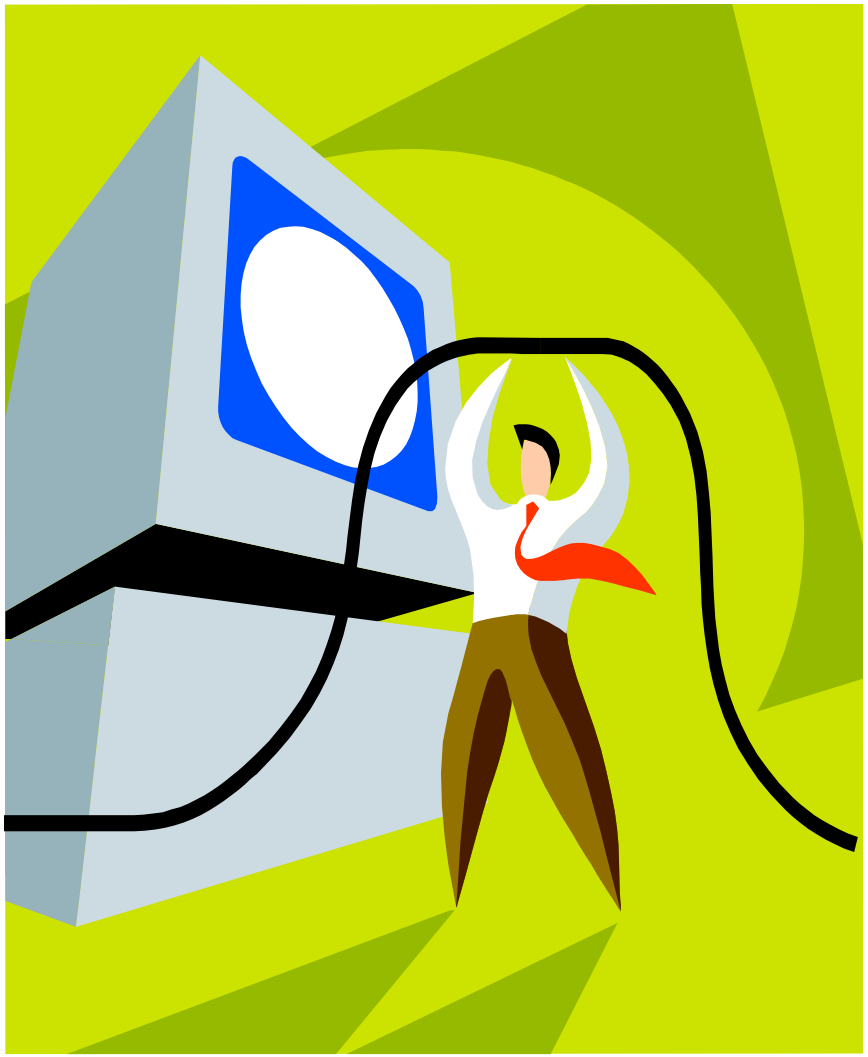


Software Quality Assurance (SQA)

What Is It?
and
Why Should I Care?

Kathleen A. Laing
May 8, 2013
ASQLA Section 0700

Agenda

- Introduction
- What is SQA?
- Software Assurance vs. SQA
- Who are SQA Engineers?
- Why do we (SQA) care?
  (A word about bugs and the costs)
- Tools to Monitor Software Development
- A word about Process - CMMI
- What can We (Quality) do to Improve Software Quality?

What is Software Quality Assurance?

Is there a Difference between Software Assurance and Software Quality Assurance?

Who are SQA Engineers?

Misconceptions about SQA Engineers

- Purely a Testing Organization – not involved in the software development process. Acting only as an independent verification and validation (IV&V) organization

- Not usually involved in the overall software development process.

What is Software Quality Assurance?

The planned and systematic set of all actions and activities needed to provide adequate confidence that the:

- Software work products conform to their standards of workmanship and that quality is being built into the products.

- Organization's quality management system (or each individual process) is adequate to meet the organization's quality goals and objectives, is appropriately planned, is being followed, and is effective and efficient. [1]

1 – Software Quality Engineering handbook, Linda Westfall, ASQ Press

Is There a Difference Between Software Assurance vs. Software Quality Assurance?

The term Software Assurance is included in many Department of Defense (DoD) and NASA contracts as well as Mission Assurance Requirements (MARs).

Software Assurance includes;

- Software Quality Assurance, as well as
- Software Safety
- Software Reliability , and
- Software Independent Verification & Validation (IV&V)

Who Are SQA Engineers?

Trained, experienced quality professionals – not necessarily software engineers. They need a basic understanding of the processes and tools used to monitor development.

A SQA Engineer needs what are referred to as the "soft skills" to be effective in **influencing others toward quality**. (See ASQ CSQE Primer)

Examples of "soft skills" include;

- leadership,
- team building,
- facilitation,
- communication,
- motivation,
- conflict resolution,
- negotiation, and much more.

Why should we care about the quality of Software?

An overview of “bugs” and what they cost us.

The First Computer Bug!

Ever wondered about the origins of the term "bugs" as applied to computer technology? U.S. Navy Rear Admiral Grace Murray Hopper had a firsthand explanation.

The 74-year-old captain, who was still on active duty in 1981, was a pioneer in computer technology during World War II. In 1981 at the C. W. Post Center of Long Island University, Hopper told a group of Long Island public school administrators that the first computer "bug" was a real bug -- a moth. At Harvard one August night in 1945 Hopper and her associates were working on the "granddaddy" of modern computers, the Mark II. "Things were going badly; there was something wrong in one of the circuits of the long glass-enclosed computer," she said. "Finally, someone located the trouble spot and, using ordinary tweezers, removed the problem, a two-inch moth. From then on, when anything went wrong with a computer, we said it had bugs in it."

