Improving Quality in Semiconductor Manufacturing Software

SEMATECH
Technology Transfer 97103380A-ENG

Improving Quality in Semiconductor Manufacturing Software

Technology Transfer # **97103380A-ENG**
SEMATECH
October 31, 1997

Abstract: This document is a “call to action” for SEMATECH member companies to examine and improve the performance of software in semiconductor equipment and factories. It probes the causes and costs of software failures and provides responses and guidelines for addressing them, with the aim of producing high-quality software. The document further outlines a joint strategy for member companies to minimize factory downtime due to software problems. It also includes appendixes, directed at manufacturers and suppliers, of software improvement activities.

Keywords: Downtime, Software Process Improvement, Software Reliability, Suppliers

Authors: Ted Ziehe

Approvals: Ted Ziehe, Author
Harvey Wohlwend, Project Manager
Vern Reynolds, Director of Manufacturing Methods
Dan McGowan, Technical Information Transfer Team Leader

Table of Contents

List of Figures

List of Tables

Acknowledgements

The guidelines discussed throughout this document and presented as lists in Section 3.2 and Appendix B are a result of work performed during 1996-97 by the Project Technical Advisory Board (PTAB) of SEMATECH's Software Process Improvement (SPI) Project. The PTAB members and SPI Project staff brainstormed these ideas into existence and refined them through several rounds of review and feedback. It is hoped that their presentation here clearly shows the range and depth of experience in that group.

This publication of the guidelines and recommendations makes possible putting the project's ideas to the test of actual use in factory settings. It will take feedback from real world use to refine and increase the usefulness of this material. Users of the material are asked to share ideas and experiences and thereby make progress toward the shared goals identified in Section 5.1.

In addition to the work of the PTAB, this material also reflects the insights and understandings of the entire SEMATECH SPI Project staff, gained from work with more than 40 supplier companies and all ten member companies.

1 EXECUTIVE SUMMARY

Software failures account for more than **$1 billion** worth of unplanned downtime—that is a conservative estimate of an annual loss absorbed by SEMATECH member company factories. Several trends and conditions in the factories point toward larger losses of this type in the future. For example:

- **The Amount of Software is Increasing.** Software embedded in the tools populating the factories has been increasing by more than 25% per year. At that rate, a tool's software component doubles in three years and increases tenfold in a decade. But it is worse than that: the rate of growth is accelerating to 100% per year for some tools—responding to demands for more functionality in advanced process control, equipment control systems, user interfaces, fault detection, and self-diagnosis for the tools.
- **The Complexity of the Factory-System is Increasing.** A typical 200 mm wafer fab configured for 5000 wafer starts per month uses 25 tool types, with 250 tools installed, of which 30 are cluster tools. All the tools have embedded software. A representative pre-200 mm wafer fab had fewer tool types and more individual tools installed, but no cluster tools (less complexity). In today's 200 mm fab, 70% of the tools are connected directly to the manufacturing execution system (MES) (up significantly), and less than one-third of the tools interact directly with automated material handling (AMH) equipment. In the 300 mm generation, both percentages will be 95% or higher (much more complexity).
- **The Uses of Automation are Proliferating.** Automated material handling together with minienvironments for fabricating 300 mm wafers represent another step by the industry toward reliance on software. The need to automate the recording and collection of equipment status data throughout a fab is well recognized, and the work of implementing it has begun. Automating in situ testing and various applications of sensors further automate the fabrication process.
- **The No-Focus-on-Software Continues.** The cost of equipment for wafer fabrication has risen to 70% of a facility's overall cost, and about 40% of equipment development costs now relate to software (the percentage varies from 30% to 60%, depending on the type of equipment). Yet software gets scant attention by SEMATECH member companies until products are delivered into integration/qualification testing. Resolving software problems at this stage is costly and delays certification for use in production.

A manufacturer that is not proactively working on continuous software quality improvement risks serious problems with factory tools, with suppliers who produce software, and with overall factory operations. Unilateral action by one manufacturer has limited impact. Manufacturers must jointly influence suppliers, and manufacturers working together with their suppliers must improve their collective software engineering capabilities—or the industry is at risk of drowning in the chaos of ever-increasing amounts of unreliable software.

Understating the importance of software quality
can seriously damage your business.

In spite of this ominous prospect, few people in semiconductor manufacturing organizations (outside of software groups) pay attention to software. That effectively neutralizes any effort to increase software's reliability or other attributes of software quality.

This document, directed to SEMATECH member companies, is a call to action. It recommends actions that require rethinking two important issues:

1. Software's role in this industry
2. The responsibility manufacturers have for software quality

This material is designed to stimulate such rethinking. It confronts the reader with four simple facts about software's relationship to semiconductor manufacturing. With each fact is a recommended response and ideas for implementing the recommendation. The four facts and recommended responses are specified below.

Fact #1: Software Competence Is a Necessary Business Asset

The Recommended Response:

- Appoint a manufacturing executive who owns and actively sponsors "developing software competence" that improves software quality
- Charter a senior person as the "advocate" for factory software quality

The Advocate's Responsibilities:

- Establish and mature specific software practices the organization needs
- Develop a well-rounded competence in software acquisition
- Assure the quality of acquired software (see Fact #2)
- Work with suppliers on improving software quality (see Fact #3)
- Achieve the goals set for software quality improvement (see Fact #4)
- Work jointly with other member companies on software issues (see Fact #4)

Fact #2: Acquiring Customized Software Includes Assuring Its Quality

The Recommended Response: Assure the quality of acquired software, including practices that

- Develop and document requirements and specifications for factory software
- Define the meaning of "quality in software" and your expected level of quality
- Reject software that fails to meet your specifications and quality standards

Fact #3: Customer/Supplier Teamwork Is Essential for Factory Software Effectiveness

The Recommended Response (applies to both external and internal suppliers):

- With other member companies, deploy/enforce a policy on software quality for suppliers
- Require suppliers to provide software quality indicators
- Assess and evaluate suppliers' software capabilities
- Sponsor reviews/work projects with key suppliers to accelerate quality improvement

Fact #4: Astute Investments in Software Quality Yield Positive Returns (for both customer and supplier)

The Recommended Response:

- With an internal initiative, establish mature software practices that improve software quality
- Work jointly with other member companies in ways that leverage each company's investment

Major Elements of the Manufacturer/Customer's Internal Initiative:

- Goals with metrics for measuring progress to the goals
- Adequate staff and funding
- Strategies, milestones, and plans for achieving the goals
- Reviews that track progress and make course corrections as necessary

---

This document examines these facts and supports the recommended responses with guidelines for action. The guidelines are easily translated into actions by the following four-step process:

1. Identify non-software roles in your organization that impact the quality of acquired software.
2. Select appropriate guidelines and relate them to such roles.
3. Translate the selected guidelines into specific practices appropriate for the roles identified.
4. Ensure that staff regularly and competently use the practices derived from the guidelines.

Going beyond the manufacturer's role in improving software quality, this material also shows what suppliers must do to produce high quality software. Armed with that knowledge, a customer can assess and accelerate a supplier's progress toward consistently delivering high quality software on time.

Finally, this material outlines a joint strategy for the member companies that addresses the goal of "negligible factory downtime due to software." This strategy and its supporting tactical actions and reporting practices have been developed and deployed by the SPI Project at SEMATECH, working with various SEMATECH constituents. In any joint effort, the level of accomplishment is directly proportional to the extent of member company involvement.

Feedback from the member companies will modify and enhance the contents of this document. Revisions of the material should refine the "facts" presented here, as well as the recommended "responses."

2 SOFTWARE QUALITY AND SEMICONDUCTOR MANUFACTURERS

Ignoring the manufacturer's role in improving software quality can deter business growth.

Semiconductor manufacturing required very little software for a long time. But today...

> **SOFTWARE COMPETENCE IS A NECESSARY BUSINESS ASSET (Fact #1).** Every semiconductor manufacturer is a specifier, developer, buyer, integrator, tester, user, and maintainer of software. That means your manufacturing operation is in part a software business, whether or not you attend to it as a business. What is more, your reliance on software is increasing. To support your factory's productivity and profitability, a mature, well-rounded competence in software practices is essential.

The change in software's role has been dramatic. The trends currently driving it are as follows:

- More functionality and much refinement in tool controls implemented as embedded software
- Increased use of equipment clusters and cells with complex control requirements
- Multiple applications of sensor technology for regulating process operations
- Widespread use of automated material handling and minienvironments
- More frequent reconfigurations of equipment in a fab for short product runs
- Overall automation of fab operations

Table 1 quantifies several aspects of the change by comparing representative factory operations today to what was typical for a pre-200 mm fab. Similar contrasts can be drawn for assembly and test operations.

Table 1 Changing Role of Software in Semiconductor Manufacturing

	Pre-200 mm	200 mm	300 mm Est.	Trends
Tool Types	20	25	30	more variety
Tools Installed	280	250	200	fewer units
Cluster Tools	10	30	40	more complexity
% with SECS[1] to MES	10	70	95	more automated communications
% served by AMH	2	30	95	more automated logistics

This document, looking at the implications of this change, calls SEMATECH member companies to take actions that improve software reliability and overall quality. The manager and staff who must act are, in some instances, unaware that their roles impact software and its quality. So the needs addressed here are as follows:

- Increase the general awareness of software issues and what can be done about them.
- Motivate manufacturers to deal proactively with the challenge of improving software quality.
- Involve manufacturing executives and managers in joint initiatives to improve software quality.

The transition to 300 mm wafers and other industry trends point to software as a major success factor in semiconductor factories.

[1] Semiconductor Equipment Communication Standard

THE RECOMMENDED RESPONSE FOR A SEMATECH MEMBER COMPANY: Software quality issues cannot be resolved unilaterally by external suppliers, not even with the help of internal software groups (your internal suppliers). Manufacturing executives and staff have a role to play. Budgets, schedules, goals, staffing levels, and priorities are at stake, all of which involve executive and management responsibilities. So, take the following actions:

- Appoint a manufacturing executive who owns and actively sponsors "the development of software competence" throughout the organization with the goal of "improving factory software quality" and the productivity/profitability of the factory.
- Charter a senior person as the "advocate" for factory software quality.

  The advocate's major responsibilities are as follows:
- Establish and mature specific software practices the organization needs.
- Develop a well-rounded competence in software acquisition and quality assurance.
- Assure the quality of acquired products that include software.
- Work with suppliers on improving software quality.
- Achieve the goals set for software quality improvement.
- Work jointly with other member companies on software issues.

The information in this section supports implementing this recommendation by addressing the following questions:

- What is the industry's software challenge? (Section 2.1)
- What steps have been taken to address the software challenge? (Section 2.2 and Appendix A)
- What is "factory software" and what does "improving its quality" mean? (Section 2.3)

Section 3 presents guidelines for the "advocate" of factory software quality recommended above. These guidelines point the way to developing organizational competence in acquiring high quality software and using it effectively. Sections 4 and 5 add supporting information about leveraging the investment in software quality that a manufacturer must make.

2.1 Software Issues in Semiconductor Manufacturing

What is the industry's software challenge?

Software's role in semiconductor manufacturing has changed. The issues discussed in this section show how aspects of that change have created a challenge for the industry.

Issue # 1: Dollars for Software. Semiconductor manufacturers and their suppliers spend a lot of money implementing software's expanded role. Table 2 suggests the size and variety of these expenditures.

Table 2 Expenditures to Implement Software's Expanding Role

Software in a representative 200 mm fab	Well in excess of 50 million lines of code
Cost to produce software	Average is $20 per line of code
Increasing volume of software	Embedded software in wafer processing equipment – Has increased more than 25% per year in recent years – This rate of increase is expected to hold for many more years
Increasing complexity of software	Embedded software in wafer processing equipment – More cluster tools – Increasing applications of sensors
Growing volume/complexity of software	Problems will also increase—non-linearly
Software engineers on staff	Many suppliers of wafer processing equipment report – More than 40% of the product development staff positions are software engineers – None are less than 30%, some are about 60% – Unfilled positions are primarily software positions

Issue # 2: Unplanned Downtime Due to Software. Defects and other factors involving software are major contributors to unplanned downtime in semiconductor factories. This loss for the SEMATECH member companies is more than $1 billion per year [1], an estimate based on the following assumptions:

- 5000 wafer starts per week
- Around-the-clock operations
- An average yield of 90%
- $1000 per wafer as the value of good production
- Five hours per week of unplanned downtime
- 150 fabs operated by member companies

A particular company can use its own data to calculate a more accurate approximation of its losses in unplanned downtime caused by software outages.

Issue # 3: Size of the Software Problem. The exact size of the software problem in this industry's production operations is yet to be accurately measured. The two main obstacles are the following:

1. Manufacturers have ignored the need for accuracy in such data.
2. Analysts rate the data that are available as very inaccurate and unreliable.

The work of producing reliable data was recently launched (see Section 5). The best available indirect indicators of the industry's software problem have been collected and published [2]. Two data points from that review are:

- 29% of the failures reported in new (beta test) equipment from January 1994 to April 1997 were caused by software (an average from 26 equipment development/improvement projects).
- Failure incidents in production fabs (caused by software in specific tools) ranged from 80% to 5% (as reported by 14 suppliers for 21 different tools covering the first six months of 1997).

Measuring the industry's software problem with accuracy is not easy. But such measurements are important. They motivate the work of improvement and provide a basis for setting improvement priorities and determining root causes.

