Meeting the Challenges of Change Management

CA Endevor® Software Change Manager and how to do lifecycle management right

Author: Michael Azoff
Publication Date: March 2012
Executive Summary

MEETING THE CHALLENGES OF CHANGE MANAGEMENT

There are a number of challenges facing organizations with significant mainframe operations. In all these organizations, it is safe to say, the mainframe is used to support mission critical applications, and the majority of these are written in COBOL. These enterprise business applications can amount to thousands of components, and the process and associated cost of migrating off of this legacy code is the least attractive option. However, the task of continued development and maintenance of this code base is becoming more challenging.

Regulatory requirements and compliance policies mean that the management of this code needs to be tighter: it needs to be audited and safeguards need to be in place to mitigate risks. For custom applications, and many of these mainframe applications are either in-house built or customized brought-in applications, the challenges are even greater: managing the software lifecycle is a challenge in itself.

Development in concurrent work styles and across parallel deployments means that sources of error are high – the complexity factor can lead to synchronization errors, as wrong libraries are used, code goes missing, or dependencies are missed causing applications to break.

The aging mainframe workforce is another factor to consider: the new generation of developers is trained on distributed systems and the difficulty of replacing mainframe and COBOL developers puts more pressure on the remaining staff.

CA Endevor® Software Change Management (CA Endevor SCM) is designed to meet these challenges for mainframe systems running IBM z/OS. The SCM name tag is an understatement in Ovum’s opinion: CA Endevor does SCM and Configuration Management, Build Management, Release Management, Deployment Management, and it has an Integrated Development Environment (IDE), named CA Endevor® Quick Edit (a separate module). From a lifecycle management view point, CA Endevor SCM encapsulates so many of the lifecycle practices in an integrated manner that it offers advantages beyond the separate management activities. Thus a build is performed with CA Endevor SCM knowing what files are up to date, which are tested and do not need recompiling, what (if any) elements are missing in a package comprising hundreds of elements.

CA Technologies released a mainframe Web Services based Eclipse User Interface (UI) that interfaces directly with CA Endevor SCM providing application developers with a modernized UI for enhanced productivity. The Web Services Application Programming Interface ensures that CA Endevor SCM-managed applications tie in with management tools on distributed machines, where in many cases, the front-end of these applications often reside. The use of CA Endevor SCM can
increase the longevity of such applications. The deep automation means that the increased workload on software change control administrators, due to retiring and irreplaceable mainframe staff, can be taken up by CA Endevor SCM. The case studies reveal that a small number of administrators can manage many hundreds of developers.

CA Endevor SCM also makes use of the CA Mainframe Software Manager™ (CA MSM). By utilizing CA MSM with CA Endevor SCM, both mainframe experts and novices can obtain and install software upgrades and carry out maintenance in less time. The product also facilitates the deployment of the software into a testing area and assists in managing the steps required to configure the product for use. The marriage of these products provides significant business value through reduced software Total Cost of Ownership (TCO), helps to ensure better adherence to best practices and facilitates more accurate results by reducing the opportunities to make errors or misinterpret instructions.

In conclusion, CA Endevor SCM can make a difference to mainframe operations, increasing use of, and continuing returns on mission critical legacy applications.

z/OS Change and Configuration Management Challenges

As IT Development functions are increasingly expected to ‘deliver more with less’, effective change and configuration management has become a corporate-demanded necessity - driven by increased business demand for technology investment, the growing complexity of business and IT operations, and the need to provide governance-dictated transparency.

From an IT Infrastructure Library (ITIL) perspective a change is defined as the addition, modification or removal of anything that could have an effect on IT Services. With change management, the process ensures that standardized methods and procedures are used for efficient and prompt handling of all changes, in order to minimize the impact of any change-related incidents upon IT service delivery.

From an application management perspective, change management should:

- Ensure standardized methods, processes and procedures are used for all changes to the application estate.
- Facilitate efficient and prompt handling of all changes presented by the Business, IT Development or any other applicable requestor.
- Maintain the proper balance between the need for change and the potential detrimental impact of change.

