

Activate your FREE member

ADVERTISEMENT

GET \$50 OFF A JOB POSTING
 And find your next star employee for less.

Post a Job

SearchSoftwareQuality.com
 Helping you develop, deploy and manage quality software.

- NEWS
- WHITE PAPERS
- WEBCASTS
- GLOSSARY
- DOWNLOADS
- EXPERTS

- MODELS/METHODOLOGIES
- REQUIREMENTS
- TESTING AND QA
- PROJECT MANAGEMENT

[Advanced](#) | [Site Index](#)
Search Pt

ADVERTISEMENT **Project Acceleration Kit** Download a free version & fast forward your project lifecycle to market
www.Microsoft.com

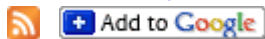
[Home](#) > [Software Quality News](#) > How to build better software using lean manufacturing, continuous quality improvement practices

Software Quality News:

How to build better software using lean manufacturing, continuous quality improvement practices

By Mike Taylor
27 Jun 2007 | SearchSoftwareQuality.com

RSS FEEDS: [Software quality news and advice](#)



Mike Taylor

Creating high-quality software is an interesting mix of art and science. At least 20 years ago innovative leaders like Brad Cox and Tom Love (inventors of Objective C and co-founders of StepStone and Productivity Products International) began describing a "software industrial revolution" in which the process of creating software would move from an art done manually by skilled craftsmen to an "industrialized" process that allowed high-quality systems to be built from well-tested reusable parts. This dream remains largely unfulfilled.

In 2007, Accenture CTO Don Rippert described "Industrialized Software Development" as one of eight major trends that Accenture has identified as likely to have major impact on IT over the next five years¹.

The Industrial Revolution in the world of "hard" manufacturing started in the early 1800s and was booming along in the early 1900s when Henry Ford tapped its principles to build the first mass-produced automobile. Amazingly, the world of software has yet to fully adopt the level of automation enjoyed by those who create complex, highly reliable hard goods like televisions, refrigerators and even the powerful computer chips that rely on programming to function.

Does the realm of large-scale, highly automated manufacturing have anything to offer to sophisticated software developers? In a word, yes. This article reaches into a specific area of modern manufacturing called lean manufacturing to find quality

improvement processes that, when combined with modern software technology, can add tremendous value to the creation of software.

What is needed is a continuous, collaborative, highly automated method of locating and correcting software defects early in the software development process.

Traditionally, approaches for improving software code quality rely on manual processes to locate and correct software defects. These techniques include manual code reviews, manually developed test cases and similar approaches. (Note that in this article we are addressing *code* quality and do not focus extensively on [requirements](#), design and other topics that also affect the overall success of a software system.)

These manual techniques are time-consuming as well as tedious and, in spite of good intentions, are often not well implemented. What is needed is a continuous, collaborative, highly automated method of locating and correcting software defects early in the software development process. This method must be effective, consistent and above all, *it must not add to the existing workload of the developer.*

Evolution of software methodologies

In recent years, software development technology has advanced to the point where code quality can be automatically built into systems while the software is being written. Simultaneously, the use of industry-accepted best practices can be monitored and enforced, all without major impact on programmer workload. Many of the successful methods used in manufacturing have been adapted to software development. Techniques used in manufacturing have inspired the successful [Agile software development](#) methodologies such as [Scrum](#) and [Extreme Programming \(XP\)](#).

[Alistair Cockburn](#), one of the authors of the Agile Development Manifesto and an expert in software development methodologies states², "Software engineering is remarkably similar to manufacturing once we notice decisions as the product that moves through a network of people. There is a very real parallel between design and manufacturing."

Improving software quality with continuous automated quality improvement and continuous automated code analysis

Using techniques we define as *Continuous Automated Quality Improvement (CAQI)* and a previously defined subset of CAQI called *Continuous Collaborative Code Analysis (C³A)*³, we will describe a highly automated, low-developer-impact approach that leverages modern technology and draws from proven tenets of successful large-scale manufacturing.

The basic tenets of CAQI are as follows:

1. The quality improvement process should start from the moment the first developer on a team begins to write code and continue throughout the life of the software system being created.
2. Every developer is responsible for improving software code quality.
3. Improving software code quality is an "every developer, every day" activity.
4. The software development team is united around a common set of quality standards, best practices, metrics and processes. The team constantly collaborates on the contents of the common set of goals and how they are used.
5. Software code quality improvement and measurement should be "ruthlessly

automated" so that it happens continuously and almost transparently for each developer as he creates code and doesn't adversely impact his workload and primary task of creating application logic.