Issue # 4: Software is Different. For years, semiconductor manufacturing progressed without software. Now that it is a critical element, it must be understood how software is different from hardware—and that software engineering differs from hardware engineering. For example:

- Software is designed and developed, but is not manufactured like hardware—so quality assurance for software must be an integral part of its design and development process.
- Software never wears out through use like hardware; variations in the use of software always have the potential for revealing defects not previously encountered.
- Since changes to software are not visible, many people consider them relatively easy to make compared to hardware changes, so that the disciplines of change control often are ignored.
- Software engineering is the least perfected of the engineering disciplines, so new software tools and methods are constantly presenting alternatives to "how we currently do it."

The industry grew up dealing with hardware and an array of basic physical processes. The industry is now confronted with developing its software capability as work continues on hardware and process challenges. To a point, embedded software can be viewed as just another component of a factory tool. But the characteristics cited above mean that processes designed for hardware sometimes will require change. A manager must come to terms with software's differences and recognize where and how to change the processes. The guidelines in Section 3 and Appendix B will help.

Issue # 5: Staffing Software Positions. Suppliers (both internal and external) are creating positions for software engineers. Some now report more than 50% of their engineering positions are devoted to software. Filling the software positions is another matter. In one company, 65% of the unfilled engineering positions relate to software. Some are unable to find capable people with applicable experience. High turnover rates for current staff often increase the challenge. Since other industries offer many opportunities for software engineers, semiconductor manufacturing must publicize its opportunities to be competitive. Also, individual companies must offer rewarding and challenging workplaces for software engineers, and that includes career paths that lead somewhere.

Issue # 6: Need-to-Know about Software. Beyond the need for experienced software engineers, a general “need to know about software” exists. For example, manufacturers are recognizing that factory planners, process engineers, purchasing agents, and factory operations staff and managers all require some level of software knowledge. Similarly, suppliers see the need for software-literate staff in sales, technical support for sales, product support after the sale, and such engineering support roles as requirements gathering, configuration management, and product testing

As software knowledge gets more important, its absence causes problems. Table 3 illustrates such problems with examples drawn from operations in both manufacturer and supplier organizations. The problems listed often challenge the goals set by management for factory productivity and profitability.

Table 3 Examples of Operational Problems Related to Software

1.	During factory planning, requirements for software are left incomplete or ignored entirely.
2.	During equipment procurement, suppliers’ software and software capability are not evaluated.
3.	Serious attention to software is often delayed until after equipment (with embedded software) is delivered.
4.	Customers force “specials” on suppliers, even when there is little value added.
5.	Suppliers accept change requests right up to the software release date (creeping requirements).
6.	Suppliers let changes, delivery timing, price competition destroy software schedules/processes.
7.	Reports of problems and downtime during factory operations are inaccurate or even nonexistent.
8.	Many customers are not equipped to qualify software changes before releasing them to production.
9.	Many customers are drowning in massive amounts of data produced during factory operations.

Equipping managers and staff who are not software specialists and lack the needed degree of software competence is a challenge to both manufacturers and supplier companies. This document helps to meet that need.

The Software Challenge for Manufacturers. Software’s role in semiconductor manufacturing challenges each company—manufacturers and suppliers alike—to develop an overall competence in software practices. The software challenge for a supplier has three parts:

1. Establish a comprehensive “software process” for developing, evolving, and maintaining software.
2. Maintain the integrity of this software process as it is used in combination with other business processes; for example, those used to develop products, and those used to market, sell, and support them.
3. Continuously improve quality in the software produced, and in the organization’s overall software capability.

Your challenge as a manufacturer also has three parts, but they are quite different. As the "software customer," you do the following:

- Acquire from many suppliers software that typically is a blend of commercial products and custom features.
- Configure the acquired software into a "factory system," thereby automating certain (and eventually perhaps all) factory operations.
- Use the system of software that is created to operate the factory, monitor its operation, and take corrective actions when needed, providing any required technical support.
- Alter the system of software in ways that evolve the factory in response to changes in the business, its markets, and the technologies used.

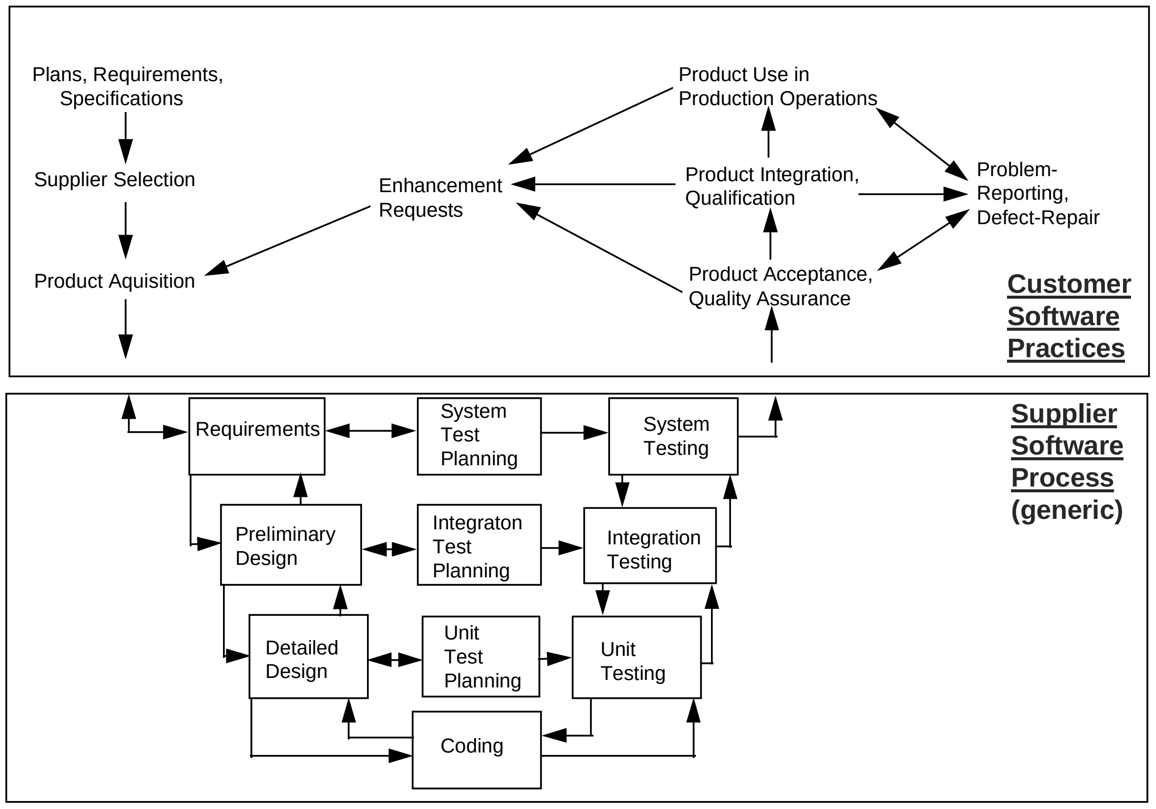
The core of your software-related role is assuring that suppliers can and do deliver products that meet your requirements and specifications, including your software quality specifications. To accomplish that means you also do the following:

- Provide suppliers requirements and specifications for software.
- Define software quality and specify the level of quality expected.
- Reject software that fails to meet your specifications and quality standards.

Establishing throughout your organization the software practices these actions require is part one of the software challenge for a manufacturer. The other two parts are

- Partnering with your suppliers
- Working jointly with other member companies

Partnering with a supplier is a way of saying, "Be sure that your software practices and your supplier's software process work effectively in combination." Figure 1 indicates ways in which the two must be compatible. A second important aspect of such partnerships involves the customer motivating suppliers to improve software quality, and taking actions that accelerate a supplier's progress in doing it.



Plans, Requirements, Specifications
Supplier Selection
Product Aquisition
Enhancement Requests
Product Use in Production Operations
Product Integration, Qualification
Product Acceptance, Quality Assurance
Problem-Reporting, Defect-Repair
Customer Software Practices
Requirements
System Test Planning
System Testing
Preliminary Design
Integraton Test Planning
Integration Testing
Detailed Design
Unit Test Planning
Unit Testing
Coding
Supplier Software Process (generic)

Figure 1 Synergy of Customer/Supplier Software Roles

Working jointly with other member companies simply means there are many ways to leverage each manufacturer's investment in software quality. This leverage comes by distributing tasks that must be done, learning from one another's best practices and experiences, sharing resources, and (perhaps most importantly) saying in unison to suppliers, "Improving software quality is a requirement," and then ensuring that this requirement is satisfied.

Software competence by manufacturers and suppliers alike is prerequisite to continuing the success of semiconductor manufacturing. An effort to establish the software practices or process in a company—thereby developing the mature, well-rounded competence in software that is needed—typically meets resistance, competes for resources with other priorities, is slowed by a lack of expertise, and encounters many other obstacles. The work is somewhat tedious and time-consuming, and progress is often not obvious. But it is essential work, and those who do not do it stand to pay a heavy toll.

2.2 Software Quality Improvement Policy

What steps have been taken to address the software challenge?

SEMATECH established the SPI Project to provide leadership and coordination in the work of improving software quality. The project's primary mission is to improve reliability and the overall quality of software, focusing on suppliers strategically important to SEMATECH members. Other SEMATECH programs deal with additional or supporting aspects of software improvement. Section 5 profiles the status of these efforts.

Through the SPI Project, SEMATECH members have established a policy on Software Quality Improvement (SQI) for suppliers. Figure 2 shows the policy in its relationships to member companies and suppliers. The policy carries the following messages to suppliers from their customers:

- Product reliability is very important to us, so give it priority and be sure to put high priority on software reliability.
- Deliver indicators of quality with each release of software, and use those indicators to drive further improvement of software quality in future deliveries.
- Improve software quality with a continuous, objective effort that matures your organization's software process and increases your organization's software capability.

The SQI Policy identifies a basic set of quality indicators (4-Ups Metrics) that suppliers use to drive continuous improvement; customers should request to see data for these indicators on a regular basis.

The SQI Policy, a call to action for suppliers, is a very succinct guideline for producing high-quality software and dealing effectively with customers. It directs suppliers to take the following actions:

- Make a public organizational commitment to software quality.
- Develop software quality goals and action plans.
- Use a software problem reporting and corrective action system.
- Use a software revision control system.

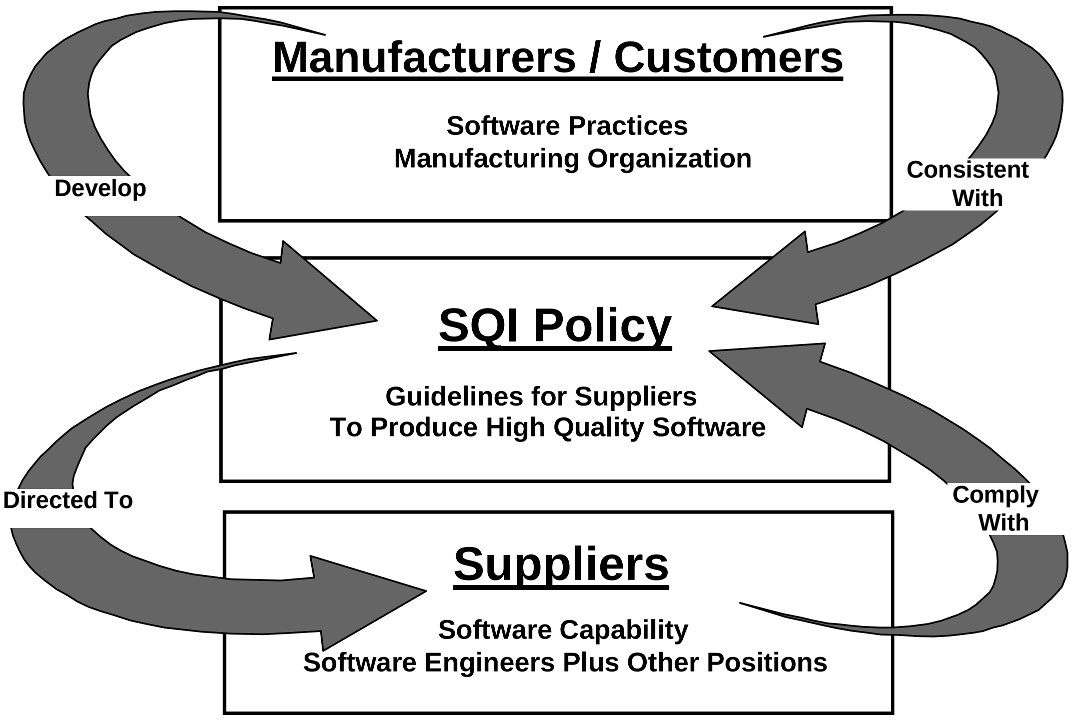


Figure 2 Relationships of Customers and Suppliers to the SQI Policy

- Charter, staff, and fund the continuous improvement of software quality.
- Provide customers the following data with each major software release:
  - Software test plan and test report
  - Release notes that include changes, known defects, work-arounds
  - Status of software standards compliance, e.g., factory communications
- Produce and use the following metrics as quality indicators (4-Ups Metrics):
  - Software defect trend
  - Software reliability (measured by mean time between failures [MTBF] and mean wafers between interrupts [MWBI])
  - On-time delivery
  - Process maturity (measured by SSQA Module 3 or Software Engineering Institute [SEI] levels)

Copies of the SQI Policy were provided to all members of SEMI/SEMATECH, an association of U.S. suppliers. With this step, SEMATECH members acknowledge that software is critical and that collectively they share responsibility for software quality. Appendix A is the complete text of the SQI Policy, including an explanation of the 4-Ups Metrics.

2.3 Terminology

What is 'factory software' ?

What does 'improving its quality' mean ?