Configuration management is the process of providing a logical model of the IT Infrastructure by identifying, controlling, maintaining, and verifying the versions of all configuration items, where a
configuration item may be described as any component, IT or otherwise, used in the delivery of IT services. Examples include:

- Hardware or software.
- Documents, e.g., Service Level Agreements or process flows, roles, and responsibilities.
- Source code.

In terms of source code management, effective configuration management should ensure that an organization starts with a documented and agreed baseline for each application, including associated attributes such as dependencies with other applications. Moving forward, an organization should then ensure that all changes to this baseline are closely controlled with a fully documented audit trail that details who did what, when, and why.

FEWER AVAILABLE EXPERTS RELYING ON MANUAL PROCEDURES LEADS TO PROBLEMS IN PRODUCTION

While many thought the mainframe would become virtually redundant in the new millennium, the platform is experiencing resurgence in popularity. Unfortunately, however, this is not matched by the level of available mainframe programming resources, which has dropped significantly in recent years. The learning of mainframe languages is following a similar path to the decline of Latin as a day-to-day language – usurped by the demand for, and the associated desire to learn, en vogue languages such as Java and Ajax.

The ability to meet the demand for mainframe programming skills is not only complicated by the lack of new blood – it is, unfortunately, under a two-pronged attack. Organizations are affected not only by the shortage of new mainframe programming resources, but also by the loss of the existing, aging mainframe workforce to retirement.

To compound this further, the organizational reliance on manual processes and procedures has many pitfalls. Firstly, manual procedures are often home-grown, and rather than being good practice are often the result of internal thinking on historical requirements that might have changed considerably since the procedures were developed without any compensating changes to the process.

Beyond the adequacy of any in-house procedures, the manual nature of their performance is reliant upon their correct and consistent application - even if the procedures were originally mapped or documented by their creators, are they still referred to during application or are they conducted from memory? In the case of the latter, there is a high probability that the procedures are inconsistently applied either through the introduction of ‘personal improvements’, short-cutting, or the ‘Chinese-whispering’ of the procedures as they are passed on to new recruits. Additionally,
steps may just be distorted, accidently left out, or forgotten; particularly in times of high workloads or time-critical deadlines.

Manual procedures also do not lend themselves well to audit-mandated controls such as the separation of duties or independent approval of business application changes. Unlike with automated, stage gate-based processes, it can be difficult to enforce these internal controls if the same individual plays multiple parts in a process and a self-handover can be used to save time, omitting the required level of approval.

**Manual procedures do not lend themselves well to audit-mandated controls such as the separation of duties or independent approval of business application changes.**

IT governance is also adversely affected in that manual processes often lack sufficient evidence of the application of required internal controls – given that the required evidence, or audit trail, is often also manually created and subject to the same risk of non-adherence or performance as the core process itself. Inevitably, the sample-based review of such controls, by either internal or external auditors, will identify instances where there is no evidence of the mandated control being applied; and, even if it is just the non-documentation of the approval, it will be taken as non-approval for compliance purposes as it cannot be evidenced and therefore proved.

Manual processes are, by definition, labor-intensive, not only within their core but also at the periphery, and the promotion of code to production is a good example. In many organizations, the use of manual processes necessitates an additional resource for the promotion piece of the overall change process. This can extend the process with a requirement for significant lead time to schedule, approve, and execute the promotion. Also, the use of a single team for promotion, whilst good from a control perspective, does offer up the equivalent of a ‘single point of failure’ and the possibility of bottlenecks.

Manual procedures can also be cumbersome, requiring scarce IT resources to undertake administrative-level tasks at the expense of higher value-add activities. This is not only a sub-optimal use of the scarce resource, but also another opportunity for process steps to be missed due to boredom or the desire to quickly get back to the ‘real work’. There is also scope for manual processes to become disjointed or bloated as work-flow bottlenecks may not be instantly identifiable (if monitored at all) and process inconsistency can cause duplication of effort or the need for reworking, most likely increasing the required level of resource and the time to achieve process completion.

