G00274528

Market Guide for Software Change and Configuration Management Software

Published: 26 August 2015

Analyst(s): Jim Duggan, Joachim Herschmann

Organizations are investing in SCCM tools to increase the speed and quality of both the development of new and changed functionality for more complex environments and its delivery. We identify the market's attributes and direction to help IT leaders choose appropriate solutions.

Key Findings

- Open-source software change and configuration management (SCCM) offerings meet the functional requirements of many development teams.
- Commercial SCCM offerings offer improved scale, performance, security and integration.
- The advantages of adopting distributed version control system (DVCS) products are substantial for developers.
- Centralized management practices, evolved from central version control system (CVCS) products, or built specifically, are needed to match the freedom of DVCS practice to the discipline required by large-scale enterprise deployments.

Recommendations

- Managers of small teams should adopt DVCSs to increase the productivity of individuals.
- Managers in large organizations should coordinate the evolution of their downstream test and release processes at the same time as introducing DVCSs.
- Enterprises adopting DVCSs should maintain or enhance downstream test and release processes to maintain required levels of visibility and control.
- Enterprises with highly interactive distributed projects, significant central control needs, or complex processes and projects should establish parallel development, application life cycle management (ALM) and administration processes as part of any introduction of DVCSs.

Market Definition

This document was revised on 3 September 2015 and 9 September 2015. The document you are viewing is the corrected version. For more information, see the Corrections page on gartner.com.

SCCM tools are key enablers of software development teams. This category of tool encompasses earlier version control systems, source code management (SCM) systems, and the more generalized CVCSs and DVCSs that can version and manage broad classes of source, binary, metadata and change set objects. The SCCM software market is of interest to managers of teams responsible for application development, maintenance and release. Basic functionality enables core development, while more advanced offerings coordinate access and versioning across multiple streams and multiple stages of development.

This functionality is designed to orchestrate the creation, modification and management of software assets throughout their life cycle. The most basic products provide access control and versioning of individual files and objects. More elaborate products add metadata, change sets and functions to provide management for workflow, parallel access and file access for other parts of the life cycle. Importantly, these tools protect source code and other development assets from accidental or malicious modifications. They also enable teams to share access to code without damaging conflicts, while establishing traceability, accountability and separation of duties.

Table 1 highlights typical functions found in SCCM tools and identifies their principal users.

Table 1. Typical Functions of Software Change and Configuration Management Tools

Principal User	Basic Functionality	Intermediate Functionality	Advanced Functionality
Individual developer	Access control Versioning of artifacts	Branching Grouping of modules	Parallel development Build support Change sets Refactoring
Development team	Assignment and tracking of work items	Staging Approval	Coordination across multiple variations Reporting
Build and release staff	Workflow Scheduling	Logging Recovery	Coordination of parallel processes

Source: Gartner (August 2015)

Market Direction

Changes in how, where and by whom software development is done have caused substantial shifts in SCCM tool usage. Demand for support of global teams, and for distributed functions like those of

Git and Mercurial, has increased. Open-source projects and established vendors have responded by adding support for DVCSs, and by improving security, auditability and integration with other development resources.

We estimate that DVCS offerings like Git and Mercurial are used by about 30% of developers. Another 30% use Subversion or older open-source CVCS offerings. The seat share of commercial products, like those of Borland, IBM, Perforce and Serena Software, is now about 40% of the total, down from about 50% in 2008.

Among open-source products, use of Git has dominated. There are, however, early signs that Mercurial and other open-source DVCS projects are gaining popularity, particularly as Web-scale development teams explore alternatives to the Git design choices within DVCS architecture.

Although developers continue to shift to DVCS products for their desktops, centralized systems continue to be deployed in parallel to DVCSs in large enterprises where control, traceability and security are of great importance. In these situations, the CVCS acts as a hub for the rollup of managed code and objects from individual workstations.

Smaller enterprises and some workgroups are using cloud-based SCCM instances like GitHub and Microsoft's Visual Studio Online. Some of them are just provisioning departmental SCCM, while others are using the cloud to broaden their access to source code, test cases and other artifacts of the development process. Adoption of cloud-based SCCM by enterprises is still in its early stages, however — it amounts to less than 5% of the total seats. Many organizations are still reluctant to broaden its use without a better understanding of the security, audit and intellectual property (IP) issues. Large enterprises are likely to keep deploying on-premises or private cloud instances of SCCM until practices to manage the security, audit and IP implications of cloud-based SCCM are more clearly understood.

Market Analysis

Changes in SCCM practices have been driven by multiple factors:

- Adoption of agile practices, such as refactoring, branch per story and continuous integrations
- Greater need to support distributed teams
- Increases in the number of types of information to be versioned
- Requirements to support multiple clients and platforms
- The need to coordinate with the versioning of packaged software and mechanical systems

Under these pressures, CVCSs, both commercial and open-source, have evolved but have also begun to give way to DVCSs.

