



Software Configuration Management for LabVIEW™ Programming Environments

by Robert Jacobs

As LabVIEW applications become increasingly more complex, developers are confronted with the challenges of not only delivering “better, faster, cheaper” solutions, but also adhering to process and quality standards developed for the mainstream programming languages. Even more stringent in the regulated industries, like life sciences or DOD, these standards are often dictated by management with an eye on the bottom line. But, regardless of the financial benefits, these standards offer programmers some real life advantages such as less rework, easier change management and more time for everything else (*because the last project was completed on schedule*).

Configuration management (CM) is only one of the professional software engineering practices advocated by the leading industry experts. However, CM is one of the easiest practices to implement, once programmers are convinced of its usefulness and provided with the necessary tools. This article will define CM, and describe why both management and programmers alike should adopt a CM strategy.

Many public and private organizations have devoted considerable time and resources to document the benefits of CM. In the *NASA Software Configuration Management Guidebook* (Software Assurance Technology Center, 1995), Lawrence E. Hyatt, Manager of NASA's Software Assurance Technology Center, states “Unmanaged change is very possibly the largest single cause of failure to deliver systems on time and within budget.” It is, therefore, easy to see how even the most diligent change management practices can become ineffective without proper attention to configuration management.

An example of the importance of CM can be seen in this all too familiar scenario. In 2001, a supplier to a major aircraft manufacturer certified that its control motors had passed a series of rigorous quality and performance tests. Those tests were developed in LabVIEW by a fully qualified and highly capable test and measurement team. Unfortunately, when the test programs were compiled, incorrect component versions were included in the final build. After delivery and installation, the product flaws were discovered, but the original supplier was forced to restart the entire project at a substantial loss of revenue.

The benefits of CM have been well documented, but graphical programming languages such as LabVIEW, pose some difficulties that are not adequately addressed by most configuration

management tools available. These difficulties are rooted in the proprietary binary format of VIs and the method LabVIEW uses to load files. For that reason, many LabVIEW testing and automation groups have ignored the obvious advantages of a formal CM process because the most common CM tools are not tailored for their needs. It is important to note that a successful CM strategy is a combination of effective CM tools and a structured process that is supported at all levels in the organization.

Configuration Management and CMM/CMMI

One of the leading sources of information on best practices for software development, including configuration management, is the Software Engineering Institute (SEI)SM of Carnegie Mellon University, creators of the Capability Maturity Model (CMM)[®] and its newest incarnation, Capability Maturity Model Integration (CMMI)[®]. Configuration management is only one of 24 process areas within CMMI, however it is one of seven fundamental areas required to progress upwards through its various levels. See *CMM and CMMI – An Overview* on page 21 or for more information about CMM or CMMI, visit www.sei.cmu.edu.

The progress most companies have made toward developing a well-defined process for software development activity has been driven by management directive. While the SEI commonly rates organizations as to their CMM/CMMI level, there are very few LabVIEW-based programming organizations, including integrators, who have risen above the third level of the CMM/CMMI rating scale. One of the principle reasons is that the tools needed to automate version tracking and revision control have had underlying issues that make them time-consuming, costly, and eventually ineffective. These issues include a lack of effective integration into the LabVIEW development environment for items such as the check-in/out procedure, the mismatch of hierarchical code status reporting with the hierarchical structure of LabVIEW source code, and a lack of synchronization between VI documentation (for example, VI history and version number), and source code control system-maintained information.

Once the necessary commitments to configuration management policy have been made, many organizations can move from CMM/CMMI Level 1, a state of chaotic development, to Level 2 or 3. This is an important step, as studies by SEI show *continues on page 22*



CMM and CMMI – An Overview

Capability Maturity Model

The Capability Maturity Model (CMM) from the Software Engineering Institute (SEI) of Carnegie Mellon University is a framework for process improvement. This commonly known model helps to define where an organization stands and, more importantly, outlines the focus areas necessary for advancing process improvement. In addition, many major public and private organizations now require compliance with one of the various CMM model levels as a basis for doing business. CMM is divided into five levels, with each level representing achievable goals in a continuous improvement strategy.

The following descriptions of each CMM level are taken directly from the SEI web site at www.sei.cmu.edu:

Level 1 – Initial. The software process is characterized as ad hoc, and occasionally even chaotic. Few processes are defined, and success depends on individual effort and heroics.

Level 2 – Repeatable. Basic project management processes are established to track cost, schedule, and functionality. The necessary process discipline is in place to repeat earlier successes on projects with similar applications.