While the above outlines how manual processes are inherently susceptible to misapplication and the breakdown of the required levels of internal control, the real cost to the business lies with the
ineffectiveness of process delivery. For example, the promotion of bad code to production is detrimental to the business on a number of levels:

- The failure to deliver changes to critical IT services in required timescales or, whilst promoted code is backed out, the inability to deliver all or part of said critical IT service. This can potentially result in a loss of revenue and degradation of brand value or customer loyalty.
- The cost of reworking code to meet the initial desired requirements, particularly given that the generally accepted rule of thumb is that for every dollar a code-fix costs at the testing stage, (or prevention costs during development), it costs US$10 during Quality Assurance (QA), and US$100 once in production to deal with failure.

The 1:10:100 Quality Rule reflects the lower cost of prevention or early detection of a fault, against detection at QA (one order greater) to discovery in the field (two orders greater).

A good indicator of ongoing deficiencies within existing manual, or semi-manual, processes (in addition to the failure to meet both internal and external compliance requirements) is the number of back-out or emergency changes required. From an ITIL problem management perspective, trend analysis of back-outs and emergency changes should be regularly conducted to identify the root cause of these persistent incidents. This will demonstrate the true cause of the code-related incidents and allow an organization to deal with the real cause of the problem rather than the visible symptoms, i.e., that persistent coding errors are only the symptom of a continually failing manual process.

Hence, an IT Development function (and the larger IT organization) needs to appreciate that they should not be firefighting individual code-related ‘fires’ but addressing the process that allows, or even encourages, them to occur. Even with the utilization of point tools, the reliance on manual processes and the people that deliver them result in both a suboptimal use of scarce IT resources as well as the potentially avoided loss of business-critical service availability.

While the above articulates the more internal, within-process drivers for moving away from the reliance on manual processes and procedures for the management of source code, there are increasing demands at both a corporate and IT-level for increased governance – encapsulating both big and small ‘C’ compliance.

Internally, compliance assessors will need assurance that there is adequate control over the source code and the movement of new or changed code into production as a minimum. There must be access controls, sufficient separation of duties, and a robust audit trail that details who did what, when, where, and why. Additionally, recent events have raised the profile of, and ongoing need for, external regulation and the IT Development function is not immune. Sarbanes-Oxley
(SOX), the Health Insurance Portability and Accountability Act (HIPAA), and other industry-mandated requirements demand that IT demonstrate the adequacy of internal controls and a required level of visibility.

Manual processes struggle to meet the demands of both internal and external-based assurance – there are too many opportunities for process steps to be forgotten, missed, or circumvented and, unfortunately, manual steps are often the ones that provide or demonstrate the adequate level of internal control. Compliance-based audits can require that internal controls be demonstrated on an application-by-application basis and, unfortunately, an already labor intensive activity becomes a Herculean task if all applications cannot be demonstrated as being managed by the same, consistently-applied process and tools.

CA Technologies reports that the average code base across CA Endevor SCM customers is 20 million lines of code, giving an asset value of circa US$360m. Compliance activity quite rightly assesses the level of control applied to corporate assets to protect them from loss or misuse (accidental or otherwise). Source code is such a corporate asset, with an industry standard ‘cost of line of code’ estimated at US$18; with large organizations managing millions of lines of code, the math is quite frightening. CA Technologies reports that the average code base across CA Endevor SCM customers is 20 million lines of code, giving an asset value of circa US$360m. Whilst the immediate cost of lost or ‘mislaid’ code can be considerable, the knock-on consequences can be much more serious as business processes are adversely impacted by the inability to address critical application issues.

Code migrations pose a logistical nightmare for large applications, and mission-critical mainframe applications can involve hundreds, sometimes thousands of component files. Many organizations create a common unit within IT to service the migration requirements of the whole department, which can run to thousands of developers. Developers have to forward their requests and wait to be serviced, creating a workflow bottleneck, and the whole process of scheduling, approving and executing a migration can require a significant amount of lead time. Development work typically occurs on both in-house and outside vendor software that needs customization. As often happens, developers working concurrently on the same code or file element create a high risk situation for bugs to be introduced. This can be the result of badly synchronized changes that override work or the result of poor merging. For example, suppose a defect is found or a change is requested and an emergency fix is prioritized while another change request (affecting the same code) is in progress – without tracking of changes the emergency fix may be lost or may conflict with other changes in progress. The challenge is to identify which changes affect the same file/code and resolve conflicts.
A related hazard occurs with parallel development. Organizations typically manage mainframes with multiple logical partitions, to reflect different production applications or different regions with unique configurations. There is a need, therefore, to manage parallel deployments using different promotion schedules.