It's pretty clear that the software development processes of many companies don't implement these tenets and can't meet the needs of developing quality software code in today's fast-paced world. Sketchy requirements are the norm. Programming teams have limited and often non-measurable quality objectives. Lack of communication causes bottlenecks and inefficiencies in the process of software development as noted by Cockburn. Use of good software design practices is haphazard. The product is handcrafted by trained specialists and automation is minimal. The code moves through the construction process (think waterfall development model) with various specialists bolting on their contributions.

Finally, it ends up in the Quality Assurance (QA) department where another group of specialists attempts to find all the errors, problems and places where the software doesn't meet the specification or will break in the hands of the end user. When issues are found, they are "thrown over the wall" to the developer for correction before the software is returned to the QA process and eventually shipped to the customer.

More information on software quality improvement

[Addressing software quality issues with development models, methods](#)

[How project management methods can improve software](#)

[The challenge of building quality into the development life cycle](#)

Having never encountered a software developer, manager or organization that didn't voice a desire to build quality software, it's surprising, that such good intentions haven't led to better quality when the rubber meets the road (or the software meets the customer). Outdated methods of software development are still being used in many organizations even after decades of intellectual and practical development of better, faster, higher-quality software development processes have been developed by institutions like Carnegie Mellon's Software Engineering Institute (SEI), commercial companies and others.

Enter lean manufacturing -- improving product quality

Meanwhile, "hard" manufacturing companies around the world have adopted a thoroughly modern manufacturing model called lean manufacturing to eliminate waste and improve product quality. The goal of lean manufacturing is to create the highest quality product in the shortest period of time at the lowest cost possible. Anyone that develops major software systems can learn a lot from the success of lean manufacturing processes.

The Toyota Production System (TPS) used in Toyota manufacturing plants is an example of a very successful implementation of lean manufacturing. Some of the main principles used to reduce waste and improve quality in the Toyota Production System include the following:

- *Hoshin* (translation: setting goals): Set goals with measurable targets to achieve business priorities.
- *Jidoka* (translation: automation with a human touch): This is the ability to detect an error, stop a production line if any issue is found that could affect quality, fix the error, identify the root cause and put in place a countermeasure. Jidoka makes it possible to build in quality during the production process.
- *Kaizen* (translation: continuous improvement): A system of continuous improvement in which instances of waste (Muda) are eliminated one by one at minimal cost. This is performed by all employees rather than specialists.
- *Jishuken* (translation: self study): This involves workers or teams identifying areas in need of continuous improvement and spreading information through the organization to stimulate Kaizen activity.

How CAQI applies to TPS principles

Continuous Automated Quality Improvement (CAQI) contains principles that incorporate the tenets of the TPS system. The CAQI principles are as follows:

- *Definitive Standards and Metrics* (applies to Hoshin): Define concrete software quality standards and metrics for an organization, team or project. The standards and metrics should enforce and measure elements of quality that are meaningful.
- *Make Everyone Responsible for Quality* (applies to Jidoka): Every software developer is responsible for quality and for driving out defects.
- *Immediately Locate and Fix Software Defects* (applies to Kaizen): Every developer is responsible for improving software quality every day. Locate software defects as code is written. Quality is NOT left to the QA department.
- *Continuous, automated quality improvement* (applies to Jishuken): Everyone on the team works together to find and share software quality issues and solutions using a technologically supported set of tools and processes that provide feedback to the developer as code is being written to help determine if it meets coding standards.