Business discussions of semiconductor manufacturing typically are couched in terms of the following:

- Product designs and yields
- Equipment for wafer fabrication, assembly, and test
- Manufacturing facilities
- Factory managers and staff

Software's role—although dramatically increased in recent years—remains largely invisible. But this document is about *factory software*—more specifically, on what it takes to produce high-quality factory software—and even more specifically on practices and processes that produce software with quality higher than is typical in semiconductor factories today.

The following words and phrases have specific meanings throughout this report:

- *Semiconductor factory,* in contrast to a semiconductor fab
- *Factory software*, as distinct from factory equipment and systems
- *Software quality* defined in terms of the *attributes of quality* most important to the manufacturers and the *quality indicators* that reflect the level of quality in the software itself
- *Sources* of factory software, specifically the *suppliers* who produce it and their *customers* who provide its requirements and specifications, and who own the task of assuring its quality

- The *business relationship* between a customer and suppliers, specifically as it relates to software

The following paragraphs briefly explain each word and phrase italicized above.

A *semiconductor factory* encompasses all operations that fabricate semiconductor components on wafers, then assemble and test salable products. Typically, such a factory is not a single location; in some instances it includes operations on two or more continents.

Table 4 provides a context of meaning for *factory software*.

In this document, the *quality* of factory software refers to the following four attributes of quality:

- Reliability: Factory software runs without causing production work stoppages.
- Usability: Factory software is easy to use by operators/managers, requiring minimal training.
- Connectability: During integration/qualification, the software quickly and effectively communicates/interacts with other software elements.
- Maintainability: Factory software is easy to repair and modify as needed.

Table 4 Meaning of *Factory Software*

Factory Operators	The managers and staff who are responsible for operations that make semiconductor components and from them produce ready-to-sell products; they accomplish this by operating factory equipment, largely through interactions with software.
Factory Software	The complex of software elements that control the equipment that makes semiconductor components on wafers, assembles/tests the components as ready-to-sell products, and provides operators with an interface for exercising control over the equipment and systems, e.g., MES, several cell controllers, the equipment controls in each tool or cluster, and other software embedded in factory equipment.
Networked Platforms	The complex of processors, storage devices, and communication/interface devices forming systems that execute factory software, interface with operators, and sense/activate equipment (platforms include elements of software distinct from factory software)
Factory Equipment	The tools and devices that store, handle, move, process, assemble, and test wafers in ways that yield semiconductor products. Some components of the networked platforms are integral parts of fab equipment, and elements of fab software are embedded within these integrated platforms.
Semiconductor Products	The final results of the collaboration by operators, software, platforms, and equipment in wafer fabrication together with product assembly and test operations.

To improve software quality means improving any of these attributes of quality. Reliability is considered most important, with connectability in second place. The other two, both important, do not enter into the discussion that follows.

Two primary indicators of quality—acceptable evidence of the software's quality level—are as follows:

- Software defect trend
- Reliability in production operations (measured by MTBF/MWBI)

In addition to attributes of quality in the software itself, the document also addresses two attributes of a mature organizational capability in software:

- On-time delivery: the ability to set and meet software delivery dates.
- Maturity rating: the objective rating of an organization's software capability based on a recognized model of mature software practices, e.g., SSQA Module 3 or the Maturity Levels of SEI's Capability Maturity Model (CMM).

Both ratings of organizational maturity apply to manufacturers as well as suppliers.

Finally, factory software comes from two principal sources, both of which influence its quality:

- The *customer* is the factory-operator, the manufacturer, who performs the following:
  - Plans and specifies factory operations (that require software)
  - Acquires products (that include software)
  - Applies some customizations to the acquired products (software, among others)
  - Assembles it all to create a factory (that includes software systems)
  - Operates the factory as a "system" that ultimately yields semiconductor products
- The *suppliers* of software that the factory-operator needs include the following:.
  - Groups internal to the customer's organization (internal suppliers)
  - External software suppliers who sell lines of products and supporting services
  - Suppliers of hardware products, i.e., equipment controlled by embedded software

These designations of customer and supplier are simple enough, but the *business relationship* between a customer and any supplier is not so simple. For example, a customer may acquire factory software by purchasing equipment that has embedded software, contracting for an MES or other standalone software product, or requesting/ordering custom software enhancements from an internal supplier. The recommendations and guidelines throughout this document can enhance and support various aspects of this business relationship that relate to software.

3 GUIDELINES FOR IMPROVING SOFTWARE: THE CUSTOMER'S ROLE

Undermanaging areas that impact software quality
can damage your business.

A customer defines quality. Customers also determine what level of quality they want suppliers to deliver, and test delivered products to see if their quality specifications have been satisfied. For semiconductor manufacturers that means the following:

> **ACQUIRING CUSTOMIZED SOFTWARE INCLUDES ASSURING ITS QUALITY. (Fact #2).** Every semiconductor factory uses a massive amount of acquired software, much of it tailored by various suppliers in response to customer need. It is important for all software to run without causing delays or downtime (reliability). Many elements also must interact effectively with software from other suppliers and with factory staff (connectability). It is your responsibility, as the customer, to prioritize these two attributes of quality (and perhaps others), specify the level of quality you expect, and do the qualification needed to assure that delivered software meets your specifications.

In the world of 300 mm wafers, software's potential for impacting factory productivity and profitability will increase significantly. It will take high quality to make that impact positive. So it is timely to focus on factory software, on the suppliers producing it, and most importantly on how you, a manufacturer, acquire software and release it for use in production with *assurance of high quality*.

This section and Appendix B offer guidelines for assuring software quality. Although the guidelines are software-related, they are for staff in non-software roles. They target people in several organizational functions, but have a single aim: to increase quality in the hundreds of elements of software acquired for and used in semiconductor factories today. This software (some of it standalone but much of it embedded in equipment) comes from many suppliers (both internal and external). In a factory, the software is installed into integrating/automating networks of computers, or it is activated by connecting the equipment that embeds it to a cell controller or MES. Once installed and activated, the software is executed in complex interactions among the factory's tools, operations staff, and managers. Assuring the quality of all that software, as it is delivered, is the purpose these guidelines serve.

THE RECOMMENDED RESPONSE FOR A SEMATECH MEMBER COMPANY: Much responsibility for quality in factory software belongs to external suppliers and internal software groups, but not complete responsibility. The actions or inaction of manufacturing executives, managers, and staff in various non-software roles also affect software quality. So take the following actions:

- Proactively assure the quality of all acquired software.
- Regularly use practices that:
  - Develop and document requirements and specifications for factory software
  - Define what you mean by quality in software and the level of quality you expect
  - Use testing and an analysis of test results as a basis for assuring software quality
  - Reject software that fails to meet your specifications and quality standards

The information in this section supports implementing this recommendation by addressing the following questions:

- How do actions by a manufacturer influence software quality? (Section 3.1)
- What guidelines assure the quality of acquired software? (Sections 3.2 and Appendix B)
- How does a manufacturer organize and staff to assure software quality? (Section 3.3)

By following these guidelines, a manufacturer establishes effective working relationships with suppliers. To enhance such relationships, Section 4 profiles the suppliers' role in software quality. Armed with this information, you as the customer can assess and accelerate a supplier's work to improve software quality.

By implementing these guidelines, a manufacturer also realizes that improving software quality involves fundamental changes. Joining forces with other manufacturers and going through the transition together leverage the investments in staff and other resources required. Section 5 identifies the joint efforts which the SEMATECH SPI Project is currently orchestrating.

3.1 Software Quality and the Manufacturer/Customer

How do actions by a manufacturer influence software quality?

In a semiconductor factory, linkages made with software form a factorywide and very complex system. Like any other customer who buys customized complexity from suppliers, you are well advised to proceed along the following lines:

1. Plan, engineer, and state your expectations as requirements/specifications, specifying how you will use software and what you want it to accomplish, and stating its performance specifications.
2. Select your suppliers according to their software capabilities, then procure products from them based on your plans and specifications, including those related to software.

3. Put together and test the products you procure, again according to your plans and specifications, and in a manner that covers all acquired software.
4. Operate your factory according to your plans and specifications, implementing, qualifying, and recording configuration or equipment changes (including software changes) as they are made.
5. Manage your organization in a manner that assures high-quality results in each stage of the business process, including a body of practices that ensures high software quality.

Factory productivity and profitability—important metrics for any semiconductor manufacturing business—are results of the third and fourth functions listed. In both functions, positive performances depend in part on two attributes of software quality: reliability and connectability. The levels of quality you achieve for these two attributes of software quality (and any others important to you) depends on how well you perform the other three functions.

Each member company has an organizational structure and business process for the five functions listed above. Figure 3 is a generic representation of the five functions and the process they form. The important point is that staff and management actions (or inaction) throughout this structure impacts the quality of acquired software. Following are some examples:

- Factory planning, engineering, specifying. When requirements for software are not developed (with other requirements), there is a void when qualifying acquired products for production use.
- Procuring factory equipment and systems. When product acceptance criteria for software are not developed, acceptance of defective software increases qualification and production costs.
- Integrating factory equipment and systems, testing, qualifying. When software quality's priority remains low to this point, integration and testing take longer and cost more.
- Operating the production-factory. When data about production-interrupts and equipment/system downtime are not accurate and complete, improving overall equipment effectiveness (OEE) is very difficult.
- Assuring high-quality software. When no one owns the requirement to have high quality factory software, more problems, higher emergency workloads, and increased cost result.

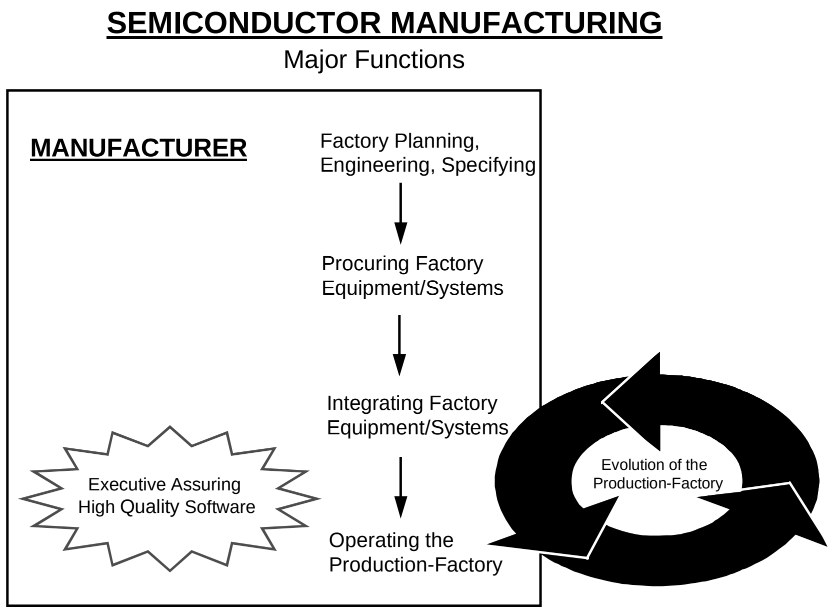

SEMICONDUCTOR MANUFACTURING
Major Functions
MANUFACTURER
Factory Planning, Engineering, Specifying
Procuring Factory Equipment/Systems
Integrating Factory Equipment/Systems
Operating the Production-Factory
Executive Assuring High Quality Software
Evolution of the Production-Factory

Figure 3 Major Functions in Semiconductor Manufacturing

In addition to the functions listed in Figure 3, customer/supplier interactions dealing with software (or that should deal with software but do not) also impact quality. Figure 4 extends the earlier graphic to show such interactions with both internal and external suppliers. In this representation, the supplier interfaces are shown as they currently exist in all too many cases. But the guidelines in this document support customer/supplier communications of the following kinds:

- Product requirements/specifications, including software quality criteria that must be satisfied
- Product descriptions and user information from the supplier to the customer
- Indicators that show what level of quality is present in the delivered software
- The requirement that suppliers continuously improve software quality
- Reasons why software quality is important for the customer, and just how important it is
- What a supplier must do to produce high-quality software, and evidence the supplier is doing so

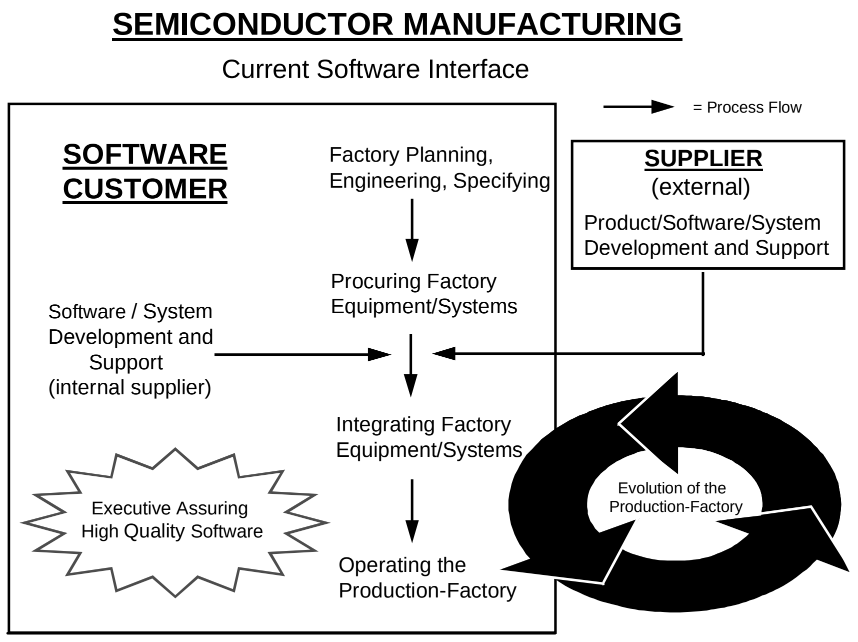

SEMICONDUCTOR MANUFACTURING
Current Software Interface
= Process Flow
SOFTWARE CUSTOMER
Factory Planning, Engineering, Specifying
Procuring Factory Equipment/Systems
Integrating Factory Equipment/Systems
Operating the Production-Factory
Software / System Development and Support (internal supplier)
Executive Assuring High Quality Software
SUPPLIER (external)
Product/Software/System Development and Support
Evolution of the Production-Factory

Figure 4 **Current Interface for Software Between Suppliers and Semiconductor Manufacturers**

Establishing practices throughout your organization based on these guidelines creates interfaces with suppliers of the type shown in Figure 5. Timely, clear, and complete communications with suppliers about software greatly enhance software quality.