The next section looks at how CA Endevor SCM can meet these challenges.

**MEETING THE CHALLENGES WITH CA ENDEVOR SOFTWARE CHANGE MANAGER**

**Lifecycle Challenge**

The population of mainframe practitioners is in a decline, predominantly comprised of developers trained in COBOL. Colleges are training the new generation of developers in languages like Java, and on distributed systems. Therefore the reliance on a shrinking pool of skilled mainframe developers will increase. There is a risk to the service quality where manual lifecycle methods are used when fewer staff are available: under pressure to do more as staff leave and are not replaced, mistakes happen more frequently and production applications become compromised. Automation to replace the manual processes is the most effective way to reduce the pressure on mainframe administrators. Examples from our two case studies reveal that in one case just one administrator at any given time is needed to oversee 4,000 developers, and in the other case 1,200 developers are managed by five administrators.

At the same time there will be many additional benefits that an automated solution like CA Endevor SCM can introduce. The key is a managed process. The lifecycle approach to software development has had a dual history although it originated with the mainframe. As distributed computing rapidly gained ascendance, its community lost the knowledge gained in the mainframe environment. Whereas the concepts of process and lifecycle management gained maturity on the mainframe, the new generation of distributed developers and managers are only recently appreciating the need for such an approach to software development, maintenance, and delivery. CA Endevor SCM encapsulates process and lifecycle management in one tool, and this is a notion that is lost on the distributed community where build tools are separate from the SCM tool. The advantage of having build and SCM integrated is the control and automation that becomes available: CA Endevor SCM knows which files in a package have changed and need to be rebuilt, and those files that have been tested and do not need recompiling. This saves unnecessary rebuilding of files and the associated wait time.

To understand the equivalent of CA Endevor SCM in terms of distributed computing lifecycle management tools: Software Change Management; Build, Deploy, and Release Management; process and lifecycle management; and IDE are all rolled into CA Endevor SCM.
Once the lifecycle is defined the resulting details can be adjusted but the basic lifecycle needs to be done right, and it will then last for years.

The starting point with CA Endevor SCM is in defining the lifecycle to be used. A process expert is recommended to identify stakeholders and discuss the needs of the developer teams and the organization. CA Endevor SCM can be used in a traditional environment using a stage-gated or waterfall process or one using an Agile software development methodology. The processes can be defined with Job Control Language (JCL) – the IBM z/OS scripting language. Once the lifecycle is defined the resulting details can be adjusted, but the basic lifecycle needs to be done right, and it will then last for years. A good way to think about this is as “templates”; the administrator can define various templates of lifecycles that different applications or projects can then use and reuse. If a new one, unique from the current inventory of available templates, is required, that can then be defined and added to a site’s “lifecycle” offering.

CA Endevor SCM users work in an Environment and although some users work in just a single, default Environment, it is possible to set up multiple Environments and define separate processes that apply in each one. In one of the case studies reported below, the process expert created four Environments:

- **Developer**: Here developers have the maximum freedom, and there is the least process control. It is possible to introduce unit testing here and set up continuous integration.
- **Quality Assurance (QA)**: The QA team performs many different types of testing in this environment.
- **Pre-production**: Once code reaches this environment it is almost ready for release. Any final checks are performed here and this environment serves as a holding point.
- **Production**: Packages migrated here are used in production and represent the released code.

A map of the lifecycle is shown in Figure 1. The Figure shows the four Environments with two Stages per Environment and the migration paths of Packages (pkg), and the round circles with arrows show the tests that are run at each Stage.
Depending on needs, many distinct Environments can be created and CA Endevor SCM provides a capability to dynamically create isolated project work areas called “sandboxes”. Sandboxes should be used whenever possible because they automatically inherit the definitions of the inventory areas to which they are defined. They do not require any extra effort by the administrator to create or use. Once created, Environments are designed to be kept: it is not a short-term or temporary space. A non-Sandbox Environment therefore represents the lifecycle template.