A DVCS is substantially different in architecture and implementation from centralized implementations of revision control, such as ClearCase, Microsoft Team Foundation Server and

Subversion. The benefits of DVCSs are local performance for the programmer, a different balance of scalability and more flexible workflows than are provided by a central check-in/check-out model. While any well-disciplined team can, in principle, be well-served by a DVCS, the distributed model found its earliest adoption in large, loosely coupled projects, such as the open-source community working on the Linux kernel. Developers can work independently, and can control what is submitted to the central store. Faster operations during development are offset by longer initial load times, and by process requirements for security and merge management.

In a DVCS, a full copy of the source code and version history can be on each participant's desktop. It is easier to create a new branch or variation, rapidly reflecting an individual's change actions, but this approach shifts network overhead and difference/merge overhead to a synchronized workflow stage. This separation enables different rules to be applied to the ways individuals change code, and to who has the power to merge code changes.

The DVCS model for source code version and configuration control has gained popularity with individual programmers because it gives them better performance, and makes the creation of new code branches simpler and quicker. In agile projects it provides a place to store changes that are being used by a small set of developers but that are not complete enough for the entire team (branch per story). This is particularly important for very large organizations and organizations with very high release velocities. The challenge for managers is to maintain the programmer benefits, while serving the team's and enterprise's needs for quality, visibility, security and consistency.

Dealing with a DVCS from a team perspective is more complex than dealing with a central system, and this complexity is at the root of most current barriers to broader use. When scaling the solution, enterprise groups have to deal with issues of security, process management, traceability and accountability. Organizations also speak of being uncomfortable with what is perceived as a larger possibility for loss of IP through misappropriation of source code. Strategies that worked with source access had previously been partitioned among programmers to reduce IP risks.

Adoption of DVCSs continues to accelerate within smaller teams with minimal requirements for centralized change or release processes. Barriers to coexistence with centralized solutions are being reduced as commercial SCCM vendors and ALM providers add aspects of DVCSs as special configurations, or as alternative back ends. This is usually done by improving or transforming how DVCS stores are merged and administered.

Commercial implementations of DVCSs with extensions for central administration, audit, traceability and accountability facilities will accelerate experimentation with DVCSs in IT settings. Examples of such DVCSs that have reached the market are GitHub Enterprise and Perforce Helix. Vendors' promotions of these new offerings will increase users' expectations, but it will take several years for development organizations to accommodate the optimistic branching models of DVCSs in organization standards originally developed around more restrictive centralized versioning. Enterprise-grade management of Git that offers important aspects of a DVCS — good merging, the ability to work offline and good collaboration — along with the security and central repository of a CVCS, will resolve most remaining concerns about the use of the DVCS model. Whether natively or in a hybrid with older styles, DVCS characteristics will be broadly adopted.

A further barrier to adoption of open-source DVCSs is the use of General Public Licenses (GPLs). This is an issue if company policy requires changes or extensions to the DVCS source code. Custom integrations to, or extensions of, the DVCS source would have to be licensed under GPL terms.

Open-source ALM projects focused on Eclipse and similar integrated development environments will try to accommodate DVCS and enterprise needs, but we expect the focus of DVCS open-source projects like Git and Mercurial to remain functionality for individual contributors. Open-source DVCS projects are likely to lag behind open-source ALM projects and commercial offerings with DVCS features in enterprise usage.

Representative Vendors

The vendors listed in this Market Guide do not imply an exhaustive list. This section is intended to provide more understanding of the market and its offerings.

Tables 2 and 3 list representative SCCM offerings. For these, we list the providing organization's ownership (public or private) or indicate that it is an open-source project; the product architecture (central or distributed); and whether the product is available for on-premises installation, or in single-tenant or multitenant hosted forms. Integration with Git repositories is indicated where other types of stores are available. We also show whether there is a branded change management or ALM product from the same provider.

Consult the vendors for details of functionality, integrations with other types of development tool, and the hosting environments supported.

Development assets can also be hosted on mainframes or midrange platforms using products from companies like CA Technologies, ISPW BenchMark Technologies, PTC, Rocket and Serena Software. Although not included in this Market Guide, these providers should also be evaluated by companies developing or supporting software on those platforms.

Table 2. Software Change and Configuration Management Offerings Available Under Commercial License

Offering	Vendor/Brand	Organizational Form	Architecture	On-Premises	Single-Tenant Hosted	Multitenant Hosted	Interface to Git	Associated ALM Offering
AWS CodeCommit	Amazon Web Services	Public	DVCS			X	X	
Bitbucket	Atlassian	Private	DVCS			X	X	X
Stash	Atlassian	Private	DVCS	X			X	X
Beanstalk	Wildbit	Private	CVCS, DVCS			X	X	
BitKeeper	BitKeeper	private	DVCS		X			
Harvest Software Change Manager	CA Technologies	Public	CVCS	X				
Plastic SCM	Codice Software	Private	CVCS, DVCS	X	X	X	X	
TeamForge SCM	Collabnet	Private	CVCS, DVCS	X	X		X	X
CloudForge	Collabnet	Private	CVCS, DVCS			X	X	X
SubversionEdge	Collabnet	Private	CVCS	X				X
GitHub Enterprise	GitHub	Private	DVCS	X			X	
GitHub.com	GitHub	Private	DVCS			X	NA	
GitLab	GitLab	Private	DVCS	X	X	X	X	
ClearCase	IBM	Public	CVCS	X				X