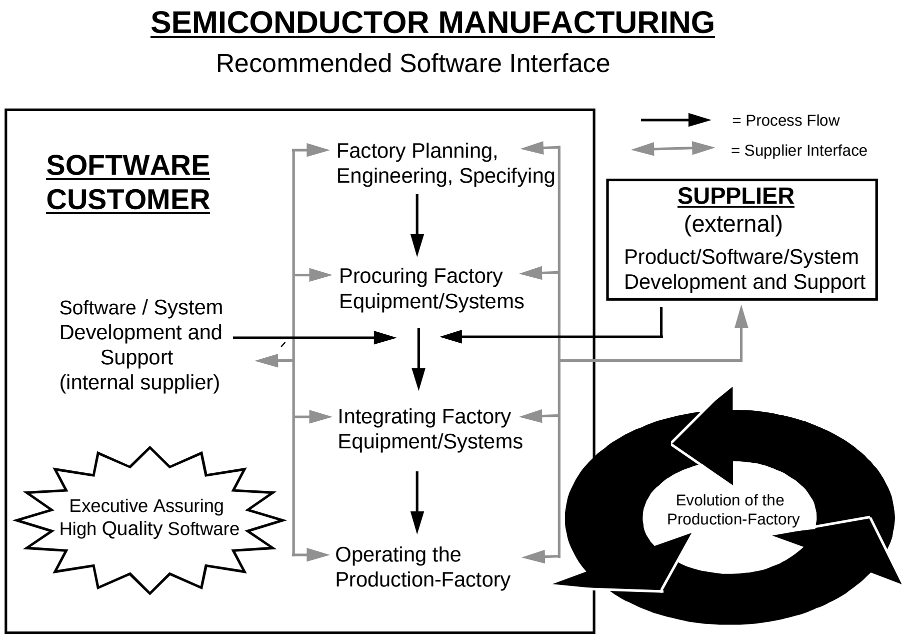

SEMICONDUCTOR MANUFACTURING
Recommended Software Interface
= Process Flow
= Supplier Interface
SOFTWARE CUSTOMER
Factory Planning, Engineering, Specifying
Procuring Factory Equipment/Systems
Integrating Factory Equipment/Systems
Operating the Production-Factory
Software / System Development and Support (internal supplier)
Executive Assuring High Quality Software
SUPPLIER (external)
Product/Software/System Development and Support
Evolution of the Production-Factory

Figure 5 **Recommended Interface for Software Between Suppliers and Semiconductor Manufacturers**

Each member company should apply the guidelines in this document and drive the software practices you develop with goals for quality improvement. Select goals that reflect your business priorities, and express them in terms of one or more of the following benefits:

- Increasing the reliability of software delivered to you
- Decreasing the downtime of your equipment due to software problems
- Increasing the OEE rating of your equipment
- Decreasing the time required to integrate, test, and qualify your equipment for production
- Eliminating surprises from new releases of software delivered to you
- Getting the earliest possible benefits from new releases of software you receive

Measure progress toward the goals at regular intervals, quickly remove barriers as they arise, provide rewards and in other ways celebrate successes, and make course corrections as needed.

3.2 How Customers Improve Software Quality

What guidelines assure the quality of acquired software?

Quality in software from suppliers can be improved through use of the Policy on Software Quality, and by requiring data from your suppliers for quality indicators like the 4-Ups Metrics (see Section 2.2). To leverage this improvement from suppliers, develop a mature, well-rounded competence in software practices throughout your own organization. Applying the guidelines in this section and Appendix B facilitates that. To apply the guidelines, take the following actions:

- Identify non-software roles that impact software quality
- Relate the guidelines in this section and Appendix B to such roles
- Translate the guidelines into specific practices appropriate for the roles identified
- Ensure that staff regularly and competently use the practices derived from the guidelines

These are guidelines for building quality into software before it is delivered, rather than improving it after the fact. That is done by establishing software practices in eight different areas, covering all functions identified in Figure 5 (see Section 3.1). These eight areas are listed below under three headings: Executive Management, Factory Development and Operation, and Supplier Relations. One or two key guidelines are listed for each area. Appendix B provides the details, along with a few supporting guidelines.

- Executive Management:

  Area 1 Sponsoring Software Quality Improvement

  1.1 Deploy organization-wide a policy on software quality improvement for suppliers.

  1.2 Ensure the organization’s own practices are consistent with the policy for suppliers.

- Factory Development and Operation:

  Area 2 Factory Planning, Engineering, Specifying

  2.1 Develop requirements/specifications for software, including quality specifications.

Area 3 Procuring Equipment and Systems

- 3.1 Use your policy on software quality improvement in the procurement process.
- 3.2 Base all procurements that include software on stated requirements/specifications.

Area 4 Integrating Equipment and Systems, Testing, Qualifying

- 4.1 Share with suppliers data about software's performance during integration.
- 4.2 Coordinate with suppliers their responsibilities for software testing prior to delivery.

Area 5 –Operating the Production Factory

- 5.1 Collect software performance data for production operations; share it with suppliers.
- 5.2 Deploy the Overall Equipment Effectiveness metric.

- Supplier Relations:

Area 6 Providing Leadership by Requiring Software Improvement

- 6.1 Require software quality improvement by reviewing quality indicators (the 4-Ups).

Area 7 Developing and Supporting Software and Systems (Internal Groups)

- 7.1 Apply your software quality improvement policy to all internal software groups.

Area 8 Accelerating External Suppliers' Progress in Software Improvement

- 8.1 Do software quality improvement projects with key suppliers to accelerate progress.

Practices developed from the guidelines often reflect a specific stage of the factory lifecycle. An example of this is the guideline from Area 2:

> Develop requirements/specifications for software, including quality specifications.

When planning a wafer fabrication facility, an important practice is putting software requirements and specifications under configuration management as they are developed. Later, this facilitates easy and accurate uses of them for procurement and acceptance testing. But once a fab is up and running, it is important to consistently update the configuration management files. That keeps the data current for use in trouble reporting and problem resolution when information is needed about changes and enhancements to the software.

As a second example, consider the first guideline from Area 3:

> Use your policy on software quality improvement in the procurement process.

When making initial procurements for a fab, the practice of assessing the software capabilities of candidate suppliers is important; findings from the assessment are used in supplier evaluations and selections. After a supplier has been selected and has delivered products for months or even years, the priority practice is to regularly participate in reviews that assess the supplier's software quality improvement goals, work plans, and progress. In this way, the customer supports software quality improvement and accelerates the supplier's progress.

The software practices an organization derives from the guidelines become integral parts of various business processes—processes like the fab planning process, the equipment and system procurement process, the equipment and system integration process, and the process for supporting use of products in the production fab. Taken collectively, software practices across the organization and throughout the factory lifecycle represent the manufacturer's software capability. The guidelines, and this document as a whole, deal with establishing and improving these individual software practices. They are not concerned with the larger issue of establishing, managing, and improving the processes within which the software practices are embedded.

In deriving and executing software practices based on the guidelines, teamwork, and communication are the keys. The following are examples:

- Saying that software quality is important (the SQI Policy) and requiring suppliers to produce objective indicators (4-Ups Metrics) of software's level-of-quality "sets the bar." Suppliers need to know the bar is set, and that you are very serious about it.
- Mature software practices by a manufacturer says to suppliers: the customer is "walking the talk."
- When a customer and supplier have a shared understanding of the suppliers' role in producing quality software, they are able to communicate much more effectively about any aspect of the process that's causing a problem—always the first step toward problem resolution.
- A shared understanding of the supplier's role positions a customer to work with strategically important suppliers in ways that accelerate their progress in improving software.

3.3 Sponsors, Advocates, and Activists for Software Quality

How does a manufacturer organize and staff to assure software quality?

Each SEMATECH member company has devised its own organizational structure for the functions listed in Figure 5 (see Section 3.1). You company's structure, together with your unique corporate culture and business plan, influences how you go about assuring quality in the software you acquire.

Setting such differences aside, your approach should incorporate the following essential functions:

- An executive "sponsor" of factory software quality
- A properly chartered "advocate" for factory software quality
- Various "activists" in non-software functions whose actions impact software quality

Table 5 lists some responsibilities associated with each of these functions. Material in both Section 4 and Section 5 provide additional background information related to many of the responsibilities listed. Your company is well advised to let the form of its implementation follow the three functions identified below.

Table 5 Manufacturer's Responsibilities for Software Quality

Sponsor	Sponsors/owns the effort Accountable for results

	Has/supports the vision for software Sets goals and priorities Provides the resources and charters Removes obstacles Participates in reviews, planning, course adjustments Monitors progress
Advocate	Has/supports the vision for software Identifies/establishes needed practices Coordinates the effort Accelerates progress Assures quality (in the name of the sponsor) Accountable for results
Activist	In each functional area – the key to success Owns/performs software practices

4 UNDERSTANDING THE SUPPLIERS' ROLE IN SOFTWARE QUALITY

Not communicating with suppliers about software
can increase costs for your business.

Customers may think suppliers have complete responsibility for "getting software right" when the truth about this aspect of your relationship with suppliers is as follows:

> **CUSTOMER/SUPPLIER TEAMWORK IS ESSENTIAL FOR FACTORY SOFTWARE EFFECTIVENESS. (Fact #3)**. Each semiconductor factory has ground rules for how it operates, and to a considerable extent the ground rules are implemented with software. To get software performing just like your staff wants it, you and your suppliers (both internal and external) must work together extensively. So you, as the customer, must invest the resources required to make your side of the customer/supplier equation effective. And you should require a similar investment by your suppliers.

A supplier works with software in ways quite unlike those of the customer. For example:

• Customers state requirements/specs.	Suppliers produce software that meets them.
• Suppliers produce/deliver products.	Customers integrate/qualify the products delivered.
• Customers define/request high quality.	Suppliers produce/provide level-of-quality indicators.
• Customers encounter problems.	Suppliers diagnose problems and deliver corrections.

Software has been called "executable knowledge" and in semiconductor factories, the knowledge content of software is certainly improved by the knowhow, experience, and insights of both parties. And communicating about software effectively is essential, especially on the subject of improving its quality.

This section is an overview of what suppliers do to improve quality in software they produce. Written for the customer, this material's purpose is to promote more customer/supplier dialogue about software, and to make their communications increasingly effective. This view of a supplier's role in improving software quality is simply an elaboration of the guidelines in the Policy for Software Quality Improvement (see Section 2.2 and Appendix A).

> ***THE RECOMMENDED RESPONSE FOR A SEMATECH MEMBER COMPANY:*** Manufacturers are making a mistake when they do not communicate adequately with a supplier about software before product delivery. How you deal with software prior to delivery—during planning and procurement—has a major impact on the quality of delivered software. So, your executive sponsor for software quality should take the following actions:
>
> - Communicate the company policy on software quality for suppliers—a statement describing what quality is, its importance to you, and how you expect the supplier to produce it.

- Require suppliers to provide software quality indicators as objective evidence of the level of quality and of progress being made toward software excellence.
- Ensure that suppliers are assessed and evaluated for their software capabilities, and that the findings are used in qualifying and selecting suppliers.
- Sponsor reviews and work projects with key suppliers to accelerate the improvement of quality in the software they produce.

The information in this section supports implementing this recommendation by addressing the following questions:

- What roles in a supplier organization effect software quality? (Section 4.1)
- What indicates progress in the work of improving software quality? (Section 4.2 and Appendix C)
- How does a supplier continuously improve software quality? (Section 4.3)

Customer/supplier relationships in this industry often are measured in years, and increasingly software matters have a prominent place in their interactions. A shared understanding of the supplier’s role in producing high quality software contributes to an effective and long-lasting relationship.

4.1 Software Quality and the Supplier

What roles in a supplier organization effect software quality?

Communicating with suppliers about software quality can be effective only if it involves the right people—those who affect software quality. And not all of those people are software specialists or in the software organization. This section identifies some key supplier roles for communications about software quality.

The core functions in producing software are as follows:

- Gather and develop requirements and specifications
- Create a design, or fit a modification into the existing design
- Produce executable code and integrate various elements of code
- Test the code and verify that it satisfies the requirements
- Deliver the code to the customer

Findings from assessments of software groups in many industries show that no more than one in three organizations perform these basic functions well. In small software organizations, an individual often performs several functions. Even so, it is important to have and follow an established process. The supplier’s software process will vary according to the type of software production:

- New development of products, systems, and software
- Product evolution and enhancement, with sustaining engineering for the software
- Software re-engineering and reuse
- Software maintenance and customization

Beyond software engineering’s core are several key support functions that are critical and integral parts of the software process:

- Problem reporting and change control
- Configuration management that includes version, release, and distribution control
- Objective verifications and validations of products and processes

Beyond the software support functions are roles that provide the leadership, experience, and drive needed to continuously improve software quality. How these roles are implemented varies widely, but without them a supplier does not make progress toward software excellence. They include:

- A vision with a focus on process and a drive to continuously improve
- Work done on policies and procedures
- Training and staff development
- A working group on metrics, measurement, and historical data
- A tools and methods working group

Finally, at least five functions outside the software organization directly impact software quality:

1. Executives and senior managers
2. Product and system engineering
3. Marketing, sales, and field support
4. Purchasing, finance, and legal
5. Organization-wide quality engineering

There is not a one-size-fits-all software process for suppliers. Each defines its own process and is responsible for maturing it so that its software meets the quality specifications that you, the customer, set. The 1996 edition of the *SPI Guidelines* discusses these roles in a supplier organization in more detail [3].

