help you to find programs that:

- ▶ Are doing screen or data I/O or both
- ▶ Appear to be the hub programs
- ▶ Are the largest, most complex, or most invoked by other programs

The advanced search allows searching on interesting attributes as well as asset name patterns. In the case of programs, you can, for example, select CICS programs that do or do not do terminal I/O.

Extending this idea, you can combine these attributes in custom queries to zero in on just the programs in which you are interested. Figure 3-4 shows a list of such custom queries, and Figure 3-5 shows the results of one of these queries.

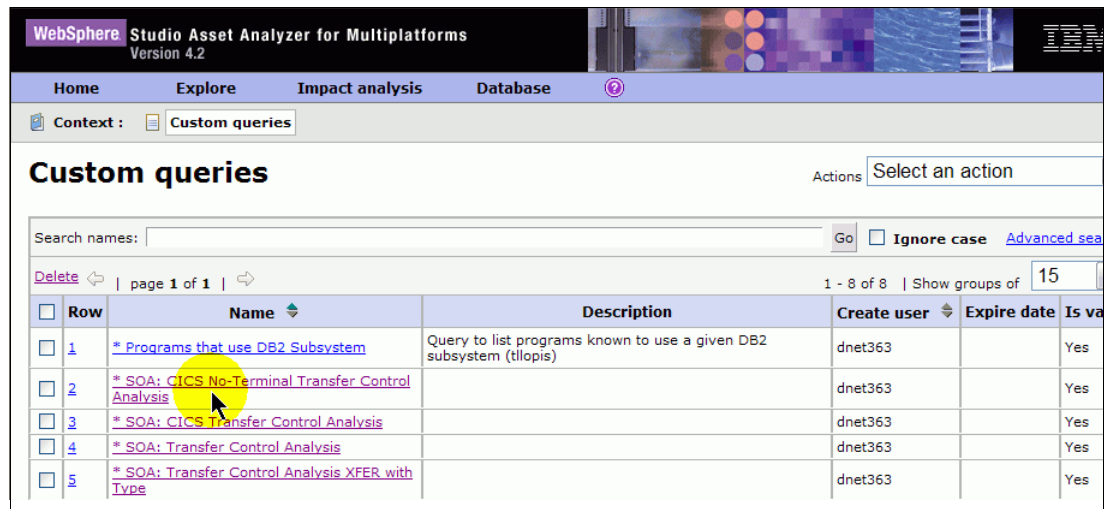


Figure 3-4 Custom queries developed to help find programs as candidates for reuse as services

Note that a program called by many other programs might be acting as a service within the application already. For example, in one proof-of-concept engagement, the client provided about 400 CICS and IMS COBOL programs. The custom query shown in Figure 3-5 found quickly the programs that are most frequently called by other programs. (In this case, frequency is a measure of the number of transfers of control found in the metadata.) One commonly-invoked program accomplished the task of address postal code (ZIP code) validation. It had a well-defined interface and could be easily made available as a service (if that were of value in the larger scheme of things).

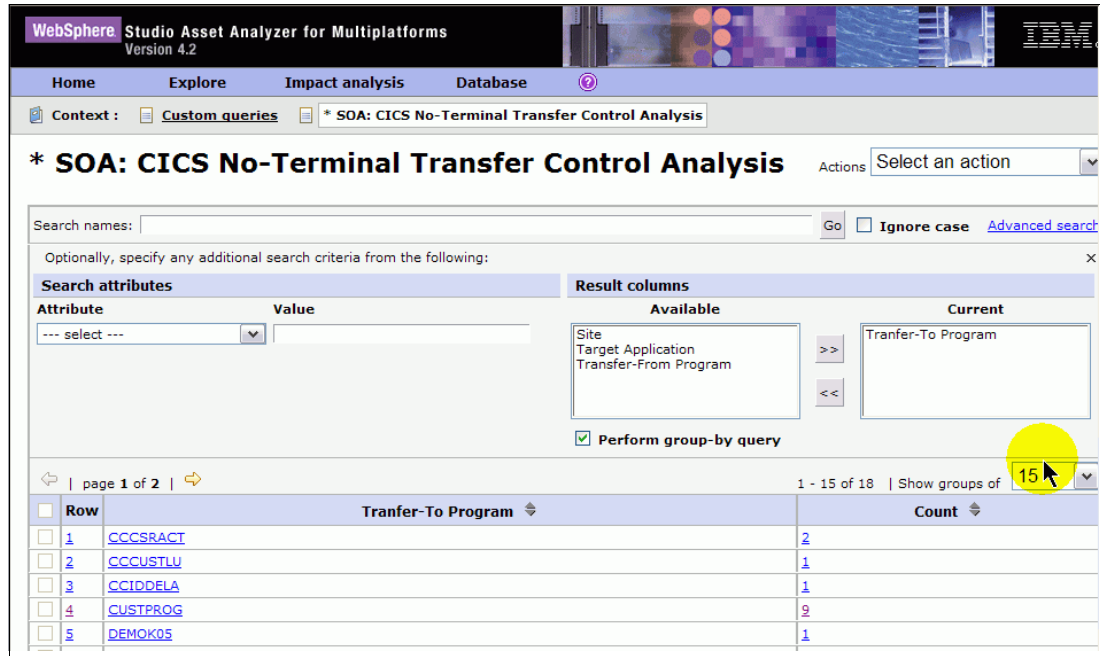


Figure 3-5 Sample output from this custom query

## Run unit remediation

Whether starting with data or programs, an analyst or developer can use WebSphere Studio Asset Analyzer to review the run unit to determine what modifications (if any) are required before reusing the run unit. Additional work required can include:

- ▶ Identifying data validation assumptions from screen interactions, and ensuring that the data validation is done somewhere, potentially as part of the contract with the calling service.
- ▶ Separating the business logic that you want to reuse from what is not relevant.
- ▶ Finding business rules that span a single source module and grouping them in a way that enables reuse.
- ▶ Removing screen calls down in a call chain and replacing this with error return codes.

WebSphere Studio Asset Analyzer provides information to the analyst or developer to enable them to do these remediations manually. The WebSphere Studio Asset Analyzer companion product, the *IBM Asset Transformation Workbench*, can help automate many of these activities. See Chapter 5, "Additional value through extension and integration with other tools and metadata repositories" on page 51 for more information.

In addition, there is the need for the producer and the consumer of a service to be explicit in ensuring that both share an understanding of the meaning of the service's data elements. This is by no means trivial, and this requirement is behind much of the recent growth in interest in metadata about data. For example, IBM recently announced the IBM Information

Server as a set of tools using common metadata for managing data across the application life cycle. For more information about the IBM Information Server, see:

[http://ibm.com/software/data/integration/info\\_server/](http://ibm.com/software/data/integration/info_server/)

## 3.2 Technical challenges to reuse: Change complexity

This section discusses briefly some technical factors that affect the degree of difficulty in reusing applications. For a more detailed look at these challenges, see the online article at:

<http://www.research.ibm.com/journal/sj/441/hess.html>

### 3.2.1 Intellectual control

Some applications in your organization are probably better understood than others, possibly because you have a team that has spent time in them recently, because of the quality of your documentation, or because of the application's size and complexity. When you do not have sufficient knowledge about an application, the risk and cost of transforming it is higher. The knowledge gap can involve very basic information, including:

- ▶ What programs make up your application
- ▶ What data these programs use
- ▶ What interfaces are available today
- ▶ Where the business logic (rules) reside

### 3.2.2 Interfaces

Current interfaces might be at the “wrong” level of abstraction. You need *business level* interfaces, preferably adhering to the single request/response pattern.