Hopper said that when the veracity of her story was questioned recently, "I referred them to my 1945 log book, now in the collection of Naval Surface Weapons Center, and they found the remains of that moth taped to the page in question."

Date: 23 Aug 1981 05:38:25-PDT
From: ARPAVAX.sjk at Berkeley
Subject: origin of bug

Computer “Bugs” Still Cause Problems

1980, NORAD reported that the US was under missile attack . The problem was caused by a faulty circuit, a possibility that the reporting software hadn’t taken into account. (caught before launch of return strike)

1983, a Soviet satellite reported incoming US missiles , but the officer in charge decided to follow his gut feeling that it was a “false alarm” and decided to do nothing. (thus avoiding WW3)

1985-87, the Therac-25 medical radiation therapy device was involved in several cases where massive overdoses of radiation were administered to patients caused by a side effect of the buggy software powering the device. A number of patients received up to 100 times the intended dose, and at least three of them died as a direct result of the radiation overdose

1999, the Mars climate orbiter crashed on impact. One of the subcontractors NASA used when building its Mars climate orbiter had used **English units** instead of the intended **metric** system, which caused the orbiter's thrusters to work incorrectly.

The cost of the project was **$327 million**, not to mention the lost time (it took almost a year for the orbiter to reach Mars).

2000, another radiation dosage error happened in Panama City where therapy planning software from US company Multidata delivered different doses depending on the order in which data was entered.

This resulted in massive overdoses for some patients, and at least **five died**. The number of deaths could potentially be much higher, but it is difficult to know how many of the **21 who died in the following years** did so as a result of their cancer or ill effects from the radiation treatment.

And Still More “Bugs”

Mid February 2013, a “bug” in software uploaded to the International Space Station left the station with out communication. The only alternative was the use of 1930’s technology – Amateur Radio “ham” communications when they were over Russia.

The news media downplayed the incident as equilevent to installing software to a home PC.

April 2013, the American Airlines reservation system was down “crashed” due to a software “glitch”. Hundreds of passengers stranded.

If you still have any doubts as to how common software bugs are, just do a search for “software bug” or “software error” or what are now being called “software glitches”, “software escapes” or “software leaks”

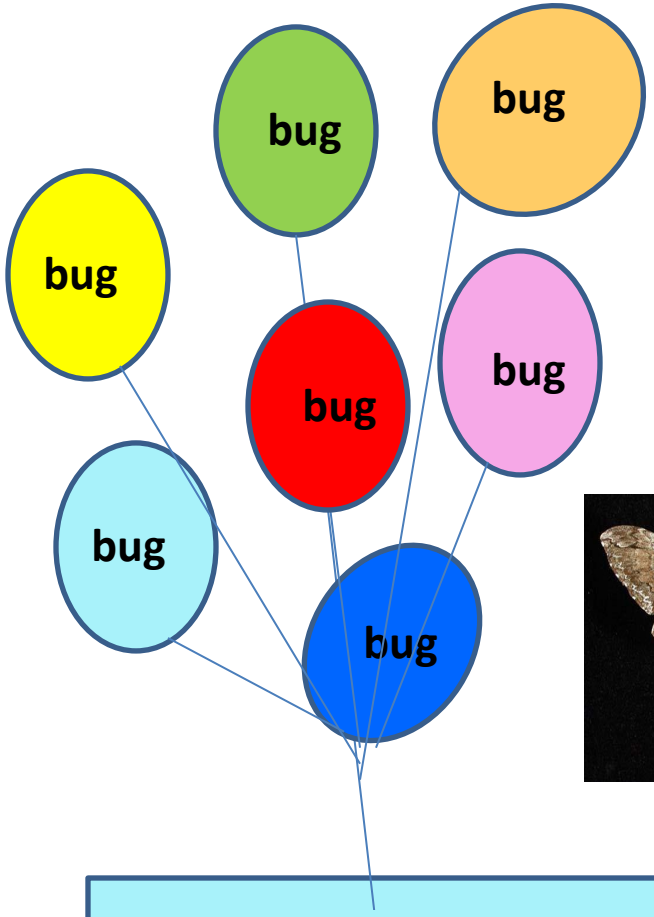
Cost of Software "Bugs"

Some bugs may seem to cause only trivial problems or inconveniences, however, errors in flight control software and software for medical equipment are examples of things that simply cannot be allowed to fail due to the high risk associated with failure.

A 2002 study commissioned by the National Institute of Standards and Technology found that software bugs cost the US economy **$59.5 billion every year** (imagine the global costs…). The study estimated that more than a third of that amount, $22.2 billion, could be eliminated by improved testing. How much more could be saved if we found these "bugs" earlier in the development process?

Defenders Against Poor Software Quality

Software Quality Assurance Engineers



Poor Software Development Practices

Why are we willing to accept software defects as normal in our everyday environment?

"Why Software Quality Assurance Practices Become Evil!" Excerpt from a Technical Paper presented in 2004[2]

"After some 35 years of being involved (and hopefully evolved) in almost every aspect of the software development business from programmer to CEO, I have a theory on why Software Quality Assurance (SQA) practices become evil.

"Let me define my meaning of evil as well, using Webster's dictionary. I do not mean evil in the sense that SQA practices are morally reprehensible or sinful or wicked (definition #1) or that they come from people who have actual or reputed bad character (definition #2)."

"Webster's third definition of evil is right on, that SQA practices cause discomfort and repulsion and are offensive. The topic of SQA practices is usually greeted by developers with the same enthusiasm as a bad odor; of course the source of the odor is often oblivious to the smell."