4.2 What Suppliers Do to Improve Software Quality

What indicates progress in the work of improving software quality?

By complying with the Policy on Software Quality Improvement (Section 2.2 and Appendix A), suppliers improve the quality of their software. That is a short response to this section’s question, and in a sense it is complete.

This section profiles a path to software excellence that simply elaborates the guidelines that the policy states. This path to excellence consists of a starting point and six stages that suppliers move through. The stages are simply a logical way of explaining the work that is involved in software excellence. Material in Appendix C identifies and profiles the methods most used in this industry for more objectively measuring an organization’s progress.

The Path to Software Excellence [4]

A supplier achieving software excellence and an athlete training for Olympic competition have some things in common: careful preparation and daily execution. Software excellence and Olympic success both are preceded by several years of coaching and hard work. Neither can be achieved by "silver bullet" methods or by those who lack the dedication to succeed. In both cases the rewards for success are great.

Following is a three-year development regime for traversing the path to software excellence. The path is discussed in terms of its starting point and six categories of work that build upon one another:

• Starting Point	Assessments, baselines, benchmarks are the starting point for improvement, and a periodic checkpoint for course correcting.
• Stage 1	Getting management "on board" is job # 1—and this includes ongoing sponsorship of the effort, process management, and project control.
• Stage 2	Improvement's core is using established, shared methods and processes throughout the organization, with consistency and effectiveness.
• Stage 3	New tools and approaches are selected and implemented effectively, when the changes produce observable benefits.
• Stage 4	An infrastructure of specialty skills and supporting services matures an organization's software capability.
• Stage 5	Mastering the "reuse of software" is the final challenge for organizations that produce software with consistently high quality.
• Stage 6	Organizations that have achieved mastery in each of the other stages are in position to focus on industry leadership.

These work categories, referred to as stages, are presented in a serial progression. But a supplier may not experience them in exactly that order or as a simple sequence. The time frames (six months for each stage) are approximations, but achievable if adequate resources and expertise are applied to the work at hand. The estimates of expense are primarily to convey the fact that this is a serious investment. Getting a return on that investment is a legitimate focus for management.

Starting Point

All people are more or less blind to their own shortcomings. That means the first challenge in achieving software excellence is to baseline current practices. A formal assessment that includes some outside participants provides the needed information. The assessment methods most frequently used in this industry are the SEI Capability Maturity Model for Software [5], the ISO 9001 Standard as explained for software by ISO 9000-3 [6], and Module 3 of SEMATECH's Standardized Supplier Quality Assessment [7]. Appendix C has information about each method.

A supplier committed to improvement combines the assessment findings with the following:

- A quantitative baseline that includes the current product portfolio, its quality levels, and the degree of customer satisfaction it has received.
- A profile of the current software process with measures of performance and productivity ranges.

This information about the supplier shows where things currently stand. The method used as a basis for the assessment identifies "next step improvements," goals to set, and the metrics of progress to the goals.

Two other accomplishments completing the launch of the initiative include the following:

- A benchmark against other organizations that represent a standard of excellence
- A plan for actions that move the supplier to its quality goals.

The supplier who improves successfully returns to this initial starting point periodically to recalibrate and reorient its improvement effort.

Stage 1: Management Issues (Start to Six Months)

Students of software problems have noted that, as a class, managerial problems are more common and more serious than any other kind. This observation is also true for fields other than software. Indeed, the well-known management consultants W. Edward Demming and Joseph M. Juran both have observed that fully two-thirds of manufacturing problems can be traced to management, with only one-third to workers [8].

Ongoing and successful software improvement always has the active sponsorship of executive managers. Active sponsorship means the manager has ownership, includes it in the organization's agenda, gives it priority, issues charters for the work, and allocates resources to it. Completing this package of essentials is the vision of an improvement champion or chief change agent who can articulate the vision and motivate others to participate in the effort. Without this kind of foundation, the investment of effort and other resources is futile.

Beyond such a foundation, a supplier at Stage 1 concentrates on getting its software staff competent in areas such as project planning, estimating, scheduling, and control; productivity and quality measurement related to software; and customer interactions related to software requirements and product delivery/acceptance. The supplier committed to moving along the path also makes sure that its management team is appropriately equipped with modern techniques and tools for planning and controlling projects.

Stage 2: Structured Methods (From 6 to 12 Months)

With its foundation in place, an improving software organization focuses on methodologies—refining its use of current and new methods to improve performance and productivity. Methods must be well established; then tools must be selected to support the methods.

Two top methods in successful software organizations support defect removal and prevention. All companies performing in the upper 10% of software quality use formal inspections of requirements, designs, and code to identify and remove defects. Inspections have the highest efficiency rating of any defect removal technique; they are twice as efficient as most forms of testing. The premier defect avoidance method is root cause analysis. It prevents defects by analyzing the causes of defects as they are detected, and making process changes that reduce reoccurrences or avoid them completely.

Other methods that characterize an improvement culture deal with testing, defect reporting and tracking, change and version control, configuration management, product builds, and release/distribution controls. Organizations supporting large amounts of legacy code work on reducing maintenance costs with methods for complexity analysis, code restructuring, and other remedial approaches. Rounding out Stage 2, an improving software community is marked by a staff that is well trained in and effectively using all of the methods appropriate for its situation. Management supports the work of Stage 2 by authorizing the staff time required to explore and pilot new methods, and/or to refine uses of current methods.

Stage 3: New Tools and Approaches (From 12 to 18 Months)

An organization at this stage on the path to excellence examines, experiments with, and selects new tools and approaches. It uses as candidates the technologies that have come on the scene during the past ten years, or are just emerging. The measurement program put in place in Stage 1 is used in demonstrating effectiveness, or a lack of it, in any of the tools or approaches considered. By this stage, the managers and staff have the basis needed for definitive evaluations of any tool or technology that promises a benefit.

Many studies of organizations automating or otherwise supporting software development have reached two similar conclusions:

1. Unless as much is invested in staff training as in the tool itself, the results will be marginal.
2. Experienced, well-trained professionals get more value from tools than do inexperienced personnel.

For example, both managers and technical staff members undergo extensive training in tools acquired at this stage. Many organizations schedule 10 to 15 training days per staff member during this stage, paying from $4000 to $6000 per staff member.

New tools and approaches are not considered nearer the outset because an organization needs the maturity developed in Stages 1 and 2 to do Stage 3 well. Similarly, implementing the decisions made during Stage 3 often takes months, even years, to complete, thus overlapping work in the subsequent stages.

Easily, this stage is the most expensive. A fully integrated tool suite supporting the entire life cycle offers considerable functionality and an attractive benefit potential. At $15,000 to $50,000 per seat, it is a significant investment—not one to make without substantial analysis of return on investment (ROI).

Stage 4: Infrastructure (From 18 to 24 Months)

At Stage 4, the focus is on an infrastructure for excellence—the range of skills and the continuing effort needed for the organization’s software capability to be healthy and state of the art. For example, the management in an organization at this stage recognizes the need for specialization in software roles. It supports practices that formalize and institutionalize the hiring and retention of specialized skills for which a demonstrated need exists. In all human activities, including software, specialists tend to outperform generalists. Larger organizations have more than 50 different kinds of software specialists. This range of expertise includes skills related to networking, user interfaces, requirements elicitation and management, system and software design, testing, configuration management, equipment and software interfaces, and quality assurance, to name a few.

Smaller organizations cannot have on staff every kind of specialist, so another critical aspect of a Stage 4 organization is established staff education policies that provide an annual training target for management and technical staff.

Once an organization completes this stage of its transition to excellence, it is a candidate for a Level 3 rating on the SEI scale for measuring software capability maturity. It also should be able to pass ISO certification. (See Appendix C for details about these measurements of software quality.)

Stage 5: Reusable Software (From 24 to 30 Months)

Full software reusability programs tend to have the highest ROI of any technology since software began. A single project can find ten or more software artifacts that have reuse potential. The value increases for each artifact identified and qualified for reuse.

However, a strong caution is indicated: successful reuse demands zero-defect materials. It is unsafe and uneconomical to reuse materials that have defects that make them unreliable. This is why reusability is offset by two years from the start of your software improvement program. It can take that long before an enterprise is good enough to develop anything worth reusing.

By the end of this period, you should be establishing goals of exceeding 50% reusable materials in every application that your enterprise develops. The greater the volume of reusable, certified material, the greater the productivity and quality levels will become.

Costs for creating reusable plans and estimates are comparatively inexpensive, assuming that you already own estimating tools that can support template creation. Since reusable materials must be certified to very high levels of reliability and quality, additional expense should be anticipated to obtain those levels. You also will incur costs educating staff members and managers about the fundamental principles of reusability and the mechanics of utilizing your library.

Stage 6: Industry Leadership (From 30 to 36 Months)

By the end of three years of concentrated effort, an enterprise should be good enough to be considered as “best in class.” Achieving this level puts quality and productivity rates in the upper 10% of all industries.

Specific numeric goals associated with such leading-edge enterprises include defect potentials of less than 1.0 defect per thousand lines of code and defect removal efficiency levels consistently higher than 95% and perhaps exceeding 99%.

For organizations that achieve these accomplishments, software will no longer be regarded as a troublesome, out-of-control technology. Schedules will be much shorter than today’s norms, and predictable within a margin of about ±5%.

Costs will be lower than today, and predictable within very small margins. Quality will be high enough so that post-release defects seldom occur, and this in turn will raise user satisfaction levels to very high levels.

Since working on successful projects in well-managed enterprises is a pleasant experience, staff morale should be very high, also. There are enormous benefits to be had from bringing software under full control, and severe hazards for the enterprises that cannot accomplish this.

By now, your managers and technical staff members will have experienced three years of very hard work. The range of expenses over this three-year period may range from $30,000 to $80,000 per staff member when equipment, tools, and training are included. These amounts could increase with improvements in office space and office ergonomics.

Software excellence is not easy to achieve and is not inexpensive. But if your enterprise depends upon software for competitive advantages in the products that you build and for efficiency in operation, do you want to move into the 21st century with anything less than a world-class software organization?

A Final Word about the Path to Software Excellence

Knowing where your organization is (as a starting point) and having the commitment and active support of an executive sponsor are two essentials that make obvious a third one: a compelling vision of semiconductor manufacturing driven with high quality software, a vision materialized with stretch goals and achievable objectives, a vision that spreads from a few to the many required to make it happen. That is what provides the dynamics needed to move along the path to software excellence.

4.3 Sponsors, Agents, and Resources for Continuous Improvement

How does a supplier continuously improve software quality?

Section 4.2 profiles the path to software excellence, and Appendix C describes some tools for measuring with objectivity an organization's position and progress along that path. (These measurement tools also suggest next steps for an organization to take to continue its advance.) But actually moving an organization down the path takes determination, motivation, some knowhow, and hard work by a lot of people. It also takes an effective mechanism for driving the organization forward.

One of the strongest motivators for suppliers is their customers. When a customer talks about software quality or (any other subject), suppliers typically listen and then take action. Communications on the subject of software quality are credible when the customer is clear and realistic about what is involved in continuously improving software and is even familiar with the specific driver mechanism and measurement tool a supplier uses. This section is an overview of a mechanism for continuous software improvement that the SPI Project regularly recommends to suppliers. It is also representative of a whole class of approaches.

The IDEAL model [9] provides a usable, understandable approach to continuous improvement. It is a simple cycle of five phases:

•	I	Initiating	Laying the groundwork for a successful improvement effort.
•	D	Diagnosing	Determining where you are relative to where you want to be.
•	E	Establishing	Planning the specifics of how you will reach your destination.
•	A	Acting	Doing the work according to the plan.
•	L	Learning	From the experience and improving your ability to change.

Figure 6 pictures the Initiating, Diagnosing, Establishing, Acting and Learning (IDEAL) model with the major steps detailed within each of the five phases. In its full form, this is a model for making major technological change in an organization. With appropriate scaling and application, it is also effective for making smaller and more limited change. The key is in the flow of the model and in its repeated and disciplined use. As an organization develops the habits and skills that come from using the model, progress toward software excellence will accelerate.

