REMAINSOFTWARE

WHITEPAPER

WHAT SHOULD ALL SMALL DEVELOPMENT TEAMS KNOW ABOUT SOFTWARE CHANGE MANAGEMENT?

JUSTIFYING SOFTWARE CHANGE MANAGEMENT (SCM) SOLUTIONS FOR SMALL TO MEDIUM-SIZED DEVELOPMENT TEAMS

Development teams in small to medium-sized businesses (SMBs)—and small independent teams in larger organizations—often forgo SCM tools for cost reasons. However, tools designed for SMBs can be affordable and deliver a significant return on investment.

Executive Summary

Business, regulatory and technology change is inevitable and relentless. As businesses introduce new products, services and processes, applications may need to be modified to accommodate those changes. Likewise, code and configuration changes might be required to take advantage of new, more efficient technologies. And when governments introduce new regulations the IT department may have to modify the company's software to comply with them.

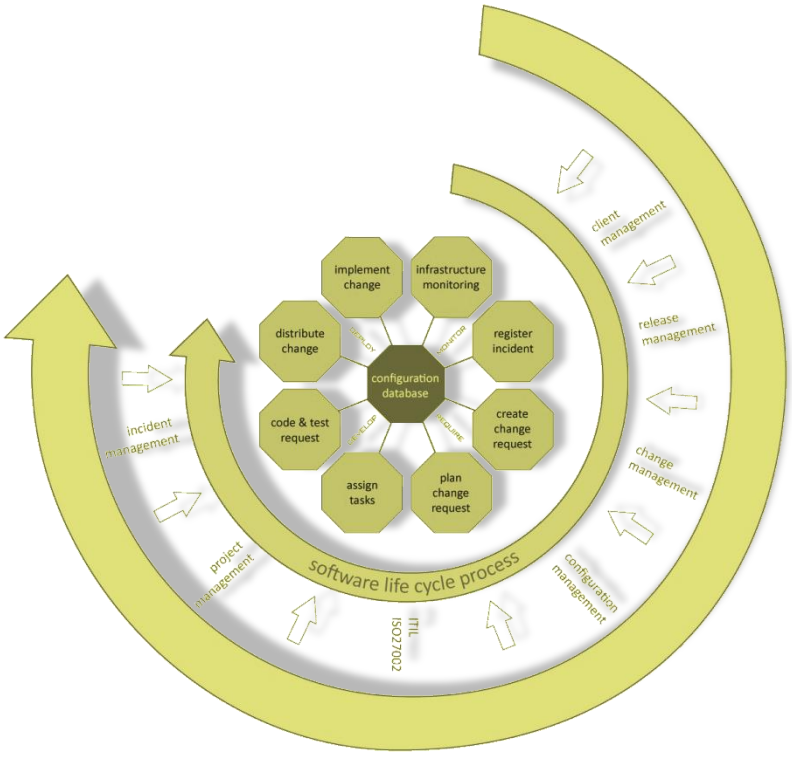
Accommodating these changes typically requires software modifications and/or development. Determining what new code has to be written or what existing code has to be changed is only part of the challenge. Because application modules are interrelated and depend on a variety of components, it's also essential to understand the relationship between the changed code and other components. That is true whether you're maintaining custom-developed software or upgrading a third-party application.

A Software Change Management (SCM) tool can automate the application asset discovery and documentation process, including the documentation of dependencies, thereby making the entire application lifecycle more efficient, while also improving the accuracy of software maintenance.

Nevertheless, a high entry cost can be a deterrent to buying an SCM tool in small and medium-sized businesses. Yet small organizations face much of the same application complexity and risk as large enterprises—and, in some ways, more. Thus, they can derive at least proportional value from SCM tools.

Small organizations face much of the same application complexity and risk as large enterprises—and, in some ways, more. Thus, they can derive at least proportional value from SCM tools.

This white paper examines these issues and presents an SCM solution designed for small to medium-sized development teams.



Application Lifecycle Issues and Risks

Most of your IT budget is likely not spent on new development. According to Forrester Research[1], only 34 percent of the typical IT budget goes to new development.

Fully 66 percent is spent on operations and maintenance. Consequently, improving productivity and accuracy in the maintenance phases of the application lifecycle can generate considerable value. As will be discussed in a later section, this is true even if you use third-party software.