[2] **Why Software Quality Assurance Practices Become Evil!**
By Gregory M. Pope (found on the ASQ Software Division website)

Why Software Quality Assurance Practices Become Evil!
(continued)

He then goes on the say that he should have named his paper;

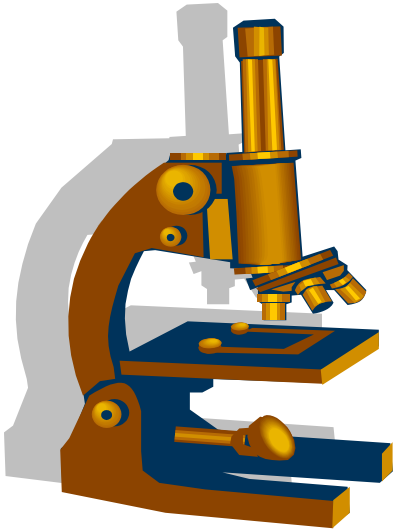
> "Why requirements management, software design, coding standards, inspections, unit test, integration test, system test, and management reviews become a source of discomfort and repulsion and are offensive."

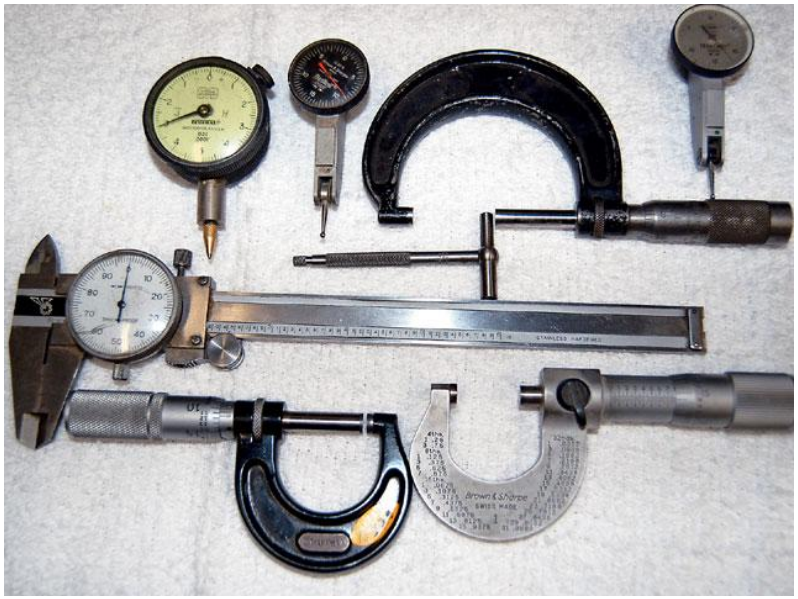
Why would he say (and believe) that SQA processes are evil? Because after reading his technical report , I discovered that the entire premise for his paper was based on the misperception that portability of processes meant from company to company, not from project to project within a company. He also did not understand the concept of tailoring processes to fit the needs of the project. This indicated to me that he fell into a common trap – speaking before you are sure of what you are saying.

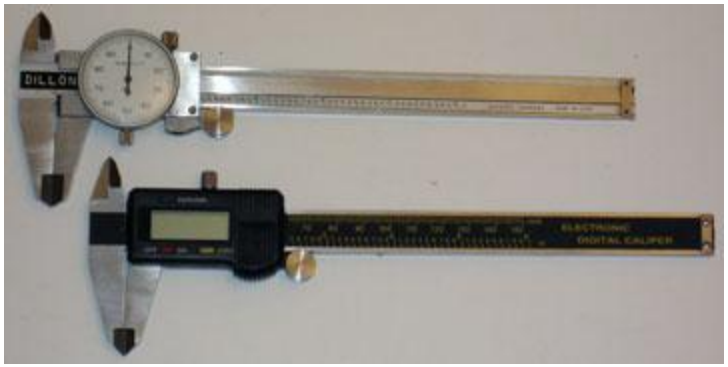
For years Quality (primarily SQA) has been told (by software engineers) that "Software Development is a Creative Process and should not be controlled by the bureaucratic processes imposed by SQA."

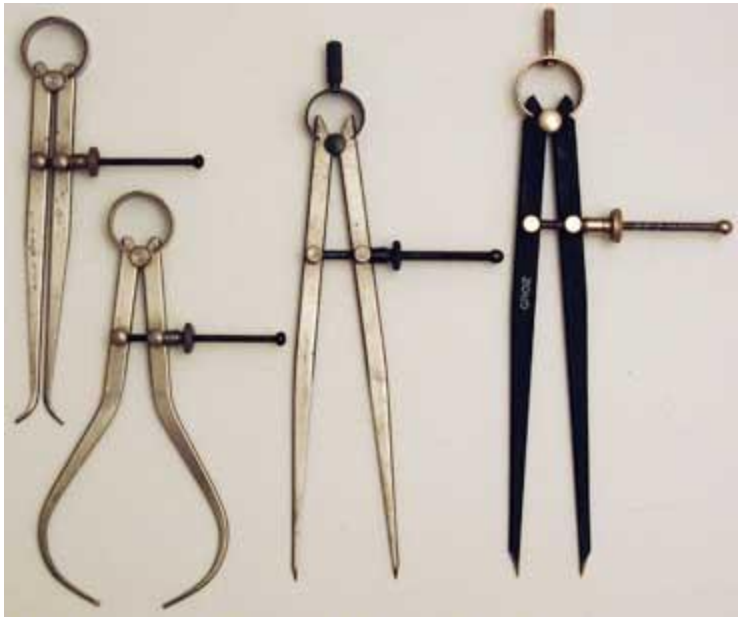
- Just what are the Tools of Software Quality?
- How can you measure something you can’s see or touch?
- How can you be sure the processes used in development are compliant?