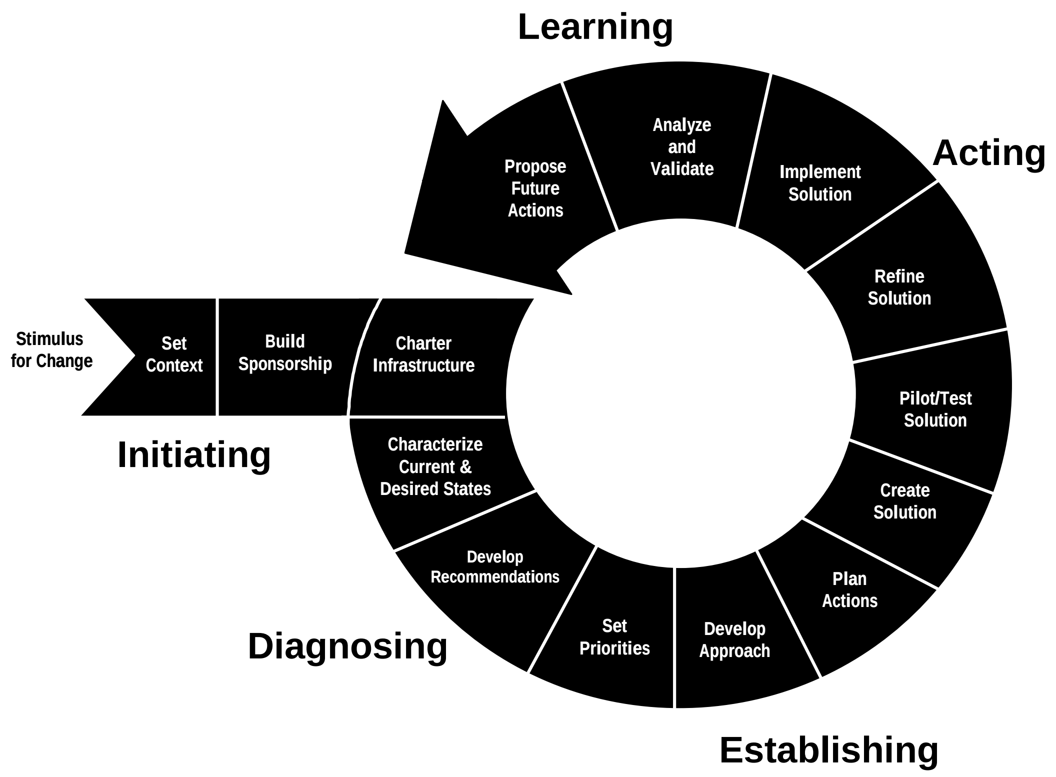


Figure 6 IDEAL Model for Continuous Improvement

5 AN INFRASTRUCTURE THAT ACCELERATES PROGRESS

Underestimating the scope of change required to improve software quality is hazardous to your business.

The idea that you can quickly fix factory software with a “silver bullet” solution is deeply embedded. Such solutions always apply to a particular aspect of the problem, and typically are promoted in terms of how quick, easy, and inexpensive they are to implement. The bad news is that each fix is, at best, only a part of the solution. Improving software quality is more like preparing for Olympic competition than winning one particular race. It takes commitment, some expertise, a period of time with perseverance, and the investment of some resources. The good news is that:

> **ASTUTE INVESTMENTS IN SOFTWARE QUALITY YIELD POSITIVE RETURNS (Fact #4).** The competitive nature of semiconductor manufacturing means that actions taken, time used, and dollars spent constantly must be evaluated in terms of the contribution made to a factory’s yield and profitability. You can improve both measures by improving software quality. It is hard and challenging work, and takes time, discipline, experience, and resources. However, these investments, astutely made, can yield (to both customer and supplier) positive returns of 5:1, based on software industry experience.

The need to change how people think about software—and to change what some people producing it actually do—means that improving software quality involves some “people issues,” work that goes well beyond the technology considerations involved.

> ***THE RECOMMENDED RESPONSE FOR A SEMATECH MEMBER COMPANY:*** The view that producing software equals writing and testing code is wrong. Today, the large organizations producing high-quality software have several dozen kinds of specialists doing it. Many suppliers in this industry are small enterprises with staffing limitations. A policy and purchase specification for high-quality software challenges the collective expertise of a supplier’s staff. So it is recommended that you address the software challenge with actions on two levels:
>
> 1. Launch and sustain an initiative that improves software quality in your own factories.
> 2. Collaborate with other member companies to leverage the investment each company makes.

The information in this section supports implementing this recommendation by addressing the questions:

- Who wants what from software? (Section 5.1)
- Is there a key to success in improving software quality? (Section 5.2)
- What is the strategy for collaboration by SEMATECH constituents? (Section 5.3)
- What are SEMATECH constituents doing to improve software? (Section 5.4)
- How will these guidelines for customers advance the work on software? (Section 5.5)

Executives understand investments, especially investments in

- Facilities and capital equipment that increases manufacturing capacity
- Skilled people with expertise who are able to materialize the company's future

But few executives in semiconductor manufacturing are experienced investors in software, and even fewer have experience with investments that improve software quality. Yes, there are many pitfalls and few guarantees. But by working together, and focusing on the basics and what others have shown to be useful, rapid progress is possible.

5.1 Shared Goals

Who wants what from software ?

Every semiconductor manufacturer wants "negligible factory downtime due to software." This aspiration is motivated by big payoffs in production operations. It also applies to the equipment-integration-and-certification phase. So Goal #1, shared by all the member companies, is to eliminate software-induced downtime in the factories. That means software that, element by element and as a composite, is as follows:

- Highly reliable based on objective measurements
- Qualified for production use without delays
- Rated by the operations staff as "very usable"

As already discussed, achieving that goal is a process, not a single accomplishment. It means suppliers (both internal and external) must mature their software capabilities. Following are indicators that show a supplier is maturing:

- Software releases are delivered on time, as scheduled.
- From release to release, delivered software has fewer known defects.
- During ramp-up, software is qualified for production use in progressively shorter times.
- In production operations, software failure rates and repair times decrease over time.

But you the manufacturer, as a customer of these suppliers, also must mature your software practices. Indicators of your maturity include the following:

- You develop requirements and specifications for software and review them with suppliers.
- You state and enforce software quality specifications.
- You evaluate supplier performance as a software provider and use that in choosing suppliers.
- You assure the quality of software before accepting it, rejecting any that fails your quality specifications.

Maturity takes time and resources. Any semiconductor manufacturer wants to minimize the time and resources used without putting in jeopardy progress toward Goal #1. That means shared Goal #2 is to leverage each member company's investment in software quality through collaborations that accomplish the following:

- Spread workload and expense
- Share lessons learned
- Overcome sluggishness and resistance
- Accelerate progress

Work toward these two shared goals is underway. The balance of Section 5 reviews what is being done and recommends how member companies can contribute to the ongoing effort.

5.2 Reaching the Goals

Is there a key-to-success in improving software quality ?

The key to improving software quality is commitment, sponsorship, and participation by executive level management.

Backing by executive management is essential, and it is the *key to getting on the path to success* and staying there—but it is not the key to success itself. Success here is a process, not an event, and there are several factors that ensure a productive and effective process (after you are on the path). Figure 7 shows that change is the essence of improving software quality, and that the targets of change are people, and the processes and technologies they use in their work.

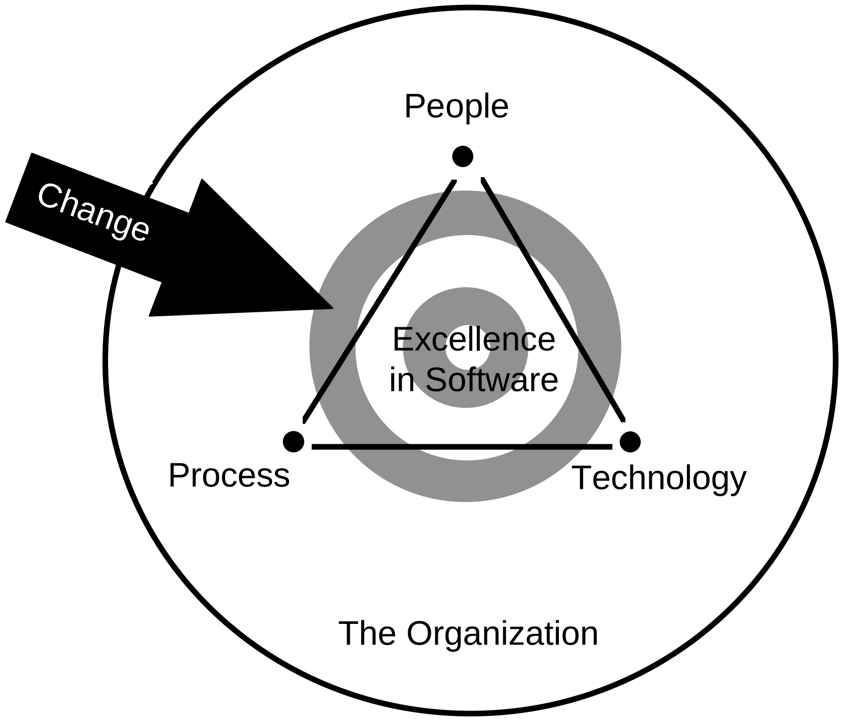


Figure 7 Factors in Moving Along the Path to Software Excellence

The people work in an organizational or business setting, so the major factors in moving down the path to software excellence include the following:

- Factor 1: Change—the path to software excellence is about change
- Factor 2: People, process, and technology—the three targets of change
- Factor 3: Interrelated targets—changing any one target has implications for the other two
- Factor 4: Organizational change—it has five essential elements: vision, skills, incentives, resources, action plan

Figure 8 pictures the outcome when any of the five essential elements needed for successful organizational change is missing.

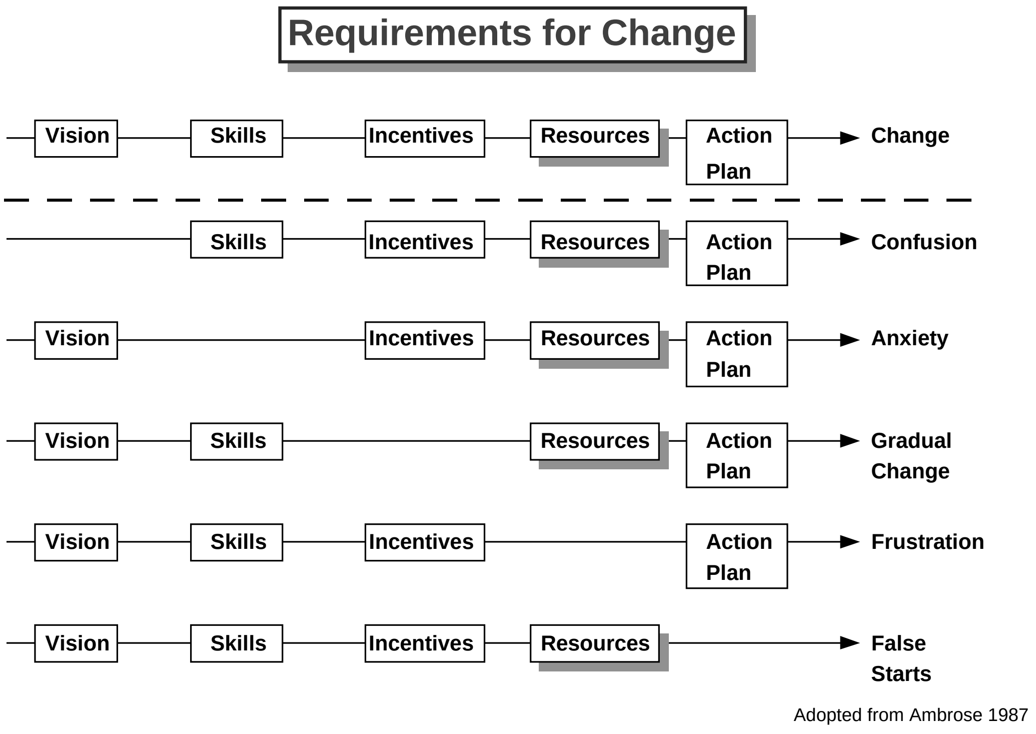


Figure 8 Five Requirements for Successful Organizational Change

5.3 Strategy for Joint Action

What is the strategy for collaboration by SEMATECH constituents ?

The strategy for a joint effort by SEMATECH constituents to improve software quality has seven elements:

1. **Policy on Software Quality, with Metrics**. Develop and deploy a policy that puts a high priority on improving software quality, profiles how a supplier is to do it, and identifies metrics for showing quality improvement. Manufacturers deploy the policy throughout their operations, and establish its use through various interactions with suppliers.
2. **Commitments to Software Quality**. Solicit and record public statements by executives and senior managers (in both customer and supplier organizations), statements of commitment to comply with the policy (suppliers) or be consistent with its provisions (customers).
3. **Assessments of Software Capability**. Using a standardized model of software capability, objectively assess individual suppliers, using the assessment findings as a basis from which to develop baselines, benchmarks, and improvement plans.

4. **Established Software Processes**. See that each supplier establishes organizational processes for software that are:
   - Well defined and shared throughout the organization
   - Consistently used and proactively managed
   - Effective and validated with indicators
   - Disciplined but adequately flexible
   - Appropriately supported by an infrastructure of specialized functions
5. **Continuous Improvement**. Motivate and help suppliers and customers alike to build and operate an infrastructure that drives continuous improvement of the practices/processes by which they specify, produce, support, and manage software.
6. **Quality Indicators**. Motivate and help suppliers produce objective indicators of both software quality and process maturity, ensuring that suppliers regularly provide these indicators to their customers, and that customers use them to accelerate improvement of software quality.
7. **Progress Reports**. Collect quality indicators from suppliers and customers, and at regular intervals report the status and progress made improving software quality and capability.

Section 5.4 identifies current activities and recent accomplishments in support of this strategy, and discusses the member companies' role in executing it successfully. Section 5.5 discusses the role these guidelines can play in executing this strategy.

5.4 Operations and Accomplishments

What are SEMATECH constituents doing to improve software ?

During 1997, work in several SEMATECH programs supported the plan for improving software quality outlined in Section 5.3. Following are highlights from this work, summarized to show the role of member companies and opportunities the members have for additional participation.