Offering	Vendor/Brand	Organizational Form	Architecture	On-Premises	Single-Tenant Hosted	Multitenant Hosted	Interface to Git	Associated ALM Offering
Synergy	IBM	Public	CVCS	X				X
Rational Team Concert (RTC)	IBM	Public	CVCS	X			X	X
AccuRev	Micro Focus/ Borland	Public	CVCS	X		X	X	X
GitCentric	Micro Focus/ Borland	Public	DVCS	X			X	
StarTeam	Micro Focus/ Borland	Public	CVCS	X		X	X	X
Team Foundation Server (TFS)	Microsoft	Public	CVCS	X	X		X	X
Visual Studio Online	Microsoft	Public	CVCS			X	X	X
Helix	Perforce	Private	CVCS, DVCS	X	X	X	X	
PTC Integrity Lifecycle Manager	PTC	Public	CVCS	X	X			X
SurroundSCM	SeaPine Software	Private	CVCS	X				
Dimensions CM	Serena	Private	CVCS, DVCS	X				X
PVCS Pro	Serena	Private	CVCS	X				X
Vault	SourceGear	Private	CVCS	X				

Offering	Vendor/Brand	Organizational Form	Architecture	On-Premises	Single-Tenant Hosted	Multitenant Hosted	Interface to Git	Associated ALM Offering
Razor	Visible Systems	Private	CVCS	X		X		
Access Control Plus	WANdisco	Public	CVCS, DVCS	X			X	
Git Access Control	WANdisco	Public	DVCS	X			X	
Git Clustering	WANdisco	Public	DVCS	X			X	
Git MultiSite	WANdisco	Public	DVCS	X			X	
SmartSVN	WANdisco	Public	CVCS	X				
SVN Client and Server	WANdisco	Public	CVCS	X				
SVN MultiSite Plus	WANdisco	Public	CVCS	X				

ALM = application life cycle management; CVCS = central version control system; DVCS = distributed version control system; NA = not applicable; SCCM = software change and configuration management

Source: Gartner (August 2015)

Table 3. Software Change and Configuration Management Offerings Available Under Open-Source License

Offering	Vendor/Brand	Organizational Form	Architecture	On-Premises	Single- Tenant Hosted	Multitenant Hosted	Interface to Git	Associated ALM
CVS	Open-source project	Open-source project	CVCS	X				
Git	Open-source project	Open-source project	DVCS	X			NA	
Mercurial	Open-source project	Open-source project	DVCS	X		X		
Subversion	Open-source project (Apache)	Open-source project	CVCS	X				
GitLab Community Edition (CE)	GitLab	Private	DVCS	X			X	

ALM = application life cycle management; CVCS = central version control system; DVCS = distributed version control system; NA = not applicable; SCCM = software change and configuration management

Source: Gartner (August 2015)

Market Recommendations

User Advice:

- Small teams with good process discipline and limited requirements for security and audit history should adopt DVCSs to increase the productivity of individuals.
- Large organizations should expect to have more difficulty introducing DVCSs, unless they carefully coordinate the evolution of their downstream test and release processes at the same time.
- Teams that need sophisticated change and release process management should evaluate DVCSs combined with a strong ALM function.
- Enterprises with highly interactive distributed projects, significant central control needs, or complex processes and projects should establish parallel development, ALM and administration processes as part of any introduction of DVCSs.
- All prospective buyers of DVCSs should understand that:
  - Although a DVCS alone may suffice on desktops, additional central management will be required for good release discipline.
  - Sophisticated change and release process management demands strong ALM functionality, even with a DVCS.
  - Architectural patterns such as service-oriented architecture and adoption of microservices reduce some of the risks of DVCS adoption.

Acronym Key and Glossary Terms

ALM	application life cycle management
CVCS	central version control system
DVCS	distributed version control system
GPL	General Public License
SCCM	software change and configuration management

Gartner Recommended Reading

Some documents may not be available as part of your current Gartner subscription.

"Magic Quadrant for Application Development Life Cycle Management"

"Distributed Version Control Systems (DVCS): Essential for Distributed Development Teams"

"IT Market Clock for Application Development, 2013"

"Hype Cycle for Application Development, 2015"

GARTNER HEADQUARTERS

Corporate Headquarters
56 Top Gallant Road
Stamford, CT 06902-7700
USA
+1 203 964 0096

Regional Headquarters
AUSTRALIA
BRAZIL
JAPAN
UNITED KINGDOM

For a complete list of worldwide locations,
visit http://www.gartner.com/technology/about.jsp