Tools for Hardware Quality

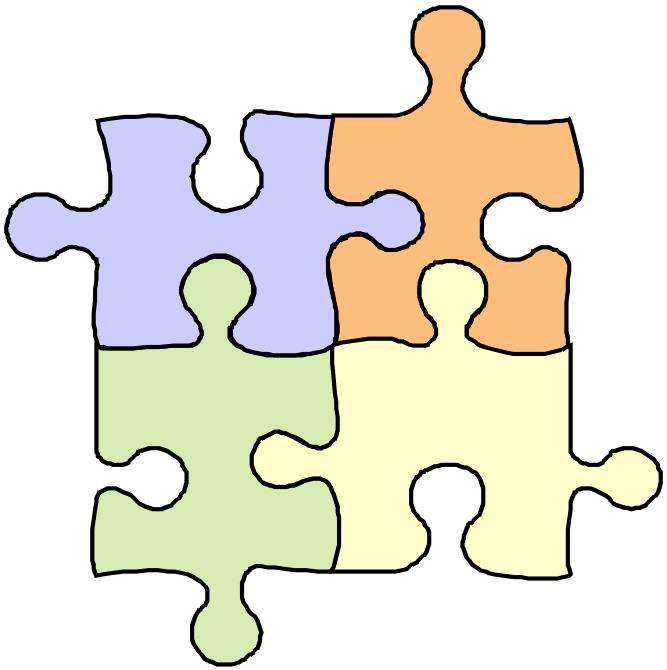








What are the tools for software quality? Is there a path (a map) to follow?



How do we fit the process pieces together to ensure software quality?

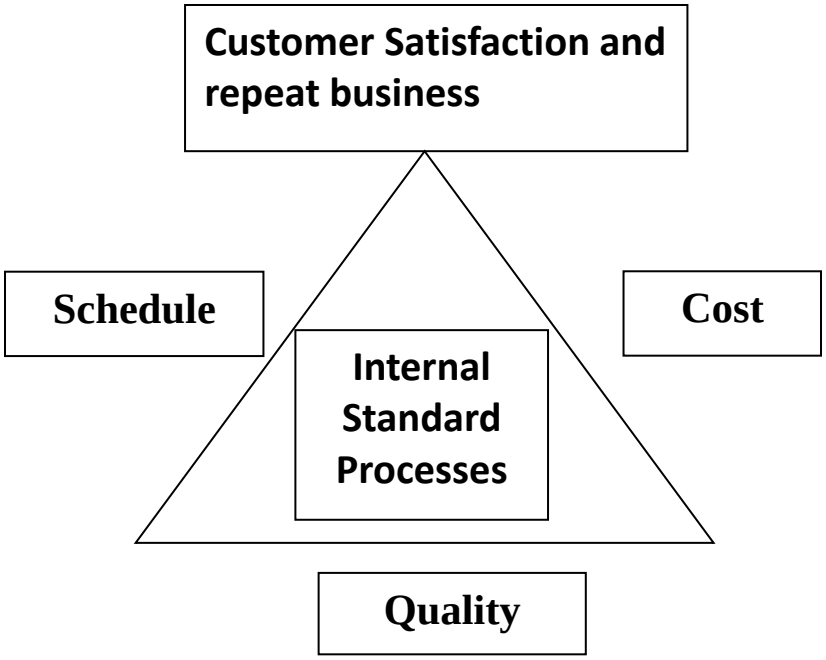
The problems that most organizations address today involve enterprise-wide solutions that require an integrated approach. Effective management of organizational assets is critical to business success.

Organizations that are product and service developers need a way to manage an integrated approach to their development and manufacturing activities as part of achieving their business objectives.

We also hear terms like Agile, SCRUM and function-based development as better ways to perform software development

Affordability

Now, more than ever, companies want to deliver products and services better, faster, and cheaper. At the same time, in the high-technology environment of the twenty-first century, nearly all organizations have found themselves building increasingly complex products and services.



Pyramid for Success

Are these new and improved approaches or just new ways to look at existing processes?

How do these new technologies and process approaches fit into ISO and CMMI?

Where do metrics fit into the process and should they be predictive (defect density) or reactive (number of defects)?

Can SQA help to answer these questions?

When Should SQA get Involved in the Process or Project?

The involvement of Software Quality Assurance should begin in the **early phases of a project** to establish plans, processes, standards, and procedures **that will add value** to the project and **satisfy the requirements** of the project and organizational policies.

Those who perform software quality assurance activities should participate in establishing plans, processes, standards, and procedures to help ensure that **they fit the project's needs** and that they will be **usable for performing quality assurance evaluations**. [1]

1 – Software Quality Engineering, Linda Westfall, ASQ Press

A Discussion of Processes to Support Quality Development

Just a warning in case you need a nap!