- **SEMATECH Software Process Improvement Program**. The SPI Project has led a broad-based cultural change to drive software quality improvement in the semiconductor industry. The major activities of this program include the following:
  - Tactical focus areas—the "Big 6" issues—with the goal of fixing today's most pressing problems:
    - ◊ Software testing
    - ◊ Configuration management (including version and release control)
    - ◊ Peer technical reviews and inspections
    - ◊ Defect reporting, tracking, and closure
    - ◊ Requirements elicitation and management
    - ◊ Project planning and tracking

- Strategic focus areas, with the goal of building infrastructure to improve tomorrow’s software:
    - ◊ Software Quality Improvement (SQI) Policy
    - ◊ Measurements and metrics (the 4-Ups Metrics in the SQI Policy)
    - ◊ Software process maturity and assessment
    - ◊ Goal definition and action planning
    - ◊ Software engineering training
    - ◊ Reusable resources, communications, and referral services

- **Tactical improvement tasks**. Member companies have launched more than ten short-term tasks to improve software quality. In each, a member company works with a key supplier to accelerate the progress being made to software excellence. These tasks also develop and exercise the interface for communications about software between manufacturer and supplier. The member company publishes the results for the benefit of other members and suppliers.
- **Multi-customer meetings with suppliers**. Representatives from several member companies hold scheduled meetings with key suppliers to review the supplier’s software performance. Managers from each software group in the company hear about customer problems with specific software releases, provide data for 4-Ups Metrics, and review with customers their plans for further improvement of software quality. During 1997, four such meetings are being held with major suppliers.
- **Equipment Productivity Improvement Teams (EPIT) Program**. The EPIT Program has conducted over 100 member company/supplier meetings focused on improving the performance, reliability, and efficiency of specific semiconductor manufacturing tools. The meetings provide an opportunity for sharing best practices and lessons learned, and reaching agreement on a prioritized list of required tool improvements. SEMATECH funding is made available as required to implement and verify these improvements. This program recently was expanded to include software as a focus topic, and preparations are underway for several “Software EPIT” meetings that will drive improvements in the software quality and reliability of specific tools.
- **Seminars on software for senior managers**. This seminar is for executives and senior managers, whether or not they have experience dealing with software issues. It is designed to cultivate and secure management commitment to sponsor improvement initiatives. It includes an overview of software issues confronting the industry, the executive’s critical role in improving software quality, and the work being done (and resources available) to improve software quality. Public offerings of this half-day seminar are provided under SEMI/SEMATECH sponsorship on both the East and West coasts. Other suppliers have scheduled presentations in-house for their own management team.
- **Software Quality Improvement Policy**. The Policy and the four key metrics it identifies were developed through a joint effort of the SPI PTAB and SPI Project. Communicated to all members of SEMI/SEMATECH, it is now being deployed by the member companies throughout their procurement and other operations. It is used as a basis for the guidelines for manufacturers in this document.

- **Standardized Supplier Quality Assessments**. Member companies request that key suppliers perform a self assessment, using the SSQA method. Assessors from several member companies form teams that validate the supplier's self-assessment. Based on the assessment findings, the supplier develops an improvement plan which the assessment validation team reviews for effectiveness. In many cases, only SSQA Module 3 is used to assess the software development and support groups. The level of activity during 1997 is as follows:
  - 10 member company-sponsored full assessments
  - 25 SPI Project sponsored assessments (software only)

  Projections in the National Technology Roadmap for Semiconductors indicate that the ratings from these assessments must improve significantly over the next three years.
- **SQI Training Courses for Suppliers**. Public offerings of a curriculum of six courses for software engineers and their managers (from suppliers and member companies) is offered at both East and West Coast locations. Some suppliers schedule courses on-site where they are customized to the particular supplier's training needs. The level of activity during 1997 is as follows:
  - 25 offerings of these courses
  - More than 875 student-days of training
- **Neutralizing the Year-2000 Problem**. The year-2000 software problem (also known as the "millennium bug") arises from the manner in which dates have been written and manipulated in software systems since the earliest days of stored program computers. Motivated to conserve space, many programmers elected to represent the year as two digits (97 represents 1997). With the dawning of the year 2000, many of the systems built around this convention will fail unless all software containing this "bug" is revised, tested thoroughly, and installed on all equipment prior to December 31, 1999.

  The member companies asked SEMATECH to initiate a project to help facilitate the correction of software running in equipment installed in the member company factories. The members will identify the equipment and software releases they want the project to monitor. SEMATECH will create a central database for monitoring all identified software versions.

  Additionally, the member companies will create a set of standard test scenarios that will be given to the equipment suppliers. As the suppliers execute these tests and certify their software to be year 2000-ready, the database will be updated with this information. That will allow each member company to track which of its tools will successfully bridge the millennium and which are yet to be addressed. Each member company also will be asked to perform their own test on a subset of the equipment to validate supplier information.
- **Tool Performance Tracking Program (TP2)**. The TP2 Project was launched during 1997 at the request of member companies. Its objective is to improve the OEE of fabs through better data collection that allows more useful analyses. The data improvements result from using a common solution (based on the SEMI E58 Automated Reliability, Availability and Maintainability Standard [ARAMS]) and automating data capture. Automation will increase data accuracy by eliminating or greatly reducing manual input. The project has plans for both "on tool" and "off tool" implementations.

- **SPI PTAB**. This steering committee for the SPI Project consists of representatives of the member companies, several key suppliers, and other constituent organizations. They meet two or three times each year to review the SPI Program and provide guidance in such areas as identifying key suppliers, establishing and completing tactical software projects, and providing overall guidance to the project.

Other activities that support the software improvement strategy are as follows:

- **SEMI/SEMATECH's Software Quality Program**. This program is just being launched. It represents a collective recognition by the suppliers that improving software quality has high priority and an acceptance of responsibility for getting the job done within their companies.
- **The National Technology Roadmap for Semiconductors (1997).** This 15-year technology projection by the industry clearly states the critical role that improving software quality has in maintaining the historical cost learning curve that has driven industry growth for 25 years.
- **The SEMI Software Reliability/Quality Task Force (E10 Subcommittee, ARAMS).** The work of this task force is on the critical path to getting reliable data from fab operations about overall equipment reliability, and the extent of the software problem in production fabs.

5.5 SEMATECH Members and these Guidelines

How will these guidelines-for-customers advance the work on software?

This document presents four facts about software's relationship with semiconductor manufacturing. With each fact there is a recommended response. These responses, together with the guidelines in Section 3.2 and Appendix B, point to actions that constitute your responsibilities for software quality.

In this industry, software excellence cannot be accomplished unilaterally; both manufacturer and supplier must be deeply committed and increasingly competent in software. If either is not, it creates problems for both and software quality suffers, along with productivity and profitability.

To recap, each of the four facts (with a summary of its recommended response) is listed below. The recommendations, like the guidelines, must be applied to your organization in light of your business plans and culture. By providing feedback from your use of this material, you will accelerate progress toward the two shared goals: negligible factory downtime due to software, and maximum return on the investment you are making in software quality.

Fact #1: Software Competence Is a Necessary Business Asset. To support your factory's productivity and profitability, a mature, well-rounded competence in software practices is essential.

The Recommended Response

Each of your manufacturing operations should do the following:

- Appoint a manufacturing executive who owns and actively sponsors developing software competence that improves software quality

- Charter a senior person as the advocate for factory software quality. The advocate should:
  - Establish and mature specific software practices the organization needs
  - Develop a well-rounded competence in software acquisition
  - Assure the quality of acquired software (see Fact #2)
  - Work with suppliers on improving software quality (see Fact #3)
  - Achieve the goals set for software quality improvement (see Fact #4)
  - Work jointly with other member companies on software issues (see Fact #4)

Fact #2: Acquiring Customized Software Includes Assuring Its Quality. It is a software customer's responsibility to produce assurance that delivered software meets your specifications, including quality requirements.

The Recommended Response

Assure the quality of acquired software, with practices that include the following:

- Develop and document requirements and specifications for factory software
- Define what you mean by quality in software and the level of quality you expect
- Reject software that fails to meet your specifications and quality standards

Fact #3: Customer/Supplier Teamwork Is Essential for Factory Software Effectiveness. As the customer, invest the resources required to make your side of the customer/supplier equation effective, and require your suppliers to make a similar investment.

The Recommended Response

Your executive sponsor for software quality takes the following steps, directing each toward both external and internal suppliers:

- Deploys and enforces a company policy on software quality for suppliers
- Requires suppliers to provide software quality indicators
- Assesses and evaluates suppliers' software capabilities
- Sponsors reviews and work-projects with key suppliers to accelerate quality improvement

Fact #4: Astute Investments in Software Quality Yield Positive Returns. Such investments, if managed effectively, can yield positive returns of 5:1 (based on software industry experience) for both the customer and supplier.

The Recommended Response

Address the software challenge with actions on two levels:

- With an internal initiative, establish mature software practices that improve software quality.
- Work jointly with other member companies in ways that leverage each company's investment.

Major elements of the manufacturer's internal initiative include:

- Goals with metrics for measuring progress to the goals
- Adequate staff and funding
- Strategies, milestones, and plans for achieving the goals
- Reviews that track the progress made and make course corrections as necessary

6 REFERENCES

[1] H. Wohlwend, et al., *Software Issues, Practices and Reliability Within the Semiconductor Industry: A SEMATECH White Paper*, Technology Transfer #96033106-TR, SEMATECH, Austin, TX, May 31, 1996.

[2] This data is from soon-to-be-published material for a SEMATECH white paper., F. Langner, *Software Reliability in Semiconductor Manufacturing Equipment*, SEMATECH, Austin, TX.

[3] T. Ziehe, *Software Process Improvement (SPI) Guidelines for Improving Software: Volume 5.0,* SEMATECH Technology Transfer #96103188A-TR, October 31, 1996.

[4] The profile of “a path to software excellence” presented in Section 4.2 is an adaptation (for the semiconductor industry) of material from an article with the same title written by Capers Jones and published in *Knowledge Base*, the newsletter of the Software Productivity Research Council, March 1994.

[5] M. Paulk, et al., *Capability Maturity Model for Software, Version 1.1*, Software Engineering Institute, Pittsburgh, PA, CMU/SEI-93-TR-024, February 1993.

M. Paulk, et al., *Key Practices of the Capability Maturity Model, Version 1.1*, Software Engineering Institute, Pittsburgh, PA, CMU/SEI-93-TR-025, February 1993.

K. Dymond, *A Guide to the CMM*, Process Inc. US, Annapolis, MD, 1995

[6] ISO 9000-3, *Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software, International Organization for Standardization*, June 1991.

M. Paulk, *A Comparison of ISO 9001 and the Capability Maturity Model for Software*, Software Engineering Institute, Pittsburgh, PA, CMU/SEI-94-TR-12, August 1994.

[7] *Standardized Supplier Quality Assessment Workbook*, SEMATECH internal document, January 15, 1996.

Standardized Supplier Quality Assessment Training Participant Guide, Technology Transfer #96053116A-TRG, SEMATECH, Austin, TX, 1996.

[8] W. Edwards Demming, *Out of the Crisis*, MIT Center for Advanced Engineering Study, Cambridge, MA, 1986.

[9] J. Gremba and C. Myers, *The IDEAL Model: A Practical Guide for Improvement*, Published in Bridge, a publication of the Software Engineering Institute, Pittsburgh, PA, Issue Three, 1997.

[10] J. Ferguson, et al., *Software Acquisition Process Maturity Questionnaire*, Software Engineering Institute, Pittsburgh, PA, Special Report, CMU/SEI-97-SR-013, July 1997.

Software Acquisition Capability Maturity Model, A wall chart published as part of Bridge, a publication of the Software Engineering Institute, Pittsburgh, PA, Issue Two, 1997

APPENDIX A
The Policy: Improve Software Quality

In the first quarter of 1997, SEMATECH member companies, through the SPI Project, distributed a statement of policy about software quality to all members of SEMI/SEMATECH. The policy delivers the following key messages to suppliers:

- Software reliability is extremely important to us, a group of your customers.
- We expect you to take the steps (indicated in the policy) needed to produce high-quality software.
- We expect you to produce indicators-of-software-quality (identified in the policy) and use them to drive continuous improvement of your software.
- We will review with you from time to time these quality indicators.

SEMATECH member companies, individually, are using the software quality policy and indicators in the following ways:

- As a basis for evaluating and qualifying suppliers prior to supplier selection
- During the procurement process as a basis for product selection
- At product delivery and acceptance time to insure software quality is improving
- Throughout ongoing product support and evolution to accelerate improvement progress

Collectively, member company representatives on the Equipment Productivity Improvement Teams are using the policy and its quality indicators in scheduled reviews of supplier software performance.

Following is the complete text of the policy with an explanation of the four key quality indicators that it names.

SEMATECH MEMBER COMPANIES SOFTWARE QUALITY IMPROVEMENT POLICY

December 1996

Software's role is critical to productivity and profitability in Semiconductor Wafer Fabrication, Assembly, and Test operations. For many suppliers, software is now also a significant factor in their competitive position and overall business success. Recognizing this, the SEMATECH Member Companies expect their equipment and system Suppliers to continuously improve reliability and the overall quality of software by the following:

1. Establish an organizational commitment to improving software quality, driven by objective Process Assessment Findings from SSQA ratings and/or SEI maturity levels.
2. Institutionalize continuous improvement by adopting Software Quality Goals and Improvement Action Plans that achieve the goals, as evidenced by recognized metrics.
3. Establish and maintain a Problem and Corrective Action Reporting System that drives continuous improvement in equipment and system reliability.
4. Establish and maintain a Software Revision Control System to assure reproducibility of all delivered software and proper application of corrections and upgrades.
5. Charter staff and allocate budget to the work of continuously improving software quality, objectively validating the quality of software produced, and verifying the software process used.
6. Provide the following data as part of each major product release:
   - Software Test Plan and Test Report.
   - Release Notes that include documentation for all changes and a list of all known problems in the delivered software, with a recommended procedure for recovery when the problem is encountered.
   - Compliance status with the GEM/SEM and SECS Interface Standards.
7. Show compliance with this policy by using the following metrics:
   - Software quality (defect trend)
   - Software reliability (MWBI/MTBF)
   - On-time delivery (schedule adherence)
   - Software process maturity (SSQA Module 3 or SEI Levels)

   Detailed specifications for these metrics are available from the SPI Project at SEMATECH, or from any Member Company representative on the SPI Project's PTAB (Project Technical Advisory Board).