Part of the problem is that, in business, standing still is falling behind because the rest of the world moves ever forward. A company that persists with outdated processes, products and technologies will quickly find itself at a significant competitive disadvantage and may not be in compliance with new and modified regulations.

Unlike in the early days of computing, modern applications are generally not large, monolithic programs with only internal subroutines. In aggregate, modern applications may be enormous—typically with many more lines of code than their early predecessors—but today they are usually composed of myriad application objects.

When an organization wants to alter a business process or add a new one, finding the affected objects can be difficult, particularly if the code is not well documented and the original developers are not available.

But that's only part of the challenge. Those modules interact in ways that may not be obvious. Consequently, a perfectly coded

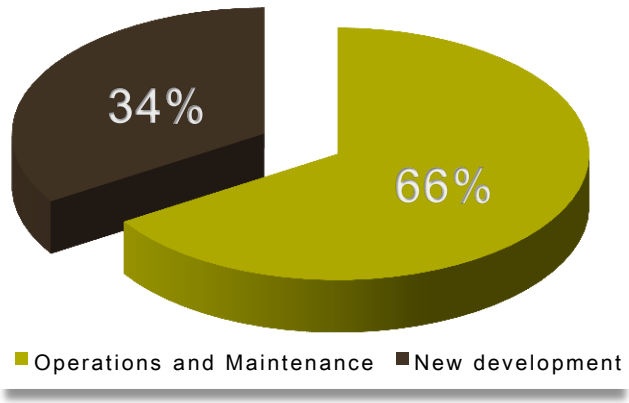
Improving productivity and accuracy in the maintenance phases of the application lifecycle can generate considerable value.

modification of one object may cause another object to fail unexpectedly.

In addition to being componentized, modern applications are typically multi-tiered, with at least database, application and user-interface layers. If changes are not coordinated across all tiers, the application may crash.

This complexity often stymies efforts to give applications a contemporary look and feel that users are familiar with from their use of other business and personal software and that meets modern standards for application design. The problem is that developers have to spend so much time unraveling the application components and interactions when fixing bugs and addressing critical changes that they don't have any time or resources left over for application modernization.

IT Budgets



Source: Forrester research

[1] *Application Modernization and Migration Trends in 2009/2010*, Forrester Consulting, November 2, 2009

Packaged Problems

Some companies fill most of their application portfolio with purchased software. They may believe that, because they didn't develop their applications they don't need SCM tools. After all, the thinking goes, the vendor is responsible for maintenance and upgrades, not us. That is true up to a point, but only up to a point.

The problem is that your organization likely depends on components external to the software package. For example, a special query that is critical to operations may have been written long ago and forgotten by the IT department, but not by the users who utterly depend on it. Or Excel spreadsheets might be populated automatically by data maintained by the application. If an upgrade changes the application's database structure, those queries and Excel spreadsheets may stop working.

In addition, the application might interact with modules from other vendors or with the few custom-developed modules that your organization has written. To ensure that *all* of the software your organization relies on will continue to work after an upgrade you need thorough visibility into those interactions.

Software and database interactions are only a few of the issues that can cause grief when upgrading packaged software. The application may also have hardware and other resource dependencies that are mostly invisible during day-to-day operations. A change to the application may make it necessary to reconfigure those resources. Likewise, changing the technology may require new versions of your application packages.

By automating the inventory of all application assets and the interactions between them, an effective SCM tool can help you to avoid these challenges.

Software and database interactions are only a few of the issues that can cause grief when upgrading packaged software. The application may also have hardware and other resource dependencies that are mostly invisible during day-to-day operations.

Small as a Special Case

When presented with the case for SCM tools, small to medium-sized companies—and small, independent teams in large enterprises—may say, "that doesn't apply to us; we're too small." They argue that because their team consists of just one or two developers, or at most only a few, communication issues inherent in large teams aren't a problem for them. Besides, they might argue, "because we have complete responsibility for the whole application portfolio, we know it inside and out."

The potential challenges with that perspective are at least two-fold.