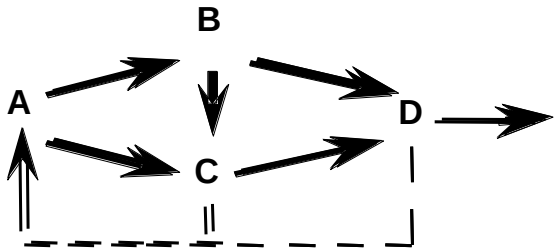
Roadmap to a Standard Software Process

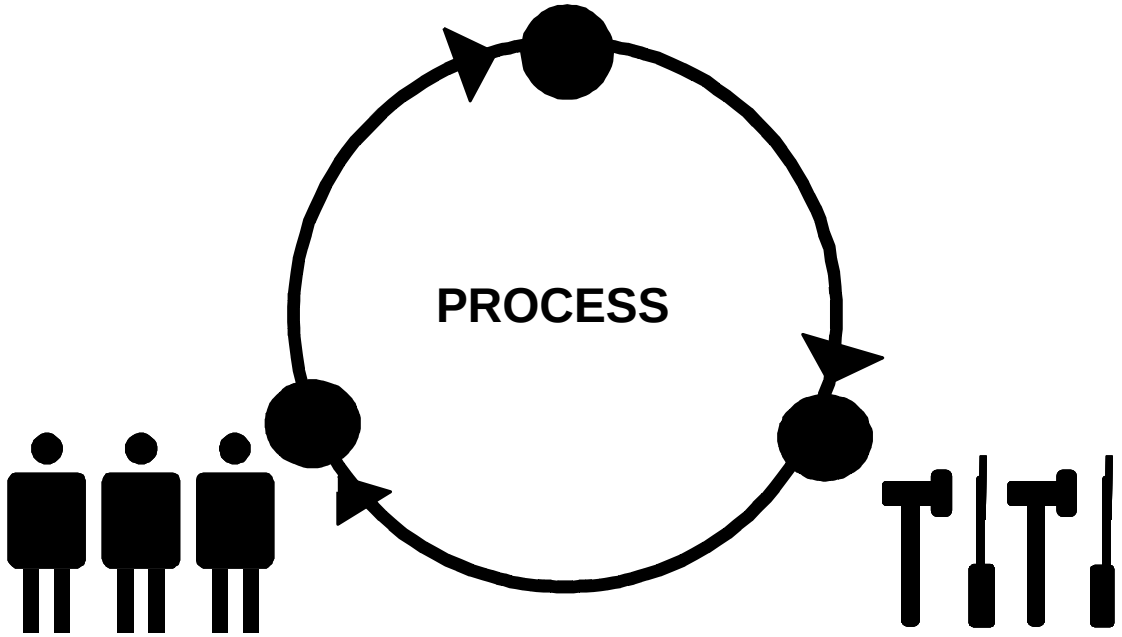
- Between 1984 & 1991 — Carnegie Mellon Software Engineering Institute (SEI) was established by the U.S. Department of Defense and worked with DOD and industry representatives to establish a model for software development.
- 1988 — At the urging of the Defense Advanced Research Projects Agency (DARPA), the SEI created the first computer emergency response team after an Internet worm crippled 10% of computers on the Internet.
- 1991 — The SEI published version 1.0 of the CMM for Software (SW-CMM). More than 30,000 people were trained in the principles and techniques of CMM, and more than 2,400 organizations are assessed on the five-level CMM scale. The SW-CMM is upgraded to CMM Integration (CMMI) in 2000.
- 2010 — SEI released v1.3 of the model was released as the CMMI for Development.

Roadmap to Success

CMMI for Development (v1.3) consists of best practices that address development and maintenance activities applied to products and services. It addresses practices that cover the product's lifecycle from conception through delivery and maintenance. The emphasis is on the work necessary to build and maintain the total product.

Procedures and methods defining the relationship of tasks





People with skills, training, and motivation

Tools and equipment

Three Critical Dimensions of Process Development

There are 22 CMMI process areas, presented here by process area – ▭ shows high maturity (Level 4 & 5) processes

Organizational Focus Processes	Support Processes
• Organizational Process Performance - 4	• Configuration Management (Hardware/Software/Documents)
• Organizational Process Definition	• Decision Analysis and Resolution
• Organizational Process Focus	• Causal Analysis and Resolution - 5
• Organizational Performance Management - 5	• Process and Product Quality Assurance
• Organizational Training	• Measurement and Analysis
Project Management Processes	**Engineering Processes**
• Project Planning	• Requirements Development
• Project Monitoring and Control	• Requirements Management
• Integrated Project Management	• Technical Solution
• Risk Management	• Product Integration
• Supplier Agreement Management	• Verification
• Quantitative Project Management - 4	• Validation

CMMI Process Approaches

The continuous representation enables an organization to select a process area (or group of process areas) and improve processes related to it. This representation uses capability levels to characterize improvement relative to an individual process area.

The staged representation uses predefined sets of process areas to define an improvement path for an organization. This improvement path is characterized by maturity levels. Each maturity level provides a set of process areas that characterize different organizational behaviors.

Process Capability vs. Process Maturity

Process Levels	CMMI Continuous Representation	CMMI Staged Representation
Level 0	**Incomplete**	
Level 1	**Performed**	**Initial**
Level 2	**Managed**	**Managed**
Level 3	**Defined**	**Defined**
Level 4		**Quantitatively Managed**
Level 5		**Optimizing**

Capability Level Descriptions

Level 0 - An *incomplete process* is a process that either is not performed or is partially performed.

Level 1 - A performed process is a process that accomplishes the needed work to produce work products. The specific goals of the process area are satisfied. Although it may result in important improvements, the improvements can be lost over time if they are not institutionalized!

Level 2 - A managed process is a performed process that is planned and executed in accordance with policy, employs skilled people having adequate resources to produce controlled outputs, involves relevant stakeholders; is monitored, controlled, and reviewed. Performance is also evaluated for adherence to its process description. The process discipline reflected helps to ensure that existing practices are retained during times of stress, for example, accelerated schedules.

Level 3 - A defined process is a managed process that is tailored from the organization's set of standard processes according to the organization's tailoring guidelines. It has a maintained process description and contributes process related experiences to the organizational process assets. Processes are managed more proactively using an understanding of the interrelationships of the process activities and detailed measures (metrics) of the process and its work products.

Maturity Level Descriptions

Maturity level 1 - Processes are usually ad hoc and chaotic. The organization usually does not provide a stable environment to support processes. Success in these organizations depends on the competence and heroics. In spite of this chaos, these organizations often produce products and services that work, but they frequently exceed the budget and schedule documented in their plans.

