A Case for Source Control Management

By
David Slater
IBM iSeries AD Market Manager

The penetration of Source Control Management tools is lighter on the IBM® iSeries™ server than on most competing systems. In this white paper, we'll review the historical reasons why SCM tooling usage has been light and why these tools are poised for dramatic market growth.

What Is Source Control Management

Source Control Management (SCM) is the process of managing the source code for your applications. It is the method whereby you securely and safely store the source for your applications; manage changes to this source code; promote changes to the source code through the various stages of application development; control the version, release and modification level of the source code; and finally manage the build/compile step of creating executable applications from the source code.
Typically when we think of Source Control Management, we think of the tools that provide this sort of function.... the basic synchronization "check in-check out" library control systems that prevent the changes of one developer from overwriting the changes created by another developer.

In many IT environments, SCM tooling is pervasive. If you have source code and you modify applications, it is assumed that you are using SCM tooling to manage your development process. However, SCM tooling is not pervasive in the iSeries marketplace. The current penetration of SCM products on iSeries development systems (iSeries servers that have application development tools) is estimated to be about 10% to 15%. This low percentage has a lot to do with the history of the iSeries system and its predecessors.

A History Lesson

In the beginning.... no let's not go back quite that far. The IT community has been promoting the virtues of modular programming for several decades. It improves the opportunities for code reuse and simplifies the task of maintenance. However, many applications in the midrange market were monolithic rather than modular. Why did this happen in an IT environment that promoted modularization? The simple answer is performance. When people created applications for the original midrange systems, they had to be extremely sensitive to the application's performance characteristics because the machines were not very powerful. In the S/38 days, programmers created modular applications by creating a set of programs that called one another. These programs were independent entities. When one program called another program, an environment had to be established, at run-time, to facilitate the interaction. And setting up this environment was not a trivial task. It involved many machine instructions to establish this environment. As a result, program-to-program calls were very expensive from a performance perspective, especially on small systems.

To prevent performance bottlenecks, IBM recommended that programmers avoid program-to-program calls whenever possible. As a result, many solutions became monolithic single-program applications. The need to have SCM tooling to manage the component parts of an application was not compelling when you had only one large part. Many organizations used comment lines and comment columns in the code to manage/document their source changes. There were other system characteristics that slowed the introduction and widespread use of SCM tools. The midrange market was characterized by small development shops with only a couple of programmers in each. The source control management process, the build process, and all other processes were not difficult to manage. The two developers just talked,and this prevented them from overwriting each other's source code changes.

Another contributing factor was the midrange system strategies. The IBM midrange systems were dedicated to preserving their customer and partner investments in applications. As a result, the portfolio of applications targeting IBM midrange systems continued to expand. Applications could be saved and restored from one release to the next. Unlike many other competitive systems, there wasn't a need to recompile your applications as you upgraded your system. The manadatory recompilation and source code changes required by many other systems contributed to the much higher penetration of SCM tools in these environments.

The Introduction of ILE

About 15 years ago, there were several studies published about the expanding application backlog in the IT industry. The primary reason given for this expanding backlog was the amount of effort dedicated to maintaining existing applications. Over 75% of the IT workforce was dedicated to maintaining current applications in production, leaving little resource to develop new applications. The pressure to reduce application maintenance costs was acute in the IBM midrange marketplace. In this market, there were many monolithic applications created to avoid the performance penalties associated with program-to-program calls. The drawback was that monolithic applications were extremely difficult to update and maintain. The Integrated Language Environment (ILE) was introduced to the AS/400 to encourage modular programming by allowing modular applications written in C to run effectively on the AS/400. Since C applications are very modular in nature and are call-intensive, they performed poorly on the AS/400 prior to the introduction of ILE because of the program call performance penalty.

ILE changed the program model on the AS/400. Source code was now compiled into modules. The modules had to be linked or bound into programs before they could be executed. Because modules could be bound together into a unit, call performance between these modules improved dramatically. With calls between bound modules, you no longer had to establish an environment to facilitate the calls.

ILE introduced modules and service programs to make AS/400 applications easier to update and maintain. Because modular programs had many parts to be controlled and maintained, a new IBM SCM tool was introduced called Application Development Manager in the same time frame as ILE. An impact analysis tool, Application Dictionary Services, was also introduced to manage the complexities of changes to modular applications. In V3R1 of AS/400, the Integrated Language Environment was extended to all of the AS/400 strategic languages.

The dramatic expansion of modular programming on the AS/400 would drive increased demand and usage of SCM tooling..... except that ILE compiler usage and the move to modular programming was less than dramatic; it took 10 years before the usage of ILE languages for new

development was over 50%. Many monolithic programs developed in the S/38 era still operate on iSeries machines. Today, most of the applications on the AS/400® and iSeries servers are still not modular ILE applications. Why is the transition taking soooo long? Customers don't rewrite applications that are working well. However, most new iSeries applications are modular. As more modular applications are introduced into the AS/400 and iSeries marketplace, the demand for SCM tools gradually increases.

Why Do I Need Source Control Management Now ?