Level 3 – Defined. The software process for both management and engineering activities is documented, standardized, and integrated into a standard software process for the organization. All projects use an approved, tailored version of the organization's standard software process for developing and maintaining software.

Level 4 – Managed. Detailed measures of the software process and product quality are collected. Both the software process and products are quantitatively understood and controlled.

Level 5 – Optimizing. Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.

Capability Maturity Model Integration

As CMM gained acceptance, it became apparent that one model would not be effective for all types of organizational structures and purposes. The purpose of CMM Integration (CMMI) is to provide guidance for improving an organization's processes and its ability to manage the development, acquisition, and maintenance of products or services. CMMI places proven approaches into a structure that helps companies appraise its organizational maturity or process area capability, establish priorities for improvement, and implement these improvements. CMMI has two representations: staged and continuous. The continuous representation uses capability levels to measure process improvement, while the staged representation uses maturity levels.

Continuous Representation Capability Levels

Capability levels, which belong to the continuous representation, apply to an organization's process-improvement achievement for each process area. There are six capability levels, numbered 0 through 5. Each capability level corresponds to a generic goal

and a set of generic and specific practices. The following is excerpted from an SEI white paper, "Capability Maturity Model® Integration (CMMISM), Version 1.1" (Carnegie Mellon Software Engineering Institute, 2002):

0 Incomplete – An incomplete process is a process that is either not performed or partially performed. One or more of the specific goals of the process area is not satisfied.

1 Performed – A performed process satisfies the specific goals of the process area. It supports and enables the work needed to produce identified output using identified inputs. A performed process satisfies all of the specific goals of the process area.

2 Managed – A managed process is a performed (capability Level 1) process that is planned and executed in accordance with an established policy, employs skilled people utilizing adequate resources, is monitored, controlled, and evaluated for adherence to its process description.

3 Defined – At the defined capability level, the organization is interested in deploying standard processes that are proven and therefore take less time and money than continually writing and deploying new processes. A defined process is described in more detail and performed more rigorously than a managed process.

4 Quantitatively Managed – A quantitatively managed process is a defined (capability Level 3) process that is controlled using statistical and other quantitative techniques. Quantitative objectives for quality and process performance are established and used as criteria in managing the process.

5 Optimizing – An optimizing process is a quantitatively managed (capability Level 4) process that is changed and adapted to meet relevant current and projected business objectives. An optimizing process focuses on continually improving the process performance through both incremental and innovative technological improvements.

Staged Representation Maturity Levels

Maturity levels, which belong to the staged representation, apply to an organization's overall maturity. There are five maturity levels, 1) Initial, 2) Managed, 3) Defined, 4) Quantitatively Managed, and 5) Optimizing. Each maturity level comprises a predefined set of process areas. The continuous representation has more specific practices than the staged representation because the continuous representation has two types of specific practices, base and advanced, whereas the staged representation has only one type of specific practice. In the continuous representation, generic practices exist for capability Levels 1 through 5, whereas, in the staged representation, only the generic practices from capability Levels 2 and 3 appear; there are no generic practices from capability Levels 1, 4, and 5. Whether used for process improvement or appraisals, both representations are designed to offer essentially equivalent results.



Software Configuration Management for LabVIEW Programming Environments *continued from page 20*

Maturity Level	Calendar Months	Effort (Work Months)	Defects Found*	Defects Shipped	Total Cost
1	29.8	593.5	1348	61	\$5,440,000
2	18.5	143.0	328	12	\$1,311,000
3	15.2	79.5	182	7	\$728,000
4	12.5	42.8	97	5	\$392,000
5	9.0	16.0	37	1	\$146,000

*Defects found during development process

Table 1: Observed CMM Benefits

this speeds time to market by 38%, eases the workload by 76%, and improves product quality by 80% (in terms of number of defects shipped). The cost at Level 2 is only 25% of the costs at Level 1 as seen in *Table 1*. A fully mature process results in a 98% cost savings for similar projects.

Implementing Configuration Management

Implementing CM is often a wide-ranging and complex task, but the advantages make it a worthy mission. It requires self-assessment, clear vision, planning, adequate funding, and cooperation at all managerial levels, and may often include the test development organization's external customer base.

In both the CMM and CMMI models, Level 1 represents an ad hoc development process that leads to the creation of a software product, but is not repeatable. Configuration management is normally not utilized or managed within the development process for a Level 1 organization. Therefore this section describes the steps needed to achieve Levels 2 through 5 pertaining to the area of configuration management.