Maturity level 2 - The status of the work products are visible to management at defined points (e.g., at major milestones, at the completion of major tasks). Commitments are established among relevant stakeholders and are revised as needed. Work products are appropriately controlled. The work products and services satisfy their specified process descriptions, standards, and procedures.

Maturity level 3 - Processes are well characterized and understood, and are described in standards, procedures, tools, and methods. The organization's set of standard processes, which is the basis for maturity level 3, is established and improved over time. These standard processes are used to establish consistency across the organization. Projects establish their defined processes by tailoring the organization's set of standard processes according to tailoring guidelines.

High Maturity (Level 4 & 5) Process Descriptions

A critical distinction between maturity levels 3 and 4 is the predictability of process performance. At maturity level 4, the performance of projects and selected sub-processes is controlled using statistical and other quantitative techniques, plus predictions are based, in part, on a statistical analysis of fine-grained process data.

A critical distinction between maturity levels 4 and 5 is the focus on managing and improving organizational performance. At maturity level 5, the organization is concerned with overall organizational performance using data collected from multiple projects. Analysis of the data identifies shortfalls or gaps in performance. These gaps are used to drive organizational process improvement that generates measureable improvement in performance.

Comparison of the CMMI Representations

Continuous Representation	***Staged Representation***
Grants explicit freedom to select the order of improvement that best meets the organization's business objectives and mitigates the organization's areas of risk	Enables organizations to have a predefined and proven improvement path
Enables increased visibility of the capability achieved in each individual process area	Focuses on a set of processes that provide an organization with a specific capability that is characterized by each maturity level
Allows improvements of different processes to be performed at different rates	Summarizes process improvement results in a simple form—a single maturity level number
Reflects a newer approach that does not yet have the data to demonstrate its ties to return on investment	Builds on a relatively long history of use that includes case studies and data that demonstrate return on investment

Example of a CMMI Specific Goals and Practices Checklist - Capability

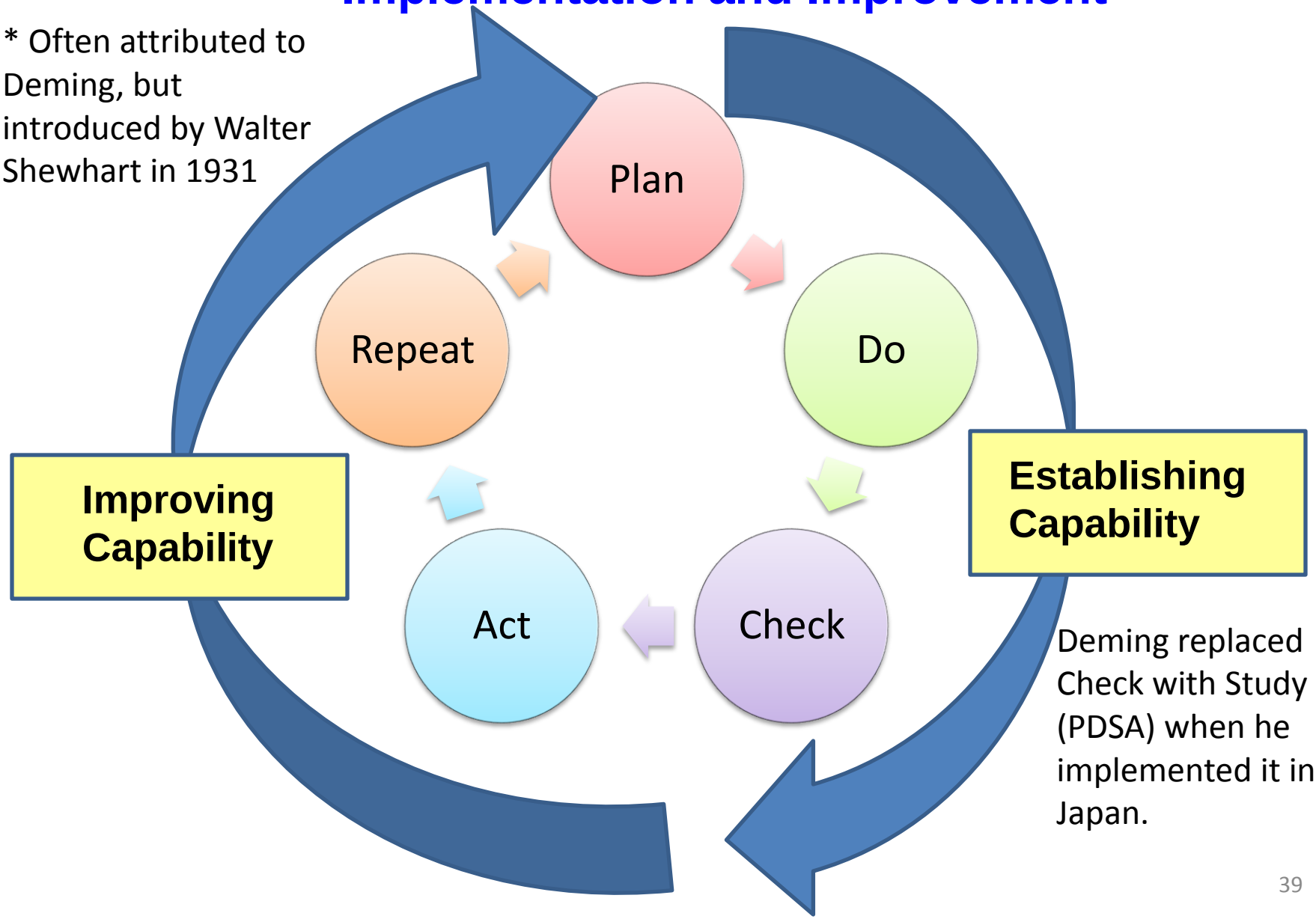
Requirements Development Process Goals and Practices				
The purpose of Requirements Development (RD) is to elicit, analyze, and establish customer, product, and product component requirements.				
SG 1 Stakeholder needs, expectations, constraints, and interfaces are collected and translated into customer requirements.	**Yes**	**No**	**N/A**	**Comments**
SP1.1-1 Are stakeholder needs, expectations, constraints, and interfaces identified and collected for all phases of the project's life cycle?				
Are customer inputs which include needs, expectations, constraints and external interfaces collected to determine what is needed or desired?				
SP1.1-2 Are stakeholder needs, expectations, constraints, and interfaces elicited for all phases of the project's life cycle?				
Are relevant stakeholders engaged using methods for eliciting needs, expectations, constraints, and external interfaces (e.g., dialogue, scenario reviews, models, simulations, prototypes, or new technology demonstrations)?				
Are conflicts in stakeholder needs, expectations, constraints, and interfaces removed and organized into related subjects based on analysis?				
SP1.2 Are stakeholder needs, expectations, constraints, and interfaces transformed into customer requirements?				
Are the stakeholder needs, expectations, constraints, and interfaces translated into documented customer requirements?				
Are methods, criteria, and constraints defined for the verification and validation processes?				