A couple of recent changes in the world of application development may significantly increase the market demand for SMC tools. There is an ever-increasing demand for Web applications. The demand for e-business solutions will drive approximately 50% of the new server volumes this year. The iSeries AD team shipped a new product called WebSphere® Development Studio for iSeries in May of 2001. This new product consolidated all of the AD tooling for both traditional and e-business development into one iSeries host product. WebSphere Development Studio is a no-charge upgrade from the iSeries AD tools such as RPG and ADTS with software subscription. Since May 2001, IBM has shipped over 50,000 copies of WebSphere Development Studio. This new product has dramatically increased the penetration of e-business development tools in the iSeries marketplace. The introduction of Web and Java development tools and the WebFacing Tool has dramatically increased the number of Java components that have to be managed. If you WebFace an application with 5,000 screens, you will have to manage a Web development project of at least 20,000 parts with an input and output JSP and an input and output Java bean per screen. The Java components of the Web development project may be distributed across multiple tiers and multiple platforms and will almost never be stored in the iSeries native file system. Managing applications in an e-business environment is complex. The need for SCM tools has increased in proportion to this complexity.

The next refresh of WebSphere Development Studio will be based on the new WebSphere Studio family of products that features the WebSphere Studio Workbench. WebSphere Studio Workbench is a new Integrated Development Environment with tooling infrastructure to enable IBM and vendor AD tooling to interoperate effectively. To promote this new cross-platform IDE and to encourage vendors to integrate their tooling offerings to this IDE, IBM donated the WebSphere Studio Workbench source code to the open-source community. Enhancements to the WebSphere Studio Workbench are now managed by a board of directors composed of representatives from the key development organizations that have decided to integrate their AD tools to WebSphere Studio Workbench. For more information on the operations and management of the WebSphere Studio Workbench project, refer to www.eclipse.org

The key integration point that facilitates the interoperability of IBM and vendor tools is the support of SCM tools. For developers to take advantage of the key benefit of WebSphere Studio Workbench, the easy interoperability of IBM and vendor tools, they will require SCM tools integrated to WebSphere Studio Workbench.

Expect demand and usage of SCM tools to increase significantly for two reasons:

- The increase in Web application development will cause an explosion of Java parts that will need to be managed.

- The introduction of WebSphere Studio Workbench tooling in WebSphere Development Studio requires SCM tooling to facilitate AD tool inter operability.

Who Needs Source Control Management?

Development teams who are involved in the following development scenarios are most likely to require SCM tools:

- Building complex modular applications requiring the library control and build control tools to maximize productivity
- Building Web applications that access iSeries data or applications. This activity creates many Java components that need to be managed. WebFacing an application with 1000 screens, will generate over 4000 Java components (JSPs, beans, and servlets) that will have to be managed.
- Maximizing the tool integration capability of WebSphere Studio Workbench while using multiple tools to create new e-business applications

At least one of the scenarios listed above will be the norm for most development organizations. SCM tools will become standard in the future.

Functions Required

There will always be debates about the functions and features that are required for SCM solutions. However, there are some key capabilities that are required to maximize the benefits of our iSeries AD strategy.

Most iSeries SCM tools started their life managing RPG, COBOL, and maybe C source code in the native file system. Since almost all iSeries source code was stored in the native file system, the SCM tools had great coverage. However, the world has changed. To be effective in the world of e-business development, SCM tools need to be able to manage C++ and Java source code in the Integrated File System.

IBM has introduced a new cross-platform Integrated Development Environment called WebSphere Studio Workbench (a.k.a. Eclipse). This new IDE provides the tooling infrastructure to allow both IBM and vendor AD tools to interoperate. The integration point for tool interoperability is an SCM interface. Therefore, SCM tools need to be able to integrate with the WebSphere Studio Workbench to maximize the value proposition of our iSeries AD strategy.

IBM Products

There are two priced features of WebSphere Development Studio for iSeries -- Application Development Manager and Application Dictionary Services. We are planning to discontinue

these products after the next release. We are providing over a year of advanced notice so our customers will have enough time to migrate to alternative tooling.

We decided to quit marketing ADS and ADM for several reasons:

- ADM and ADS didn't support C++ and Java. These are the two most important e-business development languages.
- ADM and ADS only supported the native file system. All Java and most C++ program source code is stored on the Integrated File System.
- There are strong competitive products to ADM and ADS that provide Java, C++, and IFS support. ADM and ADS are neither functionally competitive nor dominant in the iSeries marketplace
- IBM would have to make substantial enhancements to ADM and ADS to make these products functionally comparable/competitive to the vendor products. Just making ADM and ADS competitive with the dominant vendor products would not provide sufficient reason for customers to move to the IBM products.

Partner Products

We are currently working with three key iSeries partners that provide competitive products to ADM and ADS. The vendors have agreed to :

- integrate their tools to the WebSphere Studio Workbench
- support Java and C++ as well as the IFS
- provide a migration plan for ADM and ADS customers to migrate to their solutions

The three key iSeries Source Control Management tool vendors are Aldon, MKS, and SoftLanding. The vendors support the native file system and the IFS and have support for Java and C++. They have plans to integrate their SCM tools to WebSphere Studio Workbench, and they have migration plans for ADM customers to their SCM solutions. For more information about their plans to support migration and WebSphere Studio Workbench, refer to their Web sites:

- Aldon - www.aldon.com
- MKS - www.mks.com
- SoftLanding - www.softlanding.com