CA Endevor SCM performs the functions expected in a SCM system: it manages version control and automatically tracks changes. A baseline is maintained and changes are kept in a delta format for maximum efficiency. CA Endevor SCM can track changes against every entity stored in its system, whether document, script, source code or compiled code, and controls this at every stage of the lifecycle. If an item is deleted, then CA Endevor SCM automatically performs associated tasks, removing all histories of the item as well. This level of automation reduces the burden on the
administrator – he or she can trust CA Endevor SCM to perform the correct in-built process. There are cases where the administrator will want to override an in-built-process. The case of ‘delete’ is a good example – to save developers from themselves, a prudent administrator will build a process to catch deleted items and save them in case of recall.

CA Endevor SCM has a role-based security system that works with CA Top Secret®, CA ACF2® and Resource Access Control Facility (RACF), or other local site security software. The roles in the lifecycle need to be understood at the start of implementation so the process expert needs to engage with these users and build their activities into CA Endevor SCM’s process.

The most used item in CA Endevor SCM is the Package which allows for files/elements to be grouped together – for example all the files associated with a release. The Package is then treated as a single unit for rapid and easy movement, promotion, and reporting. All of the items within the Package may have a Change Control ID (CCID) which identifies it throughout its lifecycle and is stored safely. Therefore any item can be recreated years after it has been used. The in-built management system can be relied upon for large workloads and fewer administrators are needed to ensure its correct operation.

The Lifecycle Management system ensures consistent, repeatable procedures. Once development is stabilized the software supports division of duties by providing process experts capabilities to act upon the Package: Cast, Review, Execute, Ship and Commit. A developer creates and works on the items destined to be placed within a Package and then when it is ready for the next stage the Package is Cast. In a Cast CA Endevor SCM validates that elements in the Package compile, binaries and version numbers match source code, required input components are included, and any other critical components are present to ensure the Package’s integrity. Once Cast the Package is moved to the review stage where authorized approvers perform a human check and if passed, it is moved to the execution stage where it can run as a submitted job or as a foreground execution, or become a part of an automated job submission procedure. Post-execution, the Package contents may be shipped to additional locations within or beyond the operating area. A final safety stage is the Commit, so if a problem in the package is detected during the execution stage there is an opportunity to back-out the package. This operation is fast, taking seconds, and CA Endevor SCM performs all the housekeeping necessary for this task, re-introducing any modules that had been replaced in the rollout. If there are no problems then the Package is Committed and back-out information is dropped. The Promotion Package capability extends this entire operation throughout the lifecycle template by automatically preparing a Package for the authorization step immediately following successful execution from a prior stage.

The above is one lifecycle scenario and the administrator can build whatever stages (mapping and routes) in a particular environment that the site requires. The flows between environments are referred to as maps and there can be many maps that lead to Production code. As mentioned,
sites vary from needing a simple two-stage, single-environment lifecycle to ones having multiple routes to production. The key benefit of CA Endevor SCM’s lifecycle management system is that it keeps track of all elements and their dependencies so that the administrator does not have to remember what fits where or how.

The Build Management facility in CA Endevor SCM allows any process to be defined, and automated. A ‘build’ can mean producing executable modules from source, executing a JCL validation routine, a software metric measurement, a QA testing tool – there is complete flexibility. The benefits to the business are: better integration with other z/OS development tools, achieving better Return On Investment (ROI); the flexibility and deep in-built automation which encourages developers to make full use of it; and the automation ensures faster and more consistent delivery of changes and applications.

CA Endevor® Quick Edit provides a single console from which developers can perform a variety of actions.

The IDE in CA Endevor SCM is called CA Endevor® Quick Edit. It provides a single console from which developers can perform a variety of actions: it provides direct editing of source files, improves developer productivity by limiting the steps needed to make changes, and makes the most commonly used features of CA Endevor SCM available on one screen.