Example of a CMMI Generic Goals and Practices Checklist – Maturity Levels

Generic Practices for Software Configuration Management **Think of generic practices as reminders. They serve the purpose of reminding you to do things right and are expected model components**				
GG1 The specific goals of the process area are supported by the process by transforming identifiable input work products into identifiable output work products.	**Yes**	**No**	**N/A**	**Comments**
GP1.1 Perform the specific practices of the process area to develop work products and provide services to achieve the specific goals of the process area.				
Is the process performed in accordance with the Program & Company Command Media?				
GG2 The process is institutionalized as a managed process.	**Yes**	**No**	**N/A**	**Comments**
GP2.1 Establish and maintain an organizational policy for planning and performing the process.				
Has a policy been established to define how to plan & perform the process?				
SCM Elaboration This policy establishes organizational expectations for establishing and maintaining baselines, tracking and controlling changes to work products (under configuration management), and establishing and maintaining integrity of the baselines.				
GP 2.2 Establish and maintain the plan for performing the process.				
SCM Elaboration This plan for performing the configuration management process can be included in (or referenced by) the project plan, which is described in the Project Planning process area.				
Has a plan been defined and documented for performing the process?				
Has the process description been defined and documented?				
Has the plan been reviewed with relevant stakeholders and their agreement obtained?				
Is the plan revised as necessary?				
GP 2.3 Provide adequate resources for performing the process, developing the work products, and providing the services of the process. The purpose of this generic practice is to ensure that the resources necessary to perform the process as defined by the plan are available when they are needed. Resources include adequate funding, appropriate physical facilities, skilled people, and appropriate tools.				

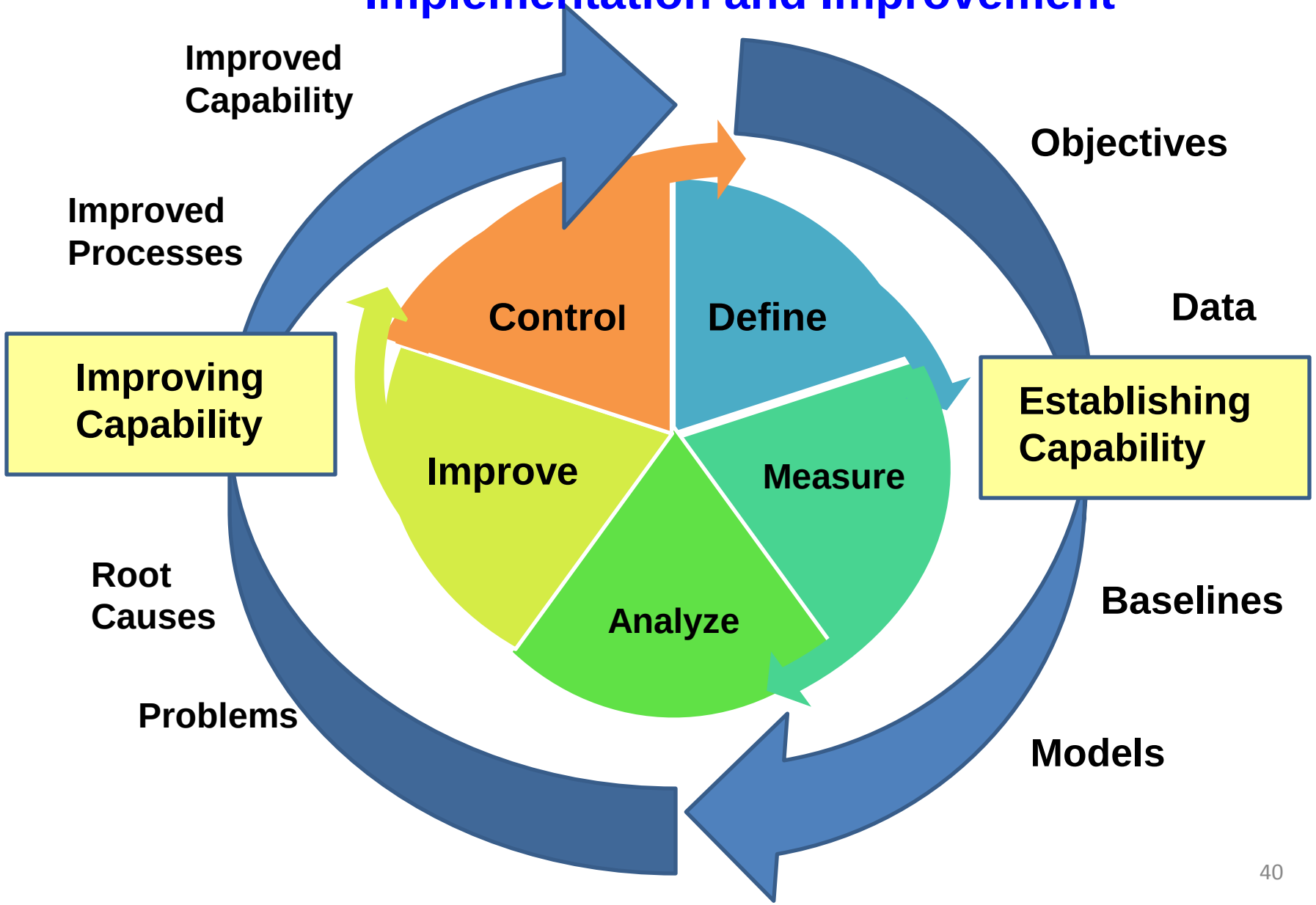
Deming * (PDCA) Supports CMMI Process Implementation and Improvement