### 3.2.3 Intermingled business logic

The business logic that you want to use can be spread throughout a number of programs and intermingled with screen or data I/O and related data validation, which might make it more difficult to isolate for reuse.

### 3.2.4 Application pattern

Your application's pattern might not be optimal for the kind of reuse that you want to do. In particular, you can have business tasks handled in batch mode, which complicates the effort to offer them as synchronous services. For example:

- ▶ Your application writes transactions to a journal, and the journal is then processed in batch mode at a later time, typically at night.
- ▶ Multiple business tasks are done in one job step as an optimization of I/O processing.
- ▶ Your batch application assumes data transformations in a specific sequence, that is A and B must run before C, but you only want to reuse C in your service.
- ▶ Data might not be immediately accessible, for example if the application uses sequential files, your program must read the file from the beginning.

The good news here is that the basic rationale for WebSphere Studio Asset Analyzer and other tools such as Asset Transformation Workbench is to help address these challenges. By automating portions of the analysis and transformation process, the cost of the exercise can

drop to the point where it becomes feasible to modernize and transform applications in ways that are not feasible when done manually.

### **3.3 Non-technical challenges to reuse**

As with many facets of the IT business, one of the biggest challenges is non-technical.

As an example, one of the largest enterprises in the world is the U.S. Department of Defense. The Association for Enterprise Integration issued an interesting set of recommendations to the U.S. Government that included technical and non-technical constraints to reuse in services. For information about this report, “Facilitating Shared Services in the DoD: A Report of the Data Sharing and Services Strategy Working Group,” see:

[http://www.afei.org/documents/DS3SharedServicesPaperFinalVersion021206c\\_000.pdf](http://www.afei.org/documents/DS3SharedServicesPaperFinalVersion021206c_000.pdf)





## Accelerate test case generation with WebSphere Studio Asset Analyzer

All development and maintenance changes bring with them the need to test. The effort to test a change can easily exceed the effort required to make it. WebSphere Studio Asset Analyzer can assist with the planning and preparation of this testing by:

- ▶ Identifying test cases.
- ▶ Identifying the resources required to conduct the test.
- ▶ Feeding an automated test bed.

This chapter discusses how to identify test cases.

## 4.1 Identifying test cases

To test a change to a program or copybook it is important to identify all the places it is used. WebSphere Studio Asset Analyzer provides the “where used” capability that allows an analyst to identify quickly a set of transactions or batch jobs that invoke a changed program. These jobs or transactions become candidate test scripts.

For a subroutine, it can be difficult to find all the places where it is invoked. A Subject Matter Expert (SME) might be aware of the invocations within a single application, but the subroutine might be used by other areas as well, sometimes without the “owner” even knowing it. WebSphere Studio Asset Analyzer provides a list of the known places from where a program is called, allowing testing to be more complete.

### 4.1.1 Identifying the resources required to conduct the test

Testing a changed component or components requires setting up a test environment. WebSphere Studio Asset Analyzer assists here by identifying the resources that are needed to execute the tests. For example, for a given batch job the Batch Job Details screen serves as a starting point. From here, the test planner can identify the resources that are required to execute the job:

- ▶ The Steps tab shows the run units the job requires.
- ▶ The DDs tab shows the libraries, data sets, and SYSOUT classes that the job uses. Some of the data set names need to be changed to reflect the use of test rather than the production data. The test planner also looks for data sets that are allocated as *OLD*, *MOD*, or *SHR* the first time that they are used. These data sets must exist for the job to run successfully. If they are used for input they can be left pointing at production data. If test data is used, it requires a name change and has to be populated with the data prior to testing.
- ▶ The Included Files tab shows the names of PROCs that the job uses. This is commonly where the changes to data set names are made.

### 4.1.2 Feeding an automated test bed

If a change affects a large number of programs, the testing can involve dozens or even hundreds of jobs. Manually identifying the resources and modifying the PROCs is time consuming. Such projects are better tested using a test bed generator such as the Test Environment Builder (TEB) from IBM Global Services.

Taking advantage of TEB, however, still requires the test planner to identify the resources that are required to populate each test environment. For a large environment with many jobs, it makes more sense to pull the information from WebSphere Studio Asset Analyzer using SQL-based queries. IBM Global Services TEB offering includes the ability to integrate with WebSphere Studio Asset Analyzer for just that reason.

To identify the resources for the test environment, the test planner identifies a series of jobs or transactions. These jobs need to be processed in the same order that they normally run so that you can differentiate the data set that is created by the job stream from the data sets that need to exist prior to the start.

The examples in the next section are taken from a REXX™ program that reads a list of batch job names and then issues queries that identify the resources that are required. The queries for a set of transactions would be similar.



The WebSphere Studio Asset Analyzer tables are keyed off of ID columns. The information about a job is retrieved using a BATCH\_JOB\_ID. For a program (or compilation unit in WebSphere Studio Asset Analyzer terms), a COMP\_UNIT\_ID is the key. The examples in the next section look up the Batch\_Job\_ID based on the name and then issue queries using that ID rather than joining the table that contains the name each time. Note that DB2 tables, columns, and their meanings are documented in the online help.

## Example job queries

The queries in this section demonstrate how to retrieve the resources for a job given its name. They are repeated for each job in the list. Ideally, the program that retrieves the information keeps track of the resources already identified and screens out duplicates prior to reporting them.

Example 4-1 finds the Batch\_Job\_ID given the job name (passed as the variable JOBPARM).

*Example 4-1 Find the Batch\_Job\_ID (passed as the variable JOBPARM)*

---

```
"SELECT BATCH_JOB_ID FROM DMH.DMH_BATCH_JOB WHERE BATCH_JOB_NAME = '"JOBPARM"'".
```

---

**Note:** The BATCH\_JOB\_ID is passed to subsequent queries.

Example 4-2 finds all the data sets for the batch job given the Batch\_Job\_ID (passed as variable PARMID).

*Example 4-2 Find the Batch\_Job\_ID (passed as variable PARMID)*

---

```
"SELECT T1.DD_NAME,T2.DATASET_NAME,T1.STATUS,T1.DISPOSITION
FROM DMH.DMH_DD_CONCAT T1, DMH.DMH_DATASET T2
WHERE T1.DATASET_ID = T2.DATASET_ID AND T1.BATCH_JOB_ID = "PARMID"
ORDER BY T1.JOB_STEP_SEQ"
```

---

**Note:** If the STATUS field is NEW for the first time a data set is allocated across all the jobs in the list, then the data set is created by the job stream and does not need to be copied into the test environment. You might want to group the data sets based on their DD names. It might be helpful to identify STEPLIBS, SYSIN control cards, and so forth.

If the data set name has parentheses and a numeric value between them, you can classify it as a Generation Data Group (GDG). Determining which GDG generations need to be present (versus only the base entry) is not as simple as evaluating the first STATUS field and can vary from system to system.

Also, if a job allocates the (-3) generation, you also need to copy the (-2) and (-1) unless you are 100% sure the test job will only run once.

Example 4-3 finds all the PROCs used by a job (passed as variable MAINPARM).

*Example 4-3 Find all the PROCs used by a job (passed as variable MAINPARM)*

---

```
"SELECT MEMBER_NAME
FROM DMH.DMH_MEM_INCLUDE
WHERE MEMBER_ID = "MAINPARM"
```

---

Example 4-4 finds the IMS databases (DBDs) that are used.