CA Endevor SCM’s Eclipse User Interface (UI) lets developers work in this popular IDE on a distributed machine without having to directly logon to CA Endevor SCM. For the new generation of CA Endevor SCM developers this should prove to be a popular option.

Testing Challenges

Testing is a critical step in the software development workflow and the speed at which application development can transition to testing tasks becomes more crucial as delivery windows continue to shrink. CA Endevor SCM and the Application Quality and Testing Tools for Mainframe products from CA Technologies are leveraged together to address this requirement. CA Technologies Application Quality and Testing Tools take advantage of CA Endevor SCM’s automatic creation, foot-printing, and storage of source code listings. This integration between products saves application developers and application testers from manually mapping source code listings to program executables to build the symbolic file which is the necessary prerequisite of all testing products. Instead, the listings are imported dynamically, on demand, from CA Endevor SCM to the CA Technologies Application Quality and Testing Tools symbolic file where they can be reclaimed as needed to display the corresponding source statements for the program currently being tested, regardless of the source code version or software development life cycle location.
Auditing Challenges

Greater compliance and regulatory requirement demands continually affect software development and maintenance. External and internal auditing is common in many businesses and CA Endevor SCM provides excellent support for auditing. Change tracking keeps a record of every change to code or files, what was run and when. Control of access helps ensure correct separation of duties. Release Management is supported by electronic approvals and automatic configuration audits can detect and report anomalies: for example, wrong versions of components used in a build. CA Endevor SCM provides a central source of truth for promotions: what went where and who approved it. The Review capability provides the setting for human approval, for example by a project manager or team leader, helping support regulatory requirements. CA Endevor SCM keeps track of all files, so there is no chance of a lost source file, for instance, and automated procedures help ensure that all components are retained and accessible and applications can be easily recreated to any point in their history.

Migration challenges

The software development workflow involves working with batches of code or files, resulting in bottlenecks. CA Endevor SCM helps reduce these bottlenecks in a number of ways. For Release Management the Package facility ensures that all files related to a specific release are kept together – CA Endevor SCM treats a Package as a distinct entity, simplifying complex management tasks. Packages are typically used for electronic approvals, configuration audits (where CA Endevor SCM automatically detects and reports anomalies, such as wrong versions of components in the last build or a last build that was not successful), automated promotion, automated restarts in event of failure, and easy rollbacks (or back-outs).

The Configuration Management features also reduce bottlenecks. Processes required to configure items, such as makes and builds, are handled automatically and consistently and according to site standards. CA Endevor SCM automatically correlates items so that executables and source code that are output are cross-referenced to the components that were used to build them. Inter-component relationships are also automatically captured, including custom, user-defined component relations, to provide impact analysis information. The key business benefits are: an automated knowledge repository of how components inter-relate, and a fast ability to perform impact analysis resulting in faster project delivery times.

Concurrent development can pose hazards as mentioned, and CA Endevor SCM can detect, capture, and resolve concurrent changes so that changes are not accidentally regressed. CA Endevor SCM provides the capability to implement a single sign-out policy (locking) to ensure that only one developer is working on an element at a time. The direct benefits of a single sign-out
policy to developers are that workload can be shared among resources more easily, programs can be moved into production more rapidly, and production programs have fewer errors to resolve.

The cost of a single sign-out is bottlenecks caused by the locks held on the resource. CA Endevor SCM can be configured so that the sign-out checks can be used as notification that other people are working on the program rather than a hard lock. The notification ‘mode’ supports concurrent development implementation that is supplemented by the CA Endevor® SCM Parallel Development component.

When combined with CA Endevor SCM, the CA Endevor SCM Parallel Development component tracks who is working on an element in parallel and can merge different versions of code together if they become out-of-sync, preventing change regression. CA Endevor SCM Parallel Development compares the original source code against the new versions and produces a Work in Process (WIP) file. The WIP file contains annotations identifying the differences between the two versions and highlighting possible code conflicts. The WIP file can be reviewed for any inconsistencies and edited to correct them, and once the differences have been reconciled, the WIP file is merged into an updated version of the source file. CA Endevor SCM Parallel Development can also be used in a standalone mode to merge changes in separate software development lifecycle paths not under CA Endevor SCM’s control.