Building quality into software

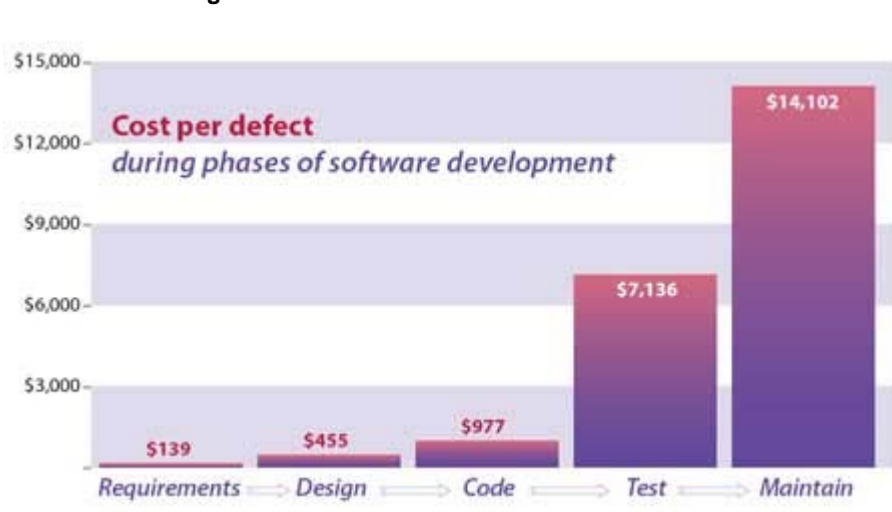
The lean manufacturing and TPS techniques can be successfully applied to the software development process to help developers and their organizations find and fix software defects earlier and continuously during the software development cycle.

Software defects drastically increase the cost of developing software. A study by Watts Humphrey, Carnegie Mellon, found, "On average, professional coders make 100 to 150 errors in every thousand lines of code they write." With an estimated 2.35 million software developers in the U.S. who are writing an average of 10 lines of code per day, this can result in 233,750 errors generated in an eight-hour day.

A study by Barry Boehm and Victor Basili⁴ found that the cost of correcting a software defect was approximately \$139 during the software requirements phase, \$455 in the design phase, \$977 during the coding phase, \$7,136 during testing, and \$14,102 in the maintenance phase after the software has been released.

In addition, there is the incalculable cost to the customer due to software delays, software that does not work as intended and software bugs, with corresponding negative impact on the image and credibility of the organization that created the software.

Costs of Correcting Defects

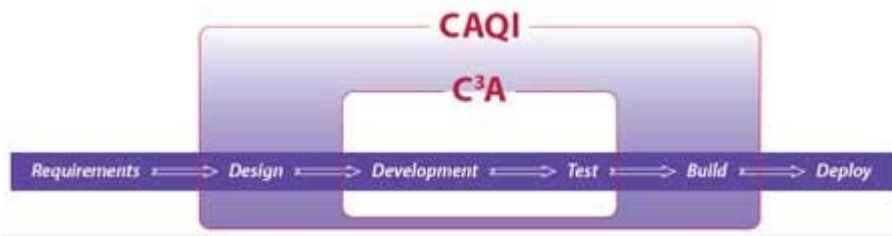


Source: B Boehm and V Basili, "Software Defect Reduction Top 10 List," *IEEE Computer*, January 2001

Boehm observed in 1987, "Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the design and requirements phase." It is imperative that quality be incorporated into all phases of software development and should be automated as much as possible to save time, locate defects early in the process and prevent rework.

Developing quality software using CAQI principles

The goal of CAQI is to create a collaborative, automated environment that involves the entire software organization, both management and developers, in creating quality software during the implementation phase of the software development life cycle. CAQI is used to instill a quality mindset in the entire technical team so that quality is the responsibility of all developers and is not viewed as only the responsibility of the QA or testing teams. At some level, CAQI and C³A impact every element of the software development life cycle, as shown in the following figure:



The CAQI process begins with having clearly defined standards that are used for code development. Advanced software tools are now available that include industry-accepted best practices, powerful code quality metrics and assessment capabilities that automatically analyze the quality of code under development and offer corrective action if necessary and desired by the developer. Such tools help an organization automate the process of building quality into software development starting with the earliest phases of code implementation. A major goal is to minimize the impact on the developer and to use automation to free him to focus on meeting the functional requirements of the software.

The process of improving code quality is enabled in the code development and testing phases using C³A features that include coding rules, audit tracking, software metrics, automated test case generation and [code coverage](#) tools. Cost-effective software tools have been developed that help organizations automate and implement CAQI and C³A to make it a part of their software development process. Companies can use these automated software tools to build better quality software, reduce software development costs and keep customers happy.