*Example 4-4 Find the IMS databases (DBDs) that are used*

---

```
"SELECT T1.FILE_NAME, T2.DATASET_NAME
FROM DMH.DMH_ONLINE_DSTORE T1, DMH.DMH_DATASET T2, DMH.DMH_IMS_DBD_DD T3,
DMH.DMH_IMS_DBD T4
WHERE T1.DATASET_ID = T2.DATASET_ID AND T1.FILE_NAME = T3.DDNAME AND T3.DBD_ID =
T4.DBD_ID AND T3.DBD_ID = "INID"
```

---

Example 4-5 finds all the IMS usage.

*Example 4-5 Find all the IMS usage*

---

```
"SELECT T2.PSB_NAME, T4.PCBNAME, T4.DBDNAME, T5.DBD_TYPE, T5.DBD_ID, T4.PROCOPT,
T4.PCB_TYPE
FROM DMH.DMH_BATCH_JOB T1, DMH.DMH_INVOKE_RU T2, DMH.DMH_IMS_PSB T3,
DMH.DMH_IMS_PCB T4, DMH.DMH_IMS_DBD T5
WHERE T2.BATCH_JOB_ID = "PARMID" AND T1.BATCH_JOB_ID = T2.BATCH_JOB_ID AND
T2.PSB_NAME = T3.PSBNAME AND T3.PSB_ID = T4.PSB_ID AND T4.DBDNAME = T5.DBDNAME"
```

---

Example 4-6 finds all the DB2 plans.

*Example 4-6 Find all the DB2 plans*

---

```
"SELECT DISTINCT PLAN_NAME
FROM DMH.DMH_INVOKE_RU
WHERE PLAN_NAME > " AND BATCH_JOB_ID= " PARMID"
```

---

Example 4-7 find all the Programs.

*Example 4-7 Finds all the Programs*

---

```
"SELECT T2.TRANSFER_TARGET, T2.TARGET_CU_ID, T2.LEVEL
FROM DMH.DMH_INVOKE_RU T1, DMH.DMH_RUN_UNIT_CNTNT T2
WHERE T1.RUN_UNIT_ID = T2.RUN_UNIT_ID AND T1.BATCH_JOB_ID= " PARMID"
```

---

**Note:** The level can be used to identify subroutines versus main programs (level < 2).

For each program identified in query 7, The TARGET\_CU\_ID is the compilation unit ID that is passed to subsequent queries.

## Example program queries

The examples in this section show how to identify program queries.

Example 4-8 identifies copybooks (program ID is passed as variable ICUID).

*Example 4-8 Identify copybooks (program ID is passed as variable ICUID)*

---

```
"SELECT MEMBER_NAME
FROM DMH.DMH_MEM_INCLUDE
WHERE MEMBER_ID = "ICUID"
```

---

Example 4-9 identifies DB2 Tables (program ID is passed as variable ICUID).


*Example 4-9 Identify DB2 Tables (program ID is passed as variable ICUID)*

---

```
"SELECT T2.SYMBOL_TEXT  
FROM DMH.DMH_SQL_TABLE T1, DMH.DMH_SYMBOL T2  
WHERE T1.COMP_UNIT_ID = "ICUID " AND T1.FULLNAME_SYM_ID = T2.SYMBOL_ID"
```

---





## **Additional value through extension and integration with other tools and metadata repositories**

The open architecture of WebSphere Studio Asset Analyzer makes it possible to extend the metadata as well as integrate with other tools. This chapter discuss some pointers when using WebSphere Studio Asset Analyzer with other tools and metadata repositories.

## 5.1 Benefits of an open architecture

This section presents the benefits of an open architecture.

### 5.1.1 Extending the metadata (tagging)

Within the WebSphere Studio Asset Analyzer user interface, you can annotate assets with free-form text, including adding new categories of annotations, and then search for relevant assets based on their content. The *taxonomy* (called a *folksonomy* in Web 2.0 lingo) can be as formal or informal as you want to make it.

Annotation text can be used in many ways:

- ▶ Categorizing (or *tagging*) assets
- ▶ Adding owner names and contact information
- ▶ Using hyperlinks to point to related information or assets in other metadata repositories

For example, some Japanese customers include the Japanese names of their assets in annotation text. One customer uses annotation text to flag JCL, programs, and related copybooks that are candidates for decommissioning after correlating WebSphere Studio Asset Analyzer and runs logs to determine which have not run since a specified date.

### 5.1.2 Using Web service

The WebSphere Studio Asset Analyzer Web service API is a good way to extract information about applications for use in independent analysis and other tools and metadata repositories. It takes just a couple of hours for a developer who is familiar with Web services and a tool such as the Rational® Application Developer or WebSphere Developer for System z (now known as *Rational Developer for System z* in Version 7.1), which takes a WSDL file to generate client-side stub classes or programs. Such a developer can start building an application quickly that consumes WebSphere Studio Asset Analyzer metadata (or adds to selected types of metadata). For more information about the WebSphere Studio Asset Analyzer Web service interface, see:

<http://ibm.com/software/awdtools/wsaa/library/>

As an example of client using this interface, you can look at WebSphere Developer for zSeries® V6 WebSphere Developer Asset Analyzer technical preview (or better yet, the updated and much improved version for Rational Developer for System z V7.1, which will probably be available by the time this paper is published).

WebSphere Developer Asset Analyzer provides COBOL and PL/I program analysis scoped to the local project (and with hooks into the enterprise-wide instance of WebSphere Studio Asset Analyzer, if it exists). This technical preview uses the Web service API to communicate between the analysis engine and WebSphere Developer for System z itself.

For additional information about Rational Developer for System z technical preview of WebSphere Developer Asset Analyzer, see:

<http://ibm.com/software/awdtools/devzseries/support>

The Web Services interface to WebSphere Studio Asset Analyzer is the preferred and stable way to interact with the product programmatically.

### 5.1.3 Using the open metrics framework

WebSphere Studio Asset Analyzer V5.1 uses an open metrics framework, which makes it possible to get source from your own metrics engine to calculate what you want. It uses the Web service interface to make source code available to your metrics engine (if you want to get it from WebSphere Studio Asset Analyzer) and provides a means to return the resulting numerical values to the WebSphere Studio Asset Analyzer repository and show these values in the metrics tables in the WebSphere Studio Asset Analyzer user interface.

**Note:** A metric can be any numerical value - not just code metrics.

### 5.1.4 Using your favorite means of issuing SQL queries

*Custom Queries*, which we discussed in Chapter 3, “Approaches to discovering code for reuse” on page 35, are a means of constructing, executing, and sharing SQL queries and their results from within the user interface of WebSphere Studio Asset Analyzer. However, it is also possible to query the repository from outside of WebSphere Studio Asset Analyzer. This is one of the advantages of having the metadata in DB2. You can use QMF™, SPUFI, a DB2 client, or any other means of issuing SQL queries to generate results you can use. For example, some customer shops create batch reports, which then get e-mailed to interested parties using their own batch processes.

You can get a fast start on writing queries against the WebSphere Studio Asset Analyzer database by looking at the queries used in the WebSphere Studio Asset Analyzer user interface. You can append &SHOWSQL=1 to any URL, refresh the Web page, and then the redrawn page displays the SQL along with the associated results.

Example 5-1 finds copybooks that are no longer used. The query creates delete control cards for copybooks that are not referenced anywhere. The output of the query can be fed into &HLQ.SDMHCNTL(DMHJCLBU), which is part of WebSphere Studio Asset Analyzer.