First, applications in small organizations are often as complex as applications in large enterprises. It's impossible for anyone to remember all of the details of all of the components and resources and, more importantly, the relationships and dependencies within and among those applications, components and resources. Yet, if one of those items is inadvertently ignored then, at best, the software change process may take longer than expected due to the need to track down

problems. Worse, the application may fail in production.

Second, some issues that are potentially very grave for small teams may be only minor irritants for larger organizations. A large enterprise might have an IT staff numbering in the hundreds or even thousands. Several people may share responsibility for each application. If one person isn't available, someone else probably has the same knowledge of the application as that missing person and may be able to fill in without missing a beat.

That's not true in small IT shops. Only one or two people may manage and maintain an application, or possibly the entire application portfolio. Often, they are so intimately familiar with the application(s) that documentation on the original code, changes applied over the years, module and resource dependencies and configuration settings is considered to be unnecessary. After all, that information is stored in that person's head, available whenever it is needed. At least, so the thinking goes.

However, that's not always true in the real world. If a key person quits, takes a vacation, is incapacitated or worse, that information leaves with the person. Unearthing all of the application code, data tables and fields, interactions and dependencies affected by a change then becomes a Herculean task.

These challenges are amplified when a small team is augmented by part-time developers or, in the extreme, when there is no team at all and all of the work is performed by consultants who are contracted on only an as needed basis. These contracted developers may have may have little or no experience with the organization's applications to draw on. Thus, unless adequate documentation and software change management tools are available, they may have to build that knowledge from scratch before they can begin to work on an application.

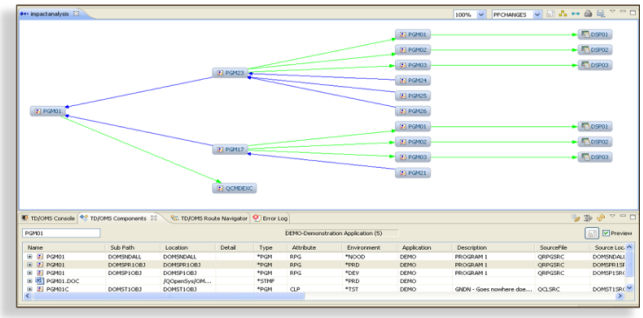
SCM tool designed specifically for small to medium-sized organizations, with a pricing model that suits their budgets, can deliver a significant return on investment.

Neither of these issues comes as a surprise to small and medium-sized organizations, but there is often still considerable resistance to adopting SCM tools. The reason is that there is another difference between them and their larger counterparts: the size of their budgets. Managers often recognize the problems, but they feel that their budgets don't allow them to afford the solution.

However, an SCM tool designed specifically for small to medium-sized organizations, with a pricing model that suits their budgets, can deliver a significant return on investment.

SCM Solutions for Small Teams

A comprehensive SCM tool identifies all application objects and artifacts—and the relationships between them—and catalogs that information in a database. This gives developers ready insight into what will be affected by a proposed change.



Software Change Management - Impact analysis

An SCM tool also archives all changed code, allowing developers to compare the current code to previous versions to determine when a bug crept in. Developers can also use this archive to revert to a previous version if necessary.

Automated documentation is another benefit of SCM tools. All objects, relationships and changes to them are automatically cataloged. This information no longer resides solely in people's heads, nor is it scattered in electronic versions of scraps of paper—spreadsheets and unconnected documents. Instead, it's in a central repository that can be accessed rapidly and methodically.

In some cases, these capabilities are more than just best practices. They are legal requirements. As regulations tighten around the world, particularly in industries such as the financial sector, all IT components, including the changes made to them, must be comprehensively documented and readily auditable. Information stored in people's heads or haphazardly documented in unconnected files may not be sufficient for regulatory compliance.

The use of SCM tools also helps to make the change process more consistent and, most importantly, accurate. By automating configuration management, an SCM tool can ensure that all configuration requirements are addressed and all relevant components are installed in the correct order. And the workflow management facilities of an SCM tool can help to ensure that change processes consistently adhere to best practices, no important steps are inadvertently missed, and no critical issues "fall through the cracks."

Software bloat that accumulates over time is as much a challenge for small shops as it is for large ones. Users frequently ask for new programs, but they usually forget to tell IT about programs they no longer use. As a result, when users ask for a software change, already overworked IT staff may spend considerable time updating a related module that is no longer used. An SCM tool that catalogs application components and their usage allows developers to spot obsolete programs.