Developing quality software using C³A tools

Improving quality using C³A audit rules and metrics

The goal of C³A is to help developers find and fix problems in their code earlier in the development cycle. Products incorporating C³A technology typically ship with hundreds of software coding "audit rules" and metrics that are based on accepted industry best practices. Based on the rules that are part of the C³A software tool, hints or implementations are suggested while software is being written. C³A uses leading-edge technology to, in effect, put an experienced software quality mentor "on the shoulder" of each developer, providing expert guidance during code development. This happens while code is being typed for the first time. C³A is like an automated, personal, real-time [code review](#). It occurs either continuously as code is written or on demand when the developer chooses to activate it.

Software tools incorporating C³A have evolved to be more than just a "canned set" of rules. They also let developers create new coding rules or modify existing rules so that they meet the specific needs of the organization. Rules can be grouped into

specific sets and distributed to developers whether they are in the next cubicle or across an ocean. C³A tools help organizations build quality into the software using standards that meet organization needs to produce consistent results and increase the return on investment in software development.

An additional benefit of using C³A tools is that it teaches developers to write better code. Rule suggestions presented while coding help the developer to learn to fix coding errors and become familiar with best-practice coding standards of the organization. Universally, developers are very smart people; a C³A tool will have to tell them only once or twice how to do something better before they begin to regularly integrate that thought into their ongoing workflow.

C³A tools incorporate the TPS principle of Hoshin (goals with targets) and Jidoka (build quality during production) by setting goals and providing feedback to help reduce defects. Both the providers and the consumers of software benefit because of improved code quality and consistent enforcement of the organization's best practices, code standards and guidelines.

Improve quality using C³A in testing and deployment

A critical area in building quality software is testing. Most companies that create software have entire QA and testing departments who write test cases, perform code coverage analysis, look for errors and verify that every line of code has been tested. This task is a major cost for companies and often the QA and test teams still don't find all the errors.

In C³A, [unit testing](#) by programmers is viewed as a critical part of the process and has been automated as much as possible to help locate software defects before they get to the QA department. Today's sophisticated software tools can even help automate the task of building tests (i.e. [JUnit](#) tests) to locate software code errors during the development phase rather than requiring teams of QA testers to find problems after the code is written.

In addition, C³A -compliant tools can measure code coverage to evaluate the effectiveness of test code at exercising all possible paths through the code. Automating software testing benefits everyone -- developers, IT managers, QA teams and customers. Developers can work on new product functionality rather than doing mundane tasks such as writing unit tests. IT managers and developers gain valuable information that can help in troubleshooting and fixing software defects early in the software lifecycle. The organization benefits from lower QA costs and better software defect detection rates. And the customers benefit from a flawless user experience.

Summary


Software development organizations face the daunting task of creating high quality, reliable, maintainable software while still delivering their systems on time and within budget. Software tools incorporating CAQI and C³A use the tenets of lean manufacturing to help achieve these seemingly opposing objectives. By creating goals with targets (Hoshin), making every software developer responsible for improving quality and driving out defects (Jidoka), implementing a system of continuous improvements (Kaizen) to eliminate waste and working together to find and share in software quality improvement issues (Jishuken), organizations can infuse quality throughout the entire software development life cycle and simultaneously reduce cost.

CAQI and C³A principles and tools can be successfully applied to the software development process to help organizations find software defects early in the development cycle, build quality into the software, reduce costs and free developers for business critical work. In addition, companies who invest in CAQI and C³A software tools will quickly recoup their initial investment and see an increase ROI that is directly attributable to cost savings from software development time and reduced project delays. And the best part is that automation allows all this to be done without adding substantially to the already-heavy workload of the typical developer.

About the author: Mike Taylor is president, CEO and co-founder of Instantiations Inc. He has extensive senior management experience in the enterprise software development arena. During his 25-year career in the software industry, Mr. Taylor has been a founder in several successful high-technology software companies and a key participant in several successful mergers and acquisitions.

Resources:

1. Keynote presentation, CIO Forum and Executive IT Summit, May 2-3, 2007 in Philadelphia
2. Published in *CrossTalk*, April 2007. "What Engineering Has in Common with Manufacturing and Why It Matters," Alistair Cockburn.
3. Published in *WebSphere Magazine*, November 2004. "Fire Your QA Department and Increase the Quality of Your Software," Mike Taylor, president and CEO of Instantiations Inc.
4. Article by Barry Boehm (University of Southern California) and Victor R. Basili (University of Maryland), "Software Defect Reduction Top 10 List, *IEEE Computer*, January 2001.

Sound Off! -  Be the first to post a message to Sound Off!