*Example 5-1 Finding copybooks that are no longer used*

---

```
SELECT 'D '
      , T4.RESOUC_TYPE
      , T5.NAME
      , SUBSTR(T3.CONTAINER_NAME, 1, 44)
      , SUBSTR(T1.FILE_NAME, 1, 10)
FROM DMH.DMH_FILE          T1
   , DMH.DMH_FILE_TO_CONT T2
   , DMH.DMH_CONTAINER    T3
   , DMH.DMH_RESOURCE_MNGR T4
   , DMH.DMH_SITE         T5
WHERE T1.LANGUAGE_CD      = 'COB'
      AND T1.FILE_ID       = T2.FILE_ID
      AND T2.CONTAINER_ID  = T3.CONTAINER_ID
      AND T3.RESOURCE_MNGR_ID = T4.RESOURCE_MNGR_ID
      AND T3.SITE_ID       = T5.SITE_ID
      AND T1.FILE_TYPE_CD  = 'INCL'
      AND T1.FILE_ID NOT IN (SELECT MEMBER_ID
                             FROM DMH.DMH_CU_INCLUDE)
ORDER BY 2, 3, 4, 5
```

---

Example 5-2 finds all inserts of IMS segments.

*Example 5-2 Finding all inserts of IMS segments*

---

```
SELECT
  T2.DATA_STORE_NAME,
  T1.COMP_UNIT_NAME,
  T1.COMP_UNIT_ID,
  T3.INSTRUCTION,
  T3.DATA_STORE_ID
FROM
  DMH.DMH_COMPILE_UNIT T1,
  DMH.DMH_DATA_STORE T2,
  DMH.DMH_DATA_RECORD T3
WHERE T1.COMP_UNIT_ID = T2.COMP_UNIT_ID
AND T2.COMP_UNIT_ID = T3.COMP_UNIT_ID
AND T2.DATA_STORE_ID = T3.DATA_STORE_ID
AND DATA_STORE_TYPE_CD = 'IMSSEGMT'
AND INSTRUCTION = 'ISRT'
```

---

Example 5-3 pulls a list of job names from WebSphere Studio Asset Analyzer into an ISPF edit session.

**Note:** JOBSTM is passed as the pattern for job names. Passing ABC% pulls all job names starting with ABC.

ADDR0W is the variable holding the line number in the file where the line is to be inserted.

Error checking is abbreviated as "...".

*Example 5-3 Pulling a list of job names from WebSphere Studio Asset Analyzer*

---

```
GETJOBS:
  ARG JOBSTM
  ADDR0W = ROW
  "ISREDIT LINE_BEFORE "ADDR0W" = INFOLINE "JOBSTM"'"
  PARMSQL2="SELECT BATCH_JOB_NAME FROM DMH.DMH_BATCH_JOB",
           "WHERE BATCH_JOB_NAME LIKE "JOBSTM"'"',
           "ORDER BY BATCH_JOB_NAME"
  ADDRESS DSNREXX EXECSQL "DECLARE C2 CURSOR FOR S2"
...
  ADDRESS DSNREXX EXECSQL "PREPARE S2 FROM :PARMSQL2"
...
  ADDRESS DSNREXX EXECSQL "OPEN C2"
...
  ENDOFC2 = N
  DO WHILE (ENDOF C2 <> Y)
    ADDRESS DSNREXX EXECSQL "FETCH C2 INTO :WJOBN"
    IF SQLCODE = 100 THEN ENDOFC2 = Y
    ADDRESS ISREDIT
    "ISREDIT LINE_AFTER "ADDR0W" = "WJOBN"'"
    ADDR0W=ADDR0W+1
  END
  ADDRESS DSNREXX "EXECSQL CLOSE C2"
  IF SQLCODE <> 0 THEN CALL DBERROR 'CLOSE C2'
RETURN
```

---



## 5.1.5 Potential drawbacks

As with custom queries, while it is possible to ask a great many interesting questions specific to your goals using this approach, there are two potential drawbacks:

- ▶ Someone in your organization needs to know enough about SQL to avoid issuing inefficient queries and needs to invest in learning the WebSphere Studio Asset Analyzer schema well enough to construct queries that return meaningful results.
- ▶ The WebSphere Studio Asset Analyzer schema might change between releases, which might require changes in your SQL statements if you want to keep using them after upgrading.

If this sounds like too much potential work to you, you might want to look at whether the Web service API provides the functionality that you need.

## 5.1.6 Using the URL application programming interface

WebSphere Studio Asset Analyzer includes a URL-based API, which makes possible mashup-like integration in the user interface of other tools. Alternatively, by simply launching a browser with a well-formed URL, another tool can present the WebSphere Studio Asset Analyzer information about a specific software asset. The user can navigate through the WebSphere Studio Asset Analyzer user interface to find the application insight relevant to the task at hand.

## 5.2 Related tools

One of the current megatrends in application development tooling is the increased use and sharing of application metadata among tools. This section discusses some ways that you can use the application insight in WebSphere Studio Asset Analyzer with metadata from other tools and repositories.

### 5.2.1 Asset Transformation Workbench and Rational Transformation Workbench

This section presents information about the Asset Transformation Workbench.

#### **Asset Transformation Workbench**

This desktop workbench provides interactive application analysis and transformations of mainframe applications. Like WebSphere Studio Asset Analyzer, the Asset Transformation Workbench parses application source code, builds a repository of metadata, and makes it available for the analyst and developer.

Compared to WebSphere Studio Asset Analyzer, the Asset Transformation Workbench provides richer interactivity and multiple simultaneous synchronized views of an application. In addition, it includes the following unique capabilities:

- ▶ Business rule discovery, categorization, and annotation
- ▶ Various code slicing algorithms to assist in componentizing or restructuring mainframe programs and reducing application complexity
- ▶ Automated categorization of programs by architectural characteristics, simplifying the effort to determine how “reuse-ready” an application might be
- ▶ Sophisticated query tool for finding code patterns and anti-patterns in program code
- ▶ More metrics, reports and graphs

Given the two tools different types of optimizations, some organizations choose both Asset Transformation Workbench and WebSphere Studio Asset Analyzer. A subset of analysts and developers use Asset Transformation Workbench to help with specific transformation projects. In addition, they provide WebSphere Studio Asset Analyzer to a larger set of developers for rapid application understanding and finding cross-application dependencies across tens (or hundreds) of millions of lines of code that have been scanned into WebSphere Studio Asset Analyzer's repository. Figure 5-1 summarizes the product positioning.