The first step in implementing CM, as part of a drive to achieve Level 2, is identification of configuration items. Configuration items may consist of VIs, support files (.mnu, .ini, .dll), documentation, and requirements. Once the configuration items are identified, organizations need to store the configuration items in a configuration management system. The configuration management system is responsible for tracking and controlling changes to configuration items in a central repository.

An example of manual configuration management can be as simple as documenting changes and saving the new file with an incremental version suffix. This type of system is not likely to be 100% successful and requires a significant amount of time and

discipline. A more reliable approach is to make an investment in a Commercial Off The Shelf (COTS) configuration management system.

Examples of COTS configuration management systems are Visual Source Safe by Microsoft®, ClearCase® by IBM®, PVCS® Dimensions™ by Merant®, and CM Synergy by Telelogic. These tools vary greatly in price and features. The majority of the features are designed around traditional script-based programming languages such as C, C++, VB, J2EE and others. Because LabVIEW VIs are source and executable code in a single binary file, developers are unable to utilize many of the standard configuration management system features. Many of these systems are not integrated into the LabVIEW development environment directly which can lead to VI overwrites as a result of name conflicts and VIs still resident in memory. If a VI overwrite occurs, it signifies a complete breakdown of the configuration management system.

The final step in implementing Level 2 CM is performing configuration audits. The role of the audits is to ensure the integrity of the items in the configuration management system and the baselines deployed. If a discrepancy is found, it may lead to a significant cost and time allocation to find the source and affected files. Therefore it is important for LabVIEW developers to choose a system that will work best to avoid these issues.

In order to move beyond Level 2, an organization needs to continually review, plan, and refine the software development process. Level 3 is characterized by organizations with a training program for the process, a well-defined configuration management system that is used on every project, peer reviews, and inter-group cooperation. Organizations that are Levels 4 and 5 have very tight control of all software development and release activities through use of templates, documentation reviews and approval, and even defect prevention screening as well as other additions to the process. There would be quantitative measures in place for all aspects of the entire software development process. These quantitative measures would be used in periodic reviews. Examples include items such as number of defects detected in each development process step and development effort versus application size. Because CMMI is a continuous improvement model, alterations to the CM strategy and software development process are normal and expected.

Conclusion

Reducing costs, decreasing time to market, and improving quality are the key benefits of CM. The commitment to CM and CMMI will result in a unique solution for every organization that undertakes it, and the rewards can make achieving the corporate goals of "better, faster, and cheaper" a reality.



For an organization to implement a software development process improvement plan, it is important to establish a vision team responsible for performing internal audits of the process and developing alterations. Having an outside firm perform audits or offer guidance is another important step in maximizing improvement efforts.

CM tools, such as the VISTA Configuration Management Tool Suite, offer LabVIEW programmers an easy method for maintaining the type of code consistency required for advancing to the next CMM level. The VISTA CM Tool Suite from VI Engineering is designed specifically for LabVIEW users and solves many of the usual configuration management issues by serving as an interface between many of the COTS configuration management systems and LabVIEW.

The following features are part of the VISTA Configuration Management Tool Suite:

- Base SCC Integration – LabVIEW integration with a 3rd party configuration management system (currently supports Visual SourceSafe and ClearCase).
- Review and Verification Tool – Report status and save configurations of VIs and projects.

- Build Tool – Create packages of VIs, sequences and support files.
- System Integrity Tool – Define a system configuration and monitor for unauthorized changes.
- CM Plan Template – Includes reuse library management, project organization, file status tracking, project labeling, change control, and configuration description.

For more information about the VISTA Configuration Management Tool Suite visit www.viengineering.com/Vista/VistaTools/CM_Tools/Index.asp.



About the author:

Robert Jacobs is President of VI Engineering, a leading test and measurement systems supplier and Select Integrator in the National Instruments Alliance program. Mr. Jacobs and Chief Technology Officer Stanley Case have introduced many popular tools for LabVIEW productivity, configuration management, and process assessment.

¹ Kitson D.H. Kitson and S. Masters, An Analysis of SEI Software Process Assessment Results: 1987-1991, Software Engineering Institute, CMU/SEI-92-TR-24, July 1992.)

© Copyright 2004 LTR Publishing, Inc. All rights reserved.

LabVIEW Technical Resource is an independently produced publication of LTR Publishing, Inc. LabVIEW is a registered trademark of National Instruments Corporation.

You have just viewed an article from the LabVIEW Technical Resource. To purchase this LTR issue with accompanying VIs or a 1 year subscription, click here.