* Often attributed to Deming, but introduced by Walter Shewhart in 1931

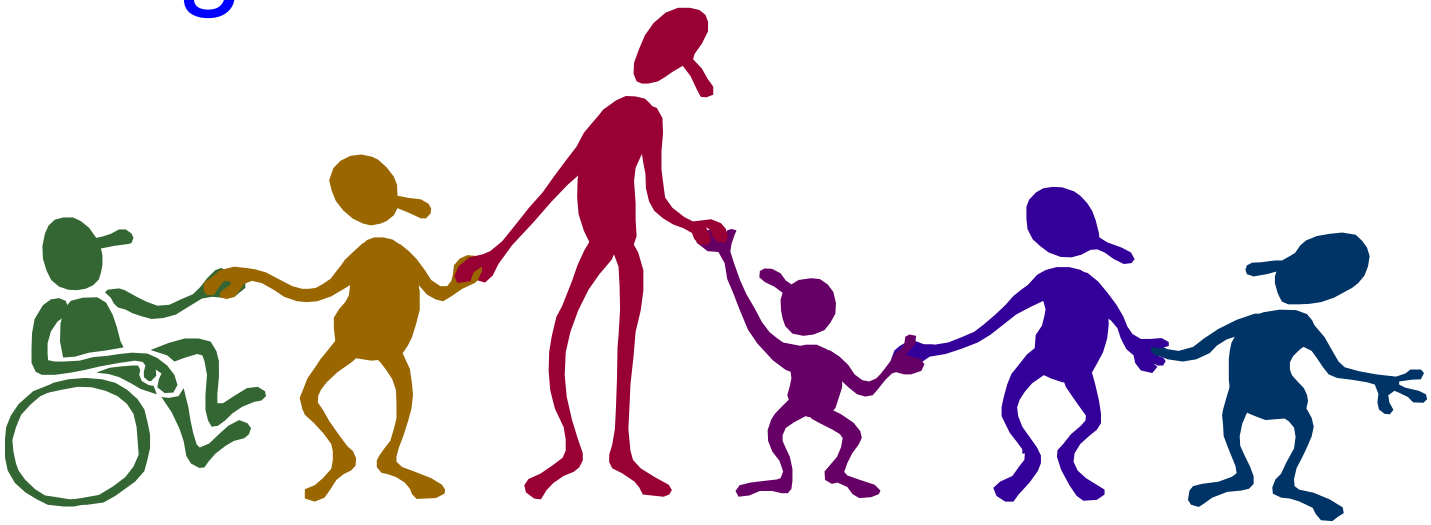


Deming replaced Check with Study (PDSA) when he implemented it in Japan.

6 Sigma (DMAIC) Supports CMMI Process Implementation and Improvement



So, what can **WE** (Quality) do to improve the process and help prevent Software “bugs” ?



Be Aware of any Software Used in Developing or Testing Products

There is a lot of software that is used in development of products which should be controlled. The term "non-deliverable" is often applied to this type of software.

Non-deliverable software is defined as any software that a company develops or procures to support the manufacture or testing of a product (hardware) that is delivered.

Examples of non-deliverable software are;

- Software used to test printed circuit cards (boards)
- Software used to control Computer Numerically Controlled (CNC) machines for fabrication of hardware
- Software used to control temperature settings during curing processes such as composites
- Software used in calibration equipment , such as optical scanners
- Software used in test sets to accept hardware
- Any other software used in the manufacture or acceptance of hardware
- Software embedded in a product that is delivered, but the software (itself) is not considered deliverable

What can you do?

- When performing a First Article Inspection (FAI) on hardware ask if the version software used in the production of the item has been identified in the FAI documentation and is also under Configuration Management (CM) control. It should be part of the FAI evidence.

- When auditing the calibration system, look at any software used in calibration equipment and ensure it is under CM control.

- Ensure any software used in the acceptance testing of a hardware deliverable has been verified and is under CM control.

- Look at the software used to control ovens or other temperature control equipment

- Be aware of what software is used to develop and/or test your products.

Reference Material

The SEI has transferred CMMI-related products and activities to the CMMI Institute, a 100%-controlled subsidiary of Carnegie Innovations, Carnegie Mellon University's technology commercialization enterprise. http://cmmiinstitute.com/

The SEI will continue to pioneer and advance new research in the field of software process management. More information about their current work is available at http://www.sei.cmu.edu/process.

Software Quality Engineering Handbook, Linda Westfall, ASQ Press

"Why Software Quality Assurance Practices Become Evil!"
By Gregory M. Pope (found on the ASQ Software Division website)

Thank you for your kind attention.

Any Questions or Comments?