The use of SCM tools helps to make the change process more consistent and, most importantly, accurate. By automating configuration management, an SCM tool can ensure that all configuration requirements are addressed and all relevant components are installed in the correct order.

An SCM solution can also provide significant value when it comes time to modernize legacy applications. By providing greater insight into application assets and their dependencies, an SCM tool can reduce the complexity of software. As a result, rather than being stuck in constant maintenance and bug-fixing mode due to inefficient and ineffective workflows and deficient information about application assets, development teams can carve out time and space to pursue new development.

All of the above SCM functionality can provide significant value to small teams. The difference between small and large IT teams is not primarily the scope of their SCM requirements, but rather the scale. Small teams don't need to support complex communications among myriad people. They also likely support a smaller application portfolio than large enterprises.

An SCM tool designed for small to medium-size enterprises recognizes these differences in scale. It might, for example, contractually or structurally limit the number of applications that can be managed with the tool in return for a lower cost. Or it might support only a limited number of total or simultaneous users, a limitation that likely will be irrelevant in a small to medium-sized shop.

Another issue for small organizations is the upfront cost of the software. Small, recurring costs might be easily born by a small organization, particularly if the organization achieves efficiencies that more than cover the cost. A large upfront licensing fee, however, can be a barrier to entry for a small firm. Consequently an SCM tool that is priced using a per-seat, subscription model is often a better fit for a small to medium-sized organization.

An SCM tool designed and priced for small teams provides SMBs with the advantages that larger enterprises receive from application lifecycle solutions, thereby helping to level the playing field and supporting the nimbleness of smaller competitors. These advantages include comprehensive, automated cataloging of application artifacts and relationships; workflow management; automated tracking of code changes; enhanced software versioning; administration of consistent best practices; and, overall, greater insight into applications and object relationships.

BENEFITS OF SCM SOLUTIONS

- *Improve visibility into application assets and the relationships between them.*
- *Ensure that application knowledge remains available when employees leave.*
- *Reduce the possibility of creating bugs due to unknown relationships and dependencies.*
- *Make it easier to track down bugs.*
- *Provide an easy way to revert to an earlier software version if necessary.*
- *Enforce consistent maintenance and development best practices.*
- *Improve and automate workflows.*
- *Reduce the complexity of software.*
- *Improve the auditability of software.*
- *Increase the productivity of the entire application lifecycle.*

About TD/OMS Compact

TD/OMS Compact is a comprehensive SCM tool designed specifically for small to medium-sized teams. Sold on a subscription basis, with no up-front licensing costs, it is affordable for even the tightest of budgets.

TD/OMS Compact is designed for organizations developing in RPG, RPG ILE and/or Java who are looking for simplicity and power in Software Change Management. This compact solution delivers all of the functionality you need for effective Software Change Management, such as incident, configuration and version management, including the following:

- **Repository** that stores all of the software configuration and change process definitions,
- Full **Graphical User Interface**,
- **Work Management** to organize and take control over your work, issues and changes made to your applications,
- **Graphical Impact Analysis**, showing all object relations in a clear, graphical way.
- **Pre-defined set-ups** make it easy for small teams to get started quickly.

About Remain Software

Established in 1992, Remain Software is an agile independent software vendor that delivers innovative solutions for the management of the entire application lifecycle, from defining requirements through design, development and up to deployment and testing.

Remain Software's customer-focused solutions help organizations to simplify and automate processes, improve workflows and teamwork, and streamline IBM i, Windows, UNIX and Linux software development. Simplified and standardized Application Lifecycle Management, time and cost savings, and improved productivity and communication within teams are just some of the benefits that help to deliver high quality applications and customer satisfaction. Taken together, these features and benefits serve to increase organizations' profitability.

Remain Software is supported by an extensive Partner Network. Together with our Value Added Resellers, we offer a broad range of services and training that maximize the benefits of our solutions.

CONTACT

REMAIN SOFTWARE

Remain B.V.
Dukatenburg 82b
3437 AE Nieuwegein

Tel: (+31)30-6005010
Fax: (+31)30-6005019

info@remainsoftware.com
www.remainsoftware.com