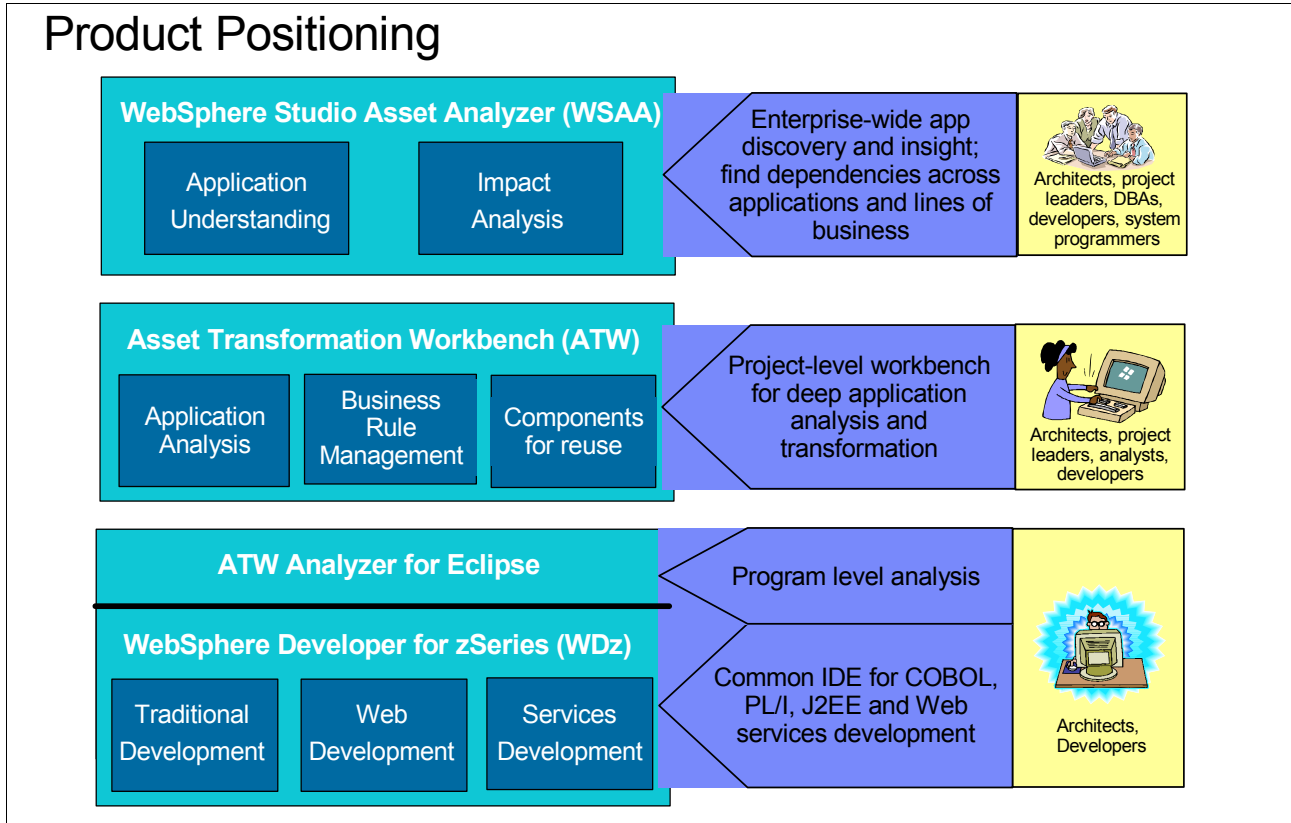


Figure 5-1 A comparison of these tools, which can be combined in a single solution

Table 5-1 provides a more detailed comparison of WebSphere Studio Asset Analyzer and Asset Transformation Workbench feature and functions.

Table 5-1 Comparison of WebSphere Studio Asset Analyzer and Asset Transformation Workbench

Usage Scenarios	WebSphere Studio Asset Analyzer	Asset Transformation Workbench
Work with z/OS Assets		
Cobol, PL/I, JCL	Y	Y
CICS, IMS, DB2	Y	Y
Assembler	Y	
Natural / Adabas		Y
Micro Focus COBOL, ACUCOBOL GT		Y
Work with Distributed Assets		
Java, Java EE, WebSphere	Y	
XML, C/C++	P	
Understand Application		
New developer	Y	Y
Outsourced AD or operations	Y	Y
Change Request / Application Maintenance	Y	Y
Compliance documentation	Y	Y
Find and manage business rules	P1	Y
Find business processes for reuse	Y	Y
Find programs and data needed for testing	Y	Y
Reduce Risk due to Changes		
Identify downstream impact	Y	Y
Project-level (millions of LOC)	Y	Y
Enterprise-wide (tens of millions of LOC)	Y	P2
Transform Application/Improve Code Maintenance/Reduce Complexity		
Refactor / restructure code / code slicing	Note 1	Y
Remove dead code	Note 2	Y
Make more accurate project estimates	Y	Y
Find programs & data needed for testing	Y	Y
Assess programs for reuse and suggest remediations	Note 1	Y
Deployment		
Use anywhere from browser	Y	P4
Run on z/OS	Y	
Scan source where it lives	Y	
Scan CICS, IMS, DB2, and WebSphere system configurations	Y	P3
Run on workstation		Y

### Table Legend:

Y = supports

P = partial support

P1 = Can discover data elements relevant to business rules

P2 = Can partition large applications into separate repositories

P3 = not WebSphere; DB2 via DDL

P4 = Static reports and business rules

**Note 1:** Can help in the analysis

**Note 2:** Can help find unused assets but doesn't identify unused code within programs

Asset Transformation Workbench includes a bridge to WebSphere Studio Asset Analyzer. Using the bridge, you can identify a set of assets defined in WebSphere Studio Asset Analyzer (typically the assets constituting an application or the results of an impact analysis) and pass this list to Asset Transformation Workbench. Then, Asset Transformation Workbench can invoke an FTP process (assuming it is set up on your mainframe) to download the sources to the workstation where Asset Transformation Workbench is running. Figure 5-2 summarizes the deployment.

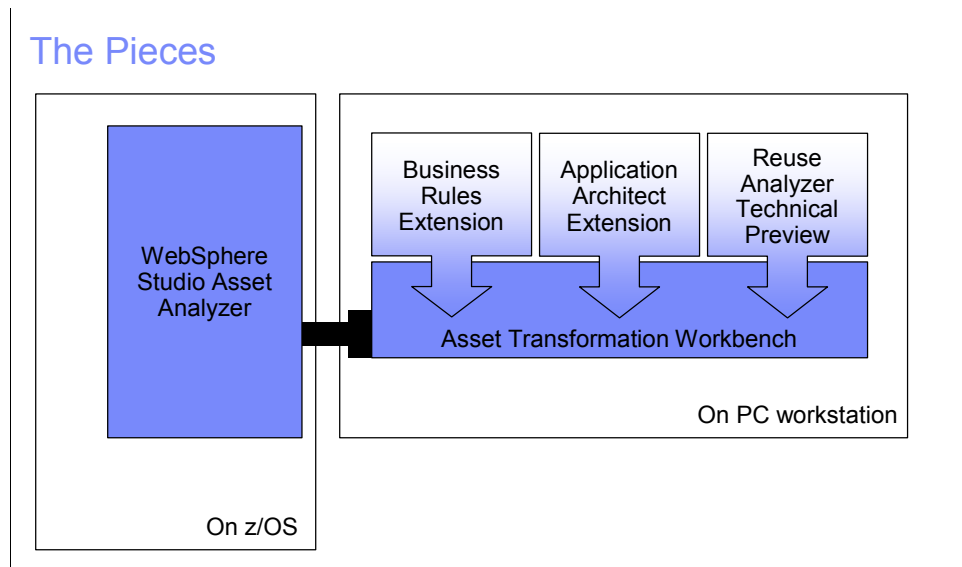


Figure 5-2 WebSphere Studio Asset Analyzer and Asset Transformation Workbench

You can find more information about the Asset Transformation Workbench at:

<http://www.ibm.com/software/awdtools/atw/>

### Rational Transformation Workbench

With release V3.1, Asset Transformation Workbench is now called *Rational Transformation Workbench*.

With the move to service-oriented architecture (SOA), many businesses are looking for ways to reuse their existing System z assets. The foundation of Rational Transformation Workbench is a knowledge base that contains rich, current insight into application portfolios:

- ▶ Detailed reports, metrics, documentation, and visualizations of the enterprise applications are readily accessible to project leaders and architects using the workbench.
- ▶ Other team members can use the Rational Transformation Workbench browser-based module to view reports generated.