Parallel deployment is managed by using logical partitions, with each one applied to a distinct production application system. This allows automated lifecycle management processes to be adapted to each application, and the automated release management helps ensure that the right components are included in each application. These features in CA Endevor SCM increase productivity and enable staff to handle an expanding workload.

*Synchronising a multi-platform roll-out, involving mainframe and distributed system servers, can now be managed by CA Change Manager Enterprise Workbench.*

Keeping a roll-out synchronised is one of the biggest hazards for the migration of modern applications into production which have components on both the mainframe and distributed machines (typically the front-end). An additional interface to CA Endevor SCM called CA Change Manager Enterprise Workbench takes control of the whole operation (mainframe and servers) as a single operation: it correlates the mainframe and distributed server parts and neither package can move through the lifecycle until the other platform’s changes are complete and all required approvals are obtained. No manual checking of the migration is needed; the requirements setup must be met before mainframe and server sides are promoted. If at any point the promotion fails (on either platform), then the application is returned to its previous state. CA Endevor SCM and CA Change Manager Enterprise Workbench bring failsafe automation to tasks that are common sources of errors.
Case Studies

CASE STUDY ONE – A US HEALTH INSURANCE COMPANY

This US-based, CA Endevor SCM customer is a division of a national federation of independent, community-based, and locally operated health insurance companies. The customer has been using CA Endevor SCM for over ten years but has seen most benefit since upgrading to what was then the current release five years ago to benefit from CA Endevor SCM’s automation features and the simplification of application management processes. The customer has over 100,000 COBOL programs being managed within CA Endevor SCM.

Business Challenges

The customer’s main business challenges related to the application, and demonstration, of robust internal controls for assurance purposes. Pre CA Endevor SCM, source code was kept in individual libraries with no central management or control, resulting in a higher risk of source code loss or version confusion. Pre upgrade, there were difficulties in meeting steep audit requirements to meet governance-mandated levels of internal control. At both points in time, with the changing of the status quo, there was the added organisational (or cultural) challenge of making programmers want to use the tool.

Outcomes

The removal of the customer’s reliance on manual procedures, and their inconsistent application, immediately improved governance and compliance – gaining better separation of responsibilities and a robust audit trail (which in itself was inconsistent within their previously manual procedures). Now application development processes and activities consistently meet the requirements of both internal and external auditors – gaining Green Diamond status. Beyond the required improvement in internal controls, the customer also benefited from the increased speed and simplicity provided by automated workflows. With promotion management, changes are no longer overly reliant upon change management resources, speeding up the process whilst still ensuring separation of duties. It has also made it easier for programmers: automation speeds up moves and promotions and the use of packages has increased productivity and the flexibility of scheduling.

CA Endevor SCM has also delivered better control over concurrent and parallel development, and build management ensures that programs are consistently compiled the same way. There has also been a reduction in lost code; whilst code can still be deleted from within CA Endevor SCM, it requires a confirmatory command to do so.

People challenges were quickly addressed by the demonstration of how CA Endevor SCM could make the lives of programmers easier and protect them against human error. The separately licensed CA Endevor Quick Edit module was quickly adopted by programmers as a personal...
efficiency enabler. Specific features were greatly appreciated such as CA Endevor SCM’s support for concurrent development, including the merge facility that runs versions together and indicates overlapping line changes. Other examples included the way that changes can be backed out in seconds (with the required level of approval) and the Eclipse UI for CA Endevor SCM that negates the need to log into the mainframe directly.

“Only five CA Endevor SCM administrators are necessary to manage 1,200 mainframe programmers” – case study spokesperson.

Whilst the customer did not record specific tangible savings accrued from the deployment of CA Endevor SCM, they did identify many efficiency or effectiveness benefits. For example, even though all of its 1,200 mainframe programmers are using CA Endevor SCM, only five CA Endevor SCM administrators are needed, the customer adding that “it runs itself”. They are also pleased with the low level of ‘trouble calls’ that result from programmers not knowing how to undertake a particular activity, which they attribute to CA Endevor SCM’s ease of use.