Share - [Digg This!](#)  [Bookmark with Del.icio.us](#)

 **SOFTWARE QUALITY RELATED LINKS**

Ads by Google

[5S Posters](#)

High quality 5S posters: UV coated Your one stop shop for everthing 5S
www.the5sstore.com

[Free Kaizen Guide](#)

Implement the Kaizen philosophy with a free 16 page guide to kaizen
www.DuraLabel.com

[Total Quality Management](#)

Tools to Support Continuous Quality Improvement in Your Workplace
vksdifference.com/improvequality

[Six Sigma Certification](#)

Get your Six Sigma Certification Online from Home. Get Free Info!
www.Kaplan.edu

[Process Planning Solution](#)

Visiprise manufacturing planning & scheduling shop floor software
www.visiprise.com/ppc-planning.htm

 **RELATED CONTENT**

■ **Software quality management**

- [How continuous integration improves software quality](#)
- [Continuous Integration: Improving Software Quality and Reducing Risk -- Chapter 2, Introducing Continuous Integration](#)
- [When software projects run over](#)
- [Jumpstart CMM/CMMI Software Process Improvements: Using IEEE Software Engineering Standards -- Chapter 1](#)
- [Essentials of Lean Six Sigma -- Chapters 1 and 4, Introduction and Improvement](#)
- [Business requirements drive Compuware's new application delivery management tools](#)

[CMMI: Guidelines for Process Integration and Product Improvement, Second Edition -](#)

- Chapter 1, About CMMI for Development
- Serena enhances ALM offering with PPM application
- Using SLOC to estimate software costs, schedules
- SEI Checklist

- **Software project management methods and approaches**
 - Jumpstart CMM/CMMI Software Process Improvements: Using IEEE Software Engineering Standards -- Chapter 1
 - How project management methods can improve software
 - Essentials of Lean Six Sigma -- Chapters 1 and 4, Introduction and Improvement
 - CMMI: Guidelines for Process Integration and Product Improvement, Second Edition - Chapter 1, About CMMI for Development
 - ALM 2.0: Application lifecycle management changing to meet development organizations' needs
 - Introducing project management methodologies and ideas
 - How to improve software project estimates
 - Problematic system drags down software project
 - Six Sigma not necessarily beneficial to software development
 - Six Forms of Software Cost Estimation -- Chapter 3, Estimating Software Costs, Second Edition

RELATED GLOSSARY TERMS

- Terms from Whatis.com – the [technology online dictionary](#)
- [collaboration diagram](#) (SearchSoftwareQuality.com)
 - [Gantt chart](#) (SearchSoftwareQuality.com)
 - [PERT chart](#) (SearchSoftwareQuality.com)
 - [rapid application development](#) (SearchSoftwareQuality.com)

RELATED RESOURCES

- 2020software.com, trial software downloads for [accounting software](#), [ERP software](#), [CRM software and business software systems](#)
- Search Bitpipe.com for the latest [white papers](#) and [business webcasts](#)
- Whatis.com, the online [computer dictionary](#)

REFERENCE DESK



Software quality management

NEWS, TIPS & MORE

- [How continuous integration improves](#) (INTERVIEW)
 - [Continuous Integration: Improving So](#) (TIP)
 - [When software projects run over](#) (EXF
 - [Using SLOC to estimate software cos](#)
- [VIEW MORE](#)



SEE ALSO

- **Related Topics:**
[Software quality management](#) , [Team leadership](#)
- **Site Highlights:**
[AJAX Learning Guide](#)
[TheServerSide.NET](#)

GET E-MAIL UPDATES

Submit your e-mail below to receive Sof related news, tech tips and more, delive

Software Testing and QA

Software Requirements

Application Security

E-mail:

Not a member? We'll activate y membership with your subscrip

[About Us](#) | [Contact Us](#) | [For Advertisers](#) | [For Business Partners](#) | [Site Index](#) | [RSS](#)

 **SEARCH**

TechTarget provides enterprise IT professionals with the information they need to perform their jobs - from developing strategy, to making IT purchase decisions and managing their organizations' IT projects - with its network of [technology-specific Web sites, events and more](#)

[TechTarget Corporate Web Site](#) | [Media Kits](#) | [Reprints](#) | [Site Map](#)

All Rights Reserved, [Copyright 2006 - 2007](#), TechTarget | [Read our Privacy Policy](#)