Rational Transformation Workbench provides application development teams with a set of highly integrated workstation-based modules that use the application metadata in the Enterprise Application Knowledgebase to help accelerate strategic and tactical initiatives. Many development, analysis, and project management activities are streamlined with Rational Transformation Workbench, including:

- ▶ Powerful analysis and assessment tools help accelerate ongoing maintenance and enhancements.
- ▶ Optional tools expose and help manage business rules, that can simplify application reuse for SOA initiatives.
- ▶ Optional re-architecting extension helps increase the productivity of teams restructuring and componentizing applications.

## 5.2.2 Rational Developer for System z

Rational Developer for System z V7.1 is a high-productivity integrated development environment (IDE) for traditional mainframe, Web, and Web service development. Built on Eclipse and a super-set of the Rational Application Developer, it includes extensions that seamlessly provide communications for local and remote projects and movement of assets between them, local syntax checking, wizards for speeding up tasks, and a lot more.

Integration and relevance with WebSphere Studio Asset Analyzer exists on a few levels:

- ▶ **Traditional development:** After analyzing a change using WebSphere Studio Asset Analyzer, a developer then makes code changes. Rational Developer for System z is the place to do this. These tools fit together in the workflow of the analysis-development process.
- ▶ **Creating Web services from existing programs:** After determining what business functions you want to reuse in Web services, Rational Developer for System z helps to create the services. The Enterprise Service Tools perspective in Rational Developer for System z V7.1 provides an Eclipse perspective that contains the tools needed to enable enterprise applications as Web Services. Project types include CICS Web Service, SOAP for CICS, Batch, TSO, UNIX System Services, IMS SOAP Gateway, and Service Flow. In previous releases, these capabilities were called *XML Services for the Enterprise* and *Service Flow Modeler*.
- ▶ **Analysis in the local project:** A technical preview for WebSphere Developer for System z and Rational Developer for System z called *WebSphere Developer Asset Analyzer* uses the WebSphere Studio Asset Analyzer technology under the covers to add application analysis capabilities to WebSphere Developer for System z and to allow the developer to see the impact of intended changes in the context of the whole enterprise.

Rational Developer for System z was previously called *WebSphere Developer for System z* in V7, *WebSphere Developer for zSeries* in V6, and *WebSphere Studio Enterprise Developer* in V5. You can find more information about Rational Developer for System z at:

<http://ibm.com/software/awdtools/rdz/>

## 5.2.3 Rational Asset Manager

As described in the product datasheet:

A collaborative software development asset management solution, IBM Rational Asset Manager software enables organizations to identify, manage and govern the design, development and consumption of software assets, including services as part of a service-oriented architecture (SOA) initiative. The software helps IT organizations deliver

innovative IT solutions while controlling costs, reducing application backlogs and improving business flexibility and responsiveness by facilitating software asset reuse.

Using Rational Asset Manager, you can increase application development productivity and enable developers to reuse software development-related assets. For example, Rational Asset Manager allows you to create, discover, and trace service assets throughout the SOA life cycle. The software also helps monitor asset integrity and utilization through a defined, enforceable, and auditable process that includes fine-grained permissions and review approval flows.

WebSphere Studio Asset Analyzer is complementary to RAM. WebSphere Studio Asset Analyzer helps identify and visualize artifact interdependencies. Interdependencies can help identify candidate assets that can be packaged categorized and actively governed using RAM. Organizations choose WebSphere Studio Asset Analyzer when they want to understand the contents, structure, and various internal attributes related an application, including the existing interrelationships and traceability between code artifacts and their data sources.

Organizations choose RAM when they want to create, classify, search, consume, and govern assets. Assets can be any group of related artifacts that are used to solve a re-occurring business issue—not just code.

Customers use Rational Asset Manager to create a categorization that organizes all the open source assets that are allowed to be searched and used within their organization. They can use RAM to define and implement a review and approval process for those open source assets. For example, have they been vetted from an IP perspective? Have they been vetted from a security perspective? They can use RAM to search for assets by asset type, category, team space, or custom metadata attributes such as tested, key word, and user defined tags. They use RAM to get asset details like what teams are using open source assets on their project or who to contact for support or collaborate on enhancing or fixing an asset.

You can find more information about Rational Asset Manager at:

<http://ibm.com/software/awdtools/ram/>

## 5.2.4 CICS Interdependency Analyzer

As we discussed in Chapter 2, “Using WebSphere Studio Asset Analyzer to cut through the complexity of application changes” on page 9, unless you add to the metadata yourself, the WebSphere Studio Asset Analyzer metadata is limited to what can be discovered by static analysis of the program source and runtime subsystem configurations. This is a limitation of all static analysis tools.

In contrast, the CICS Interdependency Analyzer records what happens in a running CICS system and records this information as metadata in a DB2 database. You can then query this database to find your program flow, what data it accesses, and your CICS affinities.

Because WebSphere Studio Asset Analyzer and CICS Interdependency Analyzer store their metadata in DB2, you combine the statically and determine dependencies dynamically to get a more complete view of your application relationships. Similarly, you can compare actual versus potential code execution (at the program level, not at the line of code level).

From the point of view of the user of CICS Interdependency Analyzer or other runtime discovery tools, WebSphere Studio Asset Analyzer offers the possibility of looking into the *black box*. Runtime discovery tools can determine that program A called program B, but they cannot tell whether any other control transfers are possible. WebSphere Studio Asset Analyzer provides a *white box* view of program A—a view of the source code and the

program's relationships to other assets (programs, data, databases, and the artifacts needed to build the program).

You can find more information about the CICS Interdependency Analyzer at:

<http://ibm.com/software/http/cics/ianaly/>

## 5.2.5 WebSphere Service Registry and Repository

Among other things, WebSphere Service Registry and Repository provides Web service discovery at runtime. WebSphere Studio Asset Analyzer provides information for the analysis of the software artifacts behind those services and can help you to identify:

- ▶ Candidate software interfaces to expose as services
- ▶ The software artifacts that implement a service description
- ▶ The impact of changes to the service definition and its implementation

You can find more information about the WebSphere Service Registry and Repository at:

<http://ibm.com/software/integration/wsrr/>

## 5.2.6 Data dictionaries

*Data dictionaries* keep the metadata rules about the definition (meaning) and format of specific pieces of data, for example postal code (ZIP code) format, telephone number edit rules, and purchase order number definition. Data dictionaries do not keep track of every instance of the data in a particular application, whereas, for example, WebSphere Studio Asset Analyzer does keep track of every instance of a phone number.

If you combine WebSphere Studio Asset Analyzer with a given data dictionary, the developers in that organization can hypertext link from the data dictionary rules to the places in an application where instances of that particular data element exist.

You can find a good definition of data dictionaries at:

[http://en.wikipedia.org/wiki/Data\\_dictionary/](http://en.wikipedia.org/wiki/Data_dictionary/)

## 5.2.7 Tivoli Change and Configuration Management Database

The IBM Tivoli® Change and Configuration Management Database (CCMDB) stores operational information that is related to system configurations and change histories and is a core enabler of an IT Service Management solution. Its data model includes application entities, their configurations, and their interrelationships. The principal user roles are in the operations team.

By contrast, WebSphere Studio Asset Analyzer goes deeper into detailed application insight gained from the source code and discovers dependencies between and among programs, data, and other software artifacts within and across applications. The principal user roles are in the application development team. The metadata in CCMDB and WebSphere Studio Asset Analyzer are complementary, and *pointer level integration* makes possible a fuller application picture to both the operations and development teams.