On a staffing level, the customer previously needed to use a Scheduling Group resource. CA Endevor SCM automation has meant that this resource has been reprioritised, the timing issues caused by manual promotion have been removed, and the speed and exactness of automated promotion have also seriously reduced the business impact of application change activity. The customer also highlighted that CA Endevor SCM’s facility for the grouping of monthly releases, in their case for hundreds of COBOL programs, also massively speeds up the process of promotion to production.

CASE STUDY TWO – A US NOT-FOR-PROFIT FINANCIAL PRODUCTS AND SERVICES PROVIDER

This US-based, CA Endevor SCM customer is a not-for-profit organisation providing financial products and services to those in the academic, medical, cultural, and research fields. The customer has used CA Endevor SCM for over three years and its largest mainframe application comprises 4,000 programs with 80,000 related elements.

COBOL is their main programming language, with over 100,000 elements managed within their CA Endevor SCM installation. It has over 100 mainframe programmers using CA Endevor SCM, with only one CA Endevor SCM administrator, and the customer’s QA function mandates CA Endevor SCM as its only acceptance route for mainframe changes.

Business Challenges

The customer’s main business challenges prior to the adoption of CA Endevor SCM related to the inability of existing practices to meet audit requirements. In particular, the customer used a manual
process for managing source code, stating that “the change process was not repeatable” and that library management was poor, resulting in lost code or the wrong version being used.

Outcomes

The customer received the required level of internal controls and the ability to demonstrate compliance. CA Endevor SCM functionality and processes have improved audit feedback with a consistent, auditable process - tracking changes in terms of who changed what and when, down to individual lines of code.

Source code is also better protected. Staff sickness or turnover is less of a concern as code is no longer ‘hidden away’ in personal Partitioned Data Sets (PDSs). The customer also experienced a reduction in the number of people needed to migrate software into production, benefitting not only from automation per se but also specific features such as being able to set approvers by software or element types.

The customer experienced very few people challenges. Many programmers had used CA Endevor SCM with previous employers and so were au fait with both its operation and benefits. The benefits of automation and the move away from the tedium of the existing manual process were obvious to developers but the change was ultimately mandated from the top for compliance purposes.

This customer found a reduction in staff numbers (promotion to production), with speed of migration reduced from two hours to 15 minutes.

The customer benefited from productivity improvements and greater work satisfaction for the developers who were empowered to perform many tasks without needing to make requests to administrators. This customer found a reduction in staff members (promotion to production), with speed of migration reduced from two hours to 15 minutes. There was also a reduction in overall errors with the added benefit of having them identified earlier in the lifecycle resulting in net cost savings to the business. This customer has experienced a reduction in lost code due to CA Endevor SCM usage that ensures that code can only be proactively deleted and never lost, and that enables the organization to set varying authority levels for deletion approval. They also appreciate the increased levels of security and the ability to pass audit assessments with less effort – it now takes less time and a lot less manpower.

The customer was also keen to espouse the strengths of CA Endevor SCM. They consider it a very stable tool, and said that in-built processes are modeled on best practices. They praised the flexibility of, and control over, implementation, the high level of security, and noted that CA Technologies product support is very good, including CA Endevor SCM user groups and conventions for knowledge sharing.

Meeting the Challenges of Change Management

© Ovum (Published 03/2012)
Finally, this customer received the following most recent audit result: “Our auditors at PricewaterhouseCoopers completed our SAS 70 audit and deemed our internal controls to be both designed appropriately and operating effectively, by issuing an optimal ‘unqualified’ opinion”. The case study spokesperson said: “From an IT Controls perspective, CA Endevor SCM was instrumental in providing the foundation not only to force repeatable software development lifecycle processing, but it allowed the organization to insert process controls at key stages throughout the Software Development Lifecycle”.

Acknowledgements

Ovum is grateful to the spokespeople from the case studies for their contribution and review of this White Paper, and not least to CA Technologies for their input and assistance.

Appendix