For more information, see the following online resources:

<http://ibm.com/software/tivoli/features/it-serv-mgmt/welcome/itsm-platform.html/>  
<http://ibm.com/software/tivoli/products/ccmdb/index.html/>

## 5.2.8 IBM Information Server

The IBM Information Server is a platform for delivering trusted information about data and increasing the value, consistency, integration, and reuse of enterprise data. It is complementary to WebSphere Studio Asset Analyzer. The IBM Information Server specializes in metadata about data. The scope of its impact analysis capabilities is tables, views, and other database objects. It does not extend its dependency awareness to the programs that use these database objects, nor does it focus on discovery of assets. These two capabilities are provided in WebSphere Studio Asset Analyzer.

You can find more information about the IBM Information Server and its constituent product modules at:

[http://ibm.com/software/data/integration/info\\_server/](http://ibm.com/software/data/integration/info_server/)





# A

## Deployment options for WebSphere Studio Asset Analyzer

WebSphere Studio Asset Analyzer is itself a composite application with several deployment options. Figure A-1 shows the typical deployment.

All of the relevant pieces of WebSphere Studio Asset Analyzer are deployed on a System z machine. The WebSphere Application Server hosts the WebSphere Studio Asset Analyzer Web applications, which include the user interface engine, impact analysis engine, and help system. The HTTP Server (delivered as part of z/OS) is used to implement some mainframe functions and the View Source feature within the WebSphere Studio Asset Analyzer user interface. The application metadata is stored in DB2.

When scanning distributed applications (Java, C++, and so forth), the optional pieces need to be deployed on the target environment. WebSphere Studio Asset Analyzer scans both Java applications in their bytecode form from a local file system and configurations of WebSphere cells and nodes. A WebSphere Application Server running on Windows or AIX is required for the scan root; its local file system can include network mounts to other file systems on other servers, including ones running Linux, Solaris, and so forth. Similarly, the server that hosts the distributed scanners can scan the configuration of WebSphere servers running on other servers and operating systems, including ones running on other UNIX and Linux platforms or even z/OS.

Users of WebSphere Studio Asset Analyzer most commonly access their application metadata through a Web browser.

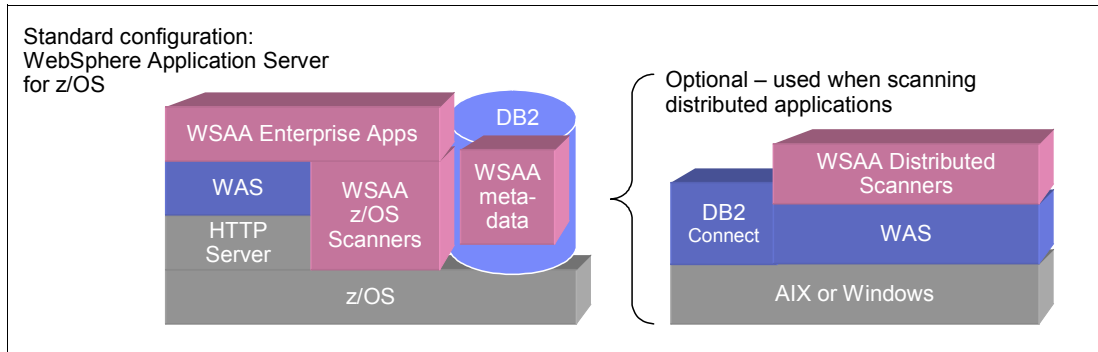


Figure A-1 The typical deployment scenario for the components of WebSphere Studio Asset Analyzer

Placing the Web applications on the mainframe is the most common scenario for a number of reasons:

- ▶ Performance, response time, and CPU: WebSphere Studio Asset Analyzer creates a large number of SQL queries when doing impact analysis and when preparing large graphs and tables of results. The proximity of the application to the metadata residing in DB2 means faster response time while at the same time avoiding CPU cycles that would be required if using DRDA® (DB2 Connect™) for remote access to DB2.
- ▶ WebSphere Studio Asset Analyzer can take advantage of System z Application Assist Processors (zAAP), a specialized processor that offloads Java-based workload from the general processors.
- ▶ The application source code is typically stored in Partitioned Data Sets (PDSes) or source configuration management systems, which are stored on the host. Keeping all the components of WebSphere Studio Asset Analyzer on the host reduces network traffic.
- ▶ When the principal users of WebSphere Studio Asset Analyzer are mainframe analysts and developers, and when the administrator of WebSphere Studio Asset Analyzer is a mainframe person, it can be desirable to keep all the components within the scope of the same organization. There can be extra overhead if a separate team is responsible for distributed Web application server operations, budgeting, and so forth.

An alternate configuration puts the WebSphere Studio Asset Analyzer enterprise applications on the distributed server as shown in Figure A-2. This scenario can be desirable if you do not have CPU available on your mainframe for WebSphere Studio Asset Analyzer or if you do not have WebSphere Application Server for z/OS installed. The downside is potential performance degradation of WebSphere Studio Asset Analyzer, increased network traffic, and CPU spent on the remote database access. The IBM System z9™ Integrated Information Processor (zIIP) specialty engines do help to offload some of this database traffic from the general processors.

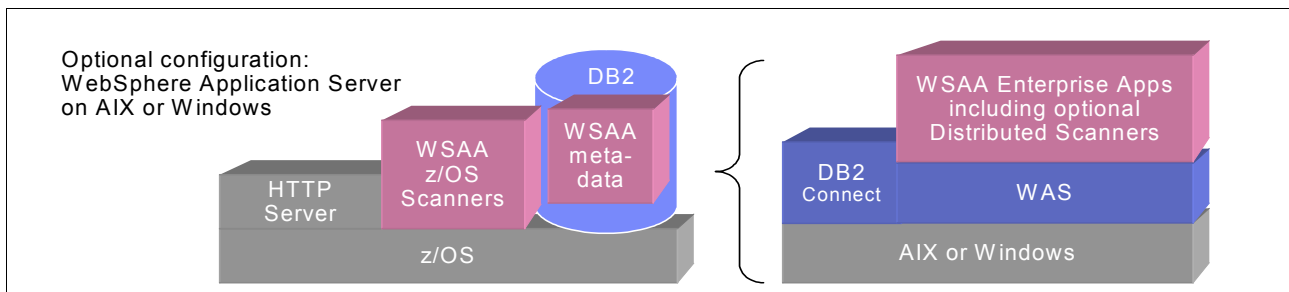


Figure A-2 An alternate deployment scenario places the .ear files (Enterprise applications) on an AIX or Windows system

Note that there are other software prerequisites and other considerations to take into account, which are discussed in the Program Directory, Getting Started and Best Practices documents. They are available at the WebSphere Studio Asset Analyzer library Web page:

<http://ibm.com/software/awdtools/wsaalibrary/>



# Related publications

We consider the publications that we list in this section particularly suitable for a more detailed discussion of the topics that we cover in this paper.

## Online resources

These Web sites are also relevant as further information sources:

- ▶ Mannion, M. and Keepence, B. “Software Reuse: Software Jigsaws.” The Computer Bulletin published by the British Computer Society September 1998.  
<http://archive.bcs.org/BCS/Products/publishing/itnow/OnlineArchive/sep98/softwarereuse.htm>
- ▶ SOA entry points  
[http://www.ibm.com/innovation/us/pointofview/soa/apr03/getting\\_started.html](http://www.ibm.com/innovation/us/pointofview/soa/apr03/getting_started.html)
- ▶ IBM Portal for SOA  
<http://ibm.com/soa>
- ▶ “Service-Oriented Architecture Compass: Business Value, Planning, and Enterprise Roadmap” by Norbert Bieberstein, Sanjay Bose, Marc Fiammante, Keith Jones, and Rawn Shah. IBM Press. The developerWorks Series. 2006. Page 98. Covers the importance of minimizing redevelopment.  
<http://www.ibmpressbooks.com/title/0131870025>
- ▶ Global Business Services Component Business Modeling  
[http://www.ibm.com/services/us/bcs/html/bcs\\_componentmodeling.html](http://www.ibm.com/services/us/bcs/html/bcs_componentmodeling.html)
- ▶ Enterprise Systems Journal, July 26, 2005. Covers how mainframe applications can be easier to reuse in an SOA than object-oriented programs.  
<http://esj.com/enterprise/article.aspx?EditorialsID=1457>
- ▶ IBM Host Access Transformation Services (HATS)  
<http://www.ibm.com/software/webservers/hats/index.html>
- ▶ Service Flow Modeler in WebSphere Developer for System z (WDz)  
<http://ibm.com/software/awdtools/devzseries/>
- ▶ IBM Information Server  
[http://ibm.com/software/data/integration/info\\_server/](http://ibm.com/software/data/integration/info_server/)
- ▶ Howard M. Hess, “Aligning technology and business: Applying patterns for legacy transformation.” IBM Systems Journal, Volume 44, Number 1, 2005.  
<http://www.research.ibm.com/journal/sj/441/hess.html>
- ▶ Facilitating Shared Services in the DoD. A Report of the Data Sharing and Services Strategy Working Group. February 12, 2006. The Association for Enterprise Integration.  
[http://www.afei.org/documents/DS3SharedServicesPaperFinalVersion021206c\\_000.pdf](http://www.afei.org/documents/DS3SharedServicesPaperFinalVersion021206c_000.pdf)
- ▶ The Web service interface for WebSphere Studio Asset Analyzer is documented in the WebSphere Studio Asset Analyzer library  
<http://ibm.com/software/awdtools/wsaa/library/>

- ▶ Developer for System z technical preview of WebSphere Developer Asset Analyzer and look in the *Downloads* section

<http://ibm.com/software/awdtools/devzseries/support>

## How to get IBM Redbooks publications

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## A

- alternate deployment scenario 64
- answering your own questions with custom queries 34
- application architecture 40
- application entropy 4
- application understanding by exploring the inventory 12
- applications 12
- applications range in size 2
- Asset Transformation Workbench 55
- associated data 36
- automated test bed 46

## B

- batch jobs 13
- benefits of open architecture 52
- BMS map for CICS 37
- business-oriented entry points 6
  - information 6
  - people 6
  - process 6

## C

- challenge to change 4
  - application Entropy 4
  - composite application dependencies 4
  - cross-application dependencies 4
  - risk to application stability 4
  - time consuming 4
- change complexity 42
- changing market and regulatory conditions 4
- CICS 13
- CICS Interdependency Analyzer 60
- CICS Transaction Gateway 39
- comparing TSO search capabilities 21
- complexity 2
- composite application 36
- composite application dependencies 4
- conditions not static 4
- cost of change 3
- cross-application dependencies 4
- custom queries 34

## D

- Data dictionaries 61
- data element details 25
- data elements 13
- data sets 13
- data stores 16
- DB2 13
- dependencies 33
- deployed applications 30
- deployment 57

- deployment options 63
- diagram of batch job 21
- discovering code for reuse 35
  - starting with data 36
  - starting with programs 37
- discovery 23
- distributed assets matching this pattern 31
- drawbacks 55

## E

- example job queries 47
- example program queries 48
- extending the metadata 52

## F

- feeding an automated test bed 46
- follow data with the impact analysis engine 22
- functions and components of WebSphere Studio Asset Analyzer 11

## H

- HATS 38

## I

- I/O record descriptor 24
- IBM Asset Transformation Workbench 41
- IBM Host Access Transformation Services 38
- IBM Information Server 62
- IBM portal for SOA 5
- identifying test cases 46
- identifying the resources required to conduct the test 46
- impact analysis 23
- impact analysis details 28–30
- impact analysis engine 22
- impact analysis results 33
- improve code maintenance 57
- IMS 13
- inventory 3
- inventory process 11
- IT-oriented entry points 6
  - connectivity 6
  - reuse 6

## J

- Java 14
- Java application exploration and impact analysis scenario 30
- job queries 47

## L

- limitations of static analysis 33

## M

mainframe assets 14  
market conditions 4  
metadata 52  
MFS screen for IMS 37

## N

new services 36  
non-technical challenges to reuse 43

## O

open architecture 12  
open metrics framework 53  
overview of the impact analysis 27

## P

paragraph flow 17  
potential drawbacks 55  
program details 16  
program diagram 18  
program metric information 15  
program queries 48  
programs 12, 39  
programs accessing data sets 37

## R

Rational Asset Manager 59  
Rational Developer for System z 59  
Rational Transformation Workbench 58  
Redbooks Web site 68  
    Contact us viii  
reduce complexity 57  
reduce risk due to changes 57  
reference graph 32  
regulatory conditions 4  
related tools 55  
reluctance to touch 3  
remediation 41  
reused application assets 36  
risk to application stability 4  
roadmap in this paper 7  
run unit diagram 19  
run unit remediation 41  
run units 12, 39  
run units as expressions of business processes 39

## S

scanners 11  
Service Flow Modeler 39  
service-oriented architecture 5  
size and complexity 2  
SOA 5  
SOA entry points 6  
source files 20  
source of copybook 26  
SQL queries 53  
starting with programs 37

micro flows to solve interface mismatch 38  
screens 37

## T

tagging 52  
technical challenges to reuse 42  
test case generation 45  
time consuming 4  
Tivoli Change and Configuration Management Database 61  
transform application 57  
transformations of data 36  
TSO and WebSphere Studio Asset Analyzer  
    functional differences 22  
typical deployment scenario 64

## U

understand application 57  
using Web service 52

## W

WebSphere Service Registry and Repository 61  
WebSphere Studio Asset Analyzer 10  
    ease of use 21  
    existing knowledge 21  
    formatting 21  
    resource usage 22  
    scanning inactive components 22  
    searching additional attributes 22  
WebSphere Studio Asset Analyzer and ATW 57  
work with distributed assets 57  
work with z/OS assets 57







**Redpaper**

# Faster Application Change and Reuse with WebSphere Studio Asset Analyzer

**Gain intellectual control over your applications**

**Cut through application complexity**

**Make changes with more confidence**

This IBM Redpaper is written for analysts, developers, architects, and technical managers with the goal of introducing ideas about mainframe application change and reuse using WebSphere Studio Asset Analyzer and related tools from IBM.

This paper focuses on mainframe applications for the following reasons:

- ▶ There are a lot of these types of applications.
- ▶ It can be very challenging to understand these applications well enough to take action to change or reuse them.
- ▶ There has been less focus on the *how* of mainframe application reuse than, for example, the reuse of Java objects and applications.

The outline of the paper is:

- ▶ **Chapter 1.** The promise and challenges of reuse
- ▶ **Chapter 2.** Using WebSphere Studio Asset Analyzer to cut through the complexity of application changes
- ▶ **Chapter 3.** Approaches to discovering code for reuse
- ▶ **Chapter 4.** Accelerate test case generation with WebSphere Studio Asset Analyzer
- ▶ **Chapter 5.** Additional value through extension and integration with other tools and metadata repositories
- ▶ **Appendix A.** Deployment options for WebSphere Studio Asset Analyzer

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:  
[ibm.com/redbooks](http://ibm.com/redbooks)**