A Supplier's Assessment Findings (Item 1), Quality Goals and Action Plan (Item 2), and data for all four metrics (Item 7) are to be provided upon request by any SEMATECH Member Company or by SEMATECH as part of a SEMATECH program.

Software Quality Key Indicators (4-Ups) for Semiconductor Equipment Suppliers

Purpose:

SEMATECH's Software Quality Improvement Policy includes a set of key indicators that semiconductor equipment suppliers will be required to provide to their customers (SEMATECH member companies) on request. These indicators are of significant value to suppliers in improving their software quality. The purpose of this document is to define the key indicators, methods for collecting the data, formats for presenting the indicators to customers, and the maximum acceptable "age" of the data presented.

Definitions:

Software Product Indicators (software defect trend, software reliability) measure the software in the supplier's product. The key indicators should present the status of the current release of the product in question. Historical data should be kept for several recent software releases, to measure improvement. If the supplier manufactures more than one product, these indicators should be presented separately for each product (or product family, in the case of a group of very similar products).

Software Defect Trend (and defect density)—A measure of the number of *critical* and *serious* post-release defects contained in the supplier's software. Key metrics are defects discovered and defects closed (fixed) during a specific time period, and defects remaining open at the end of the time period. The key indicator includes these three data items, measured monthly and plotted as a trend chart. An additional measure is defect density, which is the number of defects divided by the code size (KSLOC = thousands of lines of non-comment source code), which "normalizes" the data as the supplier's software code base grows.

Software Reliability—A measure of the ability of the software to perform to its specifications without failing for a certain number of wafers (MWBI = Mean Time Between Interrupts) or period of time (MTBF = Mean Time Between Failures). Software reliability is calculated from test results, by logging failures, and then using standard reliability methods to compute MWBI or MTBF. The key indicator is to be calculated monthly, and plotted as a trend chart.

Software Process Indicators (on-time delivery/requirements volatility and software process maturity) measure the software organization and how it works, not the software contained in the supplier's product. If the supplier has more than one software product development organization (e.g., divisions, groups), each will be required to present these indicators separately.

On-time delivery (and requirements volatility)—A measure of the time lag between the committed delivery date and the actual delivery date for a software release. The key indicator is the lag time (days late) for several recent releases, plotted as a trend chart. An additional measure is *requirements volatility*, one of the main root causes of late delivery. Key metrics are the number of new features and defect fixes actually included in the software release vs. the planned and committed content, plotted as a trend chart for several recent releases.

Software Process Maturity—A measure of the ability of the software organization to develop quality software, on time, on budget, as defined by the Software Engineering Institute Capability Maturity Model for Software (SEI/CMM) or the SEMATECH Standardized Supplier Quality Assessment (SSQA) Module 3–Software Quality. The key indicator for the SEI/CMM method is the organization's satisfaction of the SEI Level 2 and 3 Key Process Areas, as determined by an independent assessment. The key indicators for SSQA Module 3 are the scores for the twelve SSQA requirements, determined by a self-assessment and validated by a SEMATECH/member company assessment team.

Maximum "age" of data—Key indicators shall be tracked monthly, except software process maturity, which shall be based on assessment data that is not more than one year old.

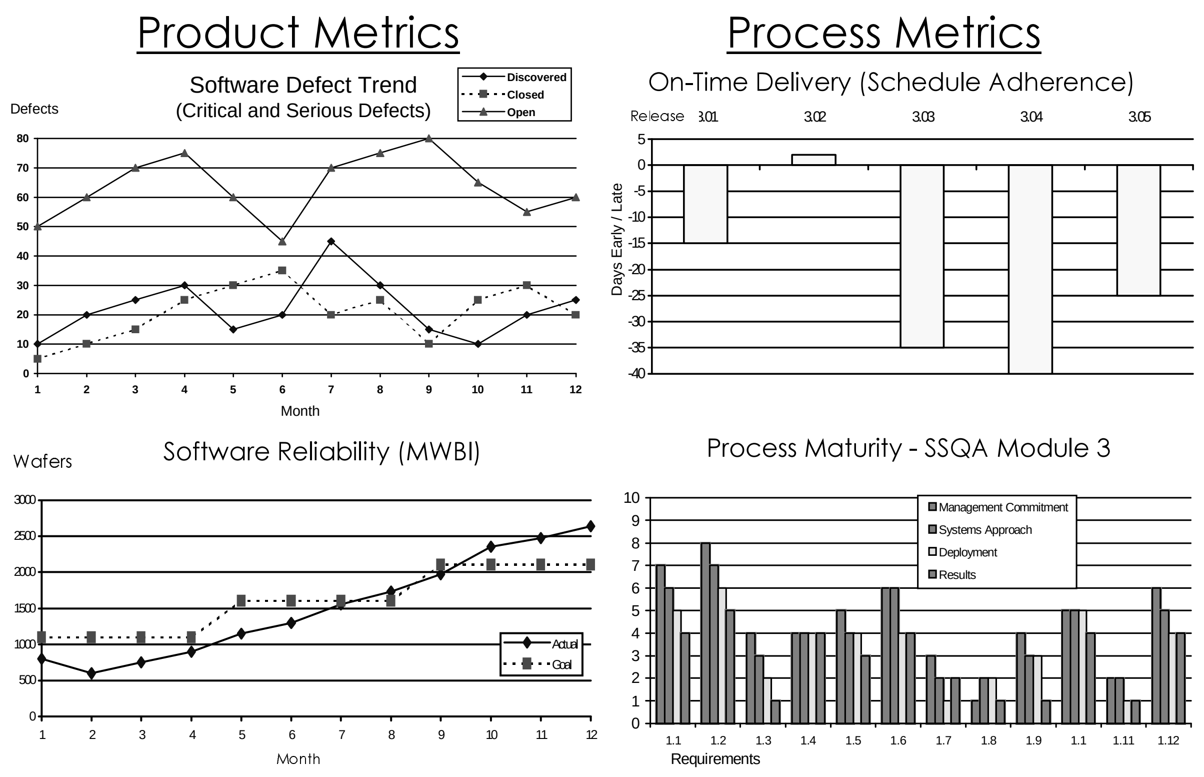


Figure 9 Software Quality Indicators (4-UPs)

Software Defect Trend

The software organization is required to have a software defect tracking and closure system (whether manual or automated), and a system of classifying defects by severity level (e.g., critical, serious, routine, minor, cosmetic). The following key indicators are to be based on *critical and serious defects only:*

Software Defects Discovered

Source: field engineering reports, software problem reports, customer trouble calls, test results.

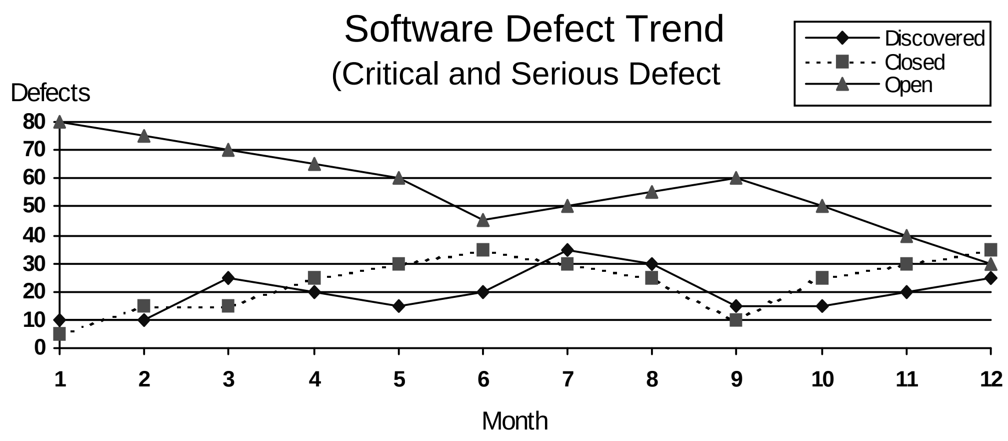
Software Defects Closed (fixed)

Source: completion of module and regression testing by supplier software engineering organization.

Software Defects Open (discovered but not yet fixed)

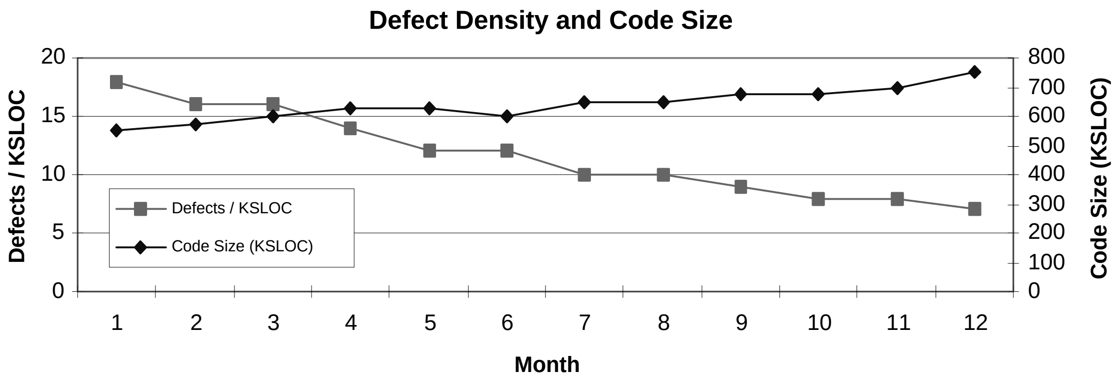
Formula: open defects at start of period + newly discovered defects–defects closed during the time period.

The defect trend data is to be measured monthly, and presented as a trend chart:



Software Defect Density

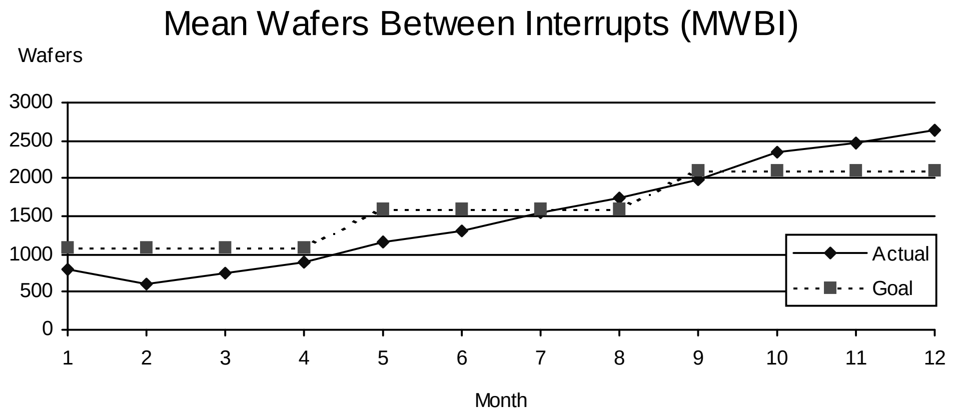
Formula: the number of open defects divided by the size of the software code (KSLOC). Both the defect density and code size should be plotted as a trend chart.



Suppliers are required to establish a software reliability goal, track their performance against that goal, and present both goal and actual on a trend chart. The test environment should closely approximate a “production environment.” Data may be collected at either the supplier’s site or at customer sites. This is an ideal opportunity for “partnering” between suppliers and their customers to measure reliability.

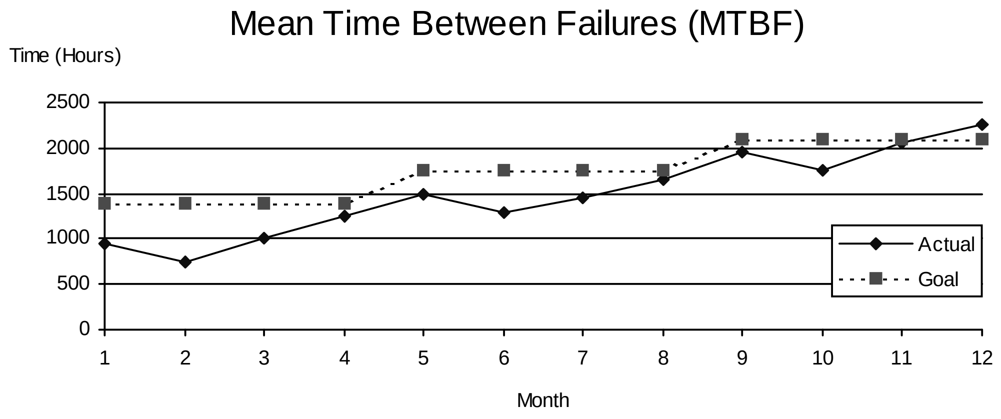
Mean Wafers Between Interrupts (MWBI)—the number of wafers processed between software failures.

MWBI is the preferred software reliability key indicator for wafer processing tools, because it measures software reliability while the supplier’s equipment is actually processing wafers (not while the equipment is in an idle state or down for maintenance or other reason). To calculate MWBI, suppliers must log the number of wafers processed between software failures and compute the average monthly.



Mean Time Between Failures (MTBF)—the duration between failures (failure interval).

Software MTBF is the preferred software reliability indicator for tools that do not process wafers and tools that process wafers in batches (where MWBI would be confusing), and is acceptable for any tool. In order to compute software MTBF, suppliers must log the time of each software failure during a test run, and compute MTBF monthly, using a standard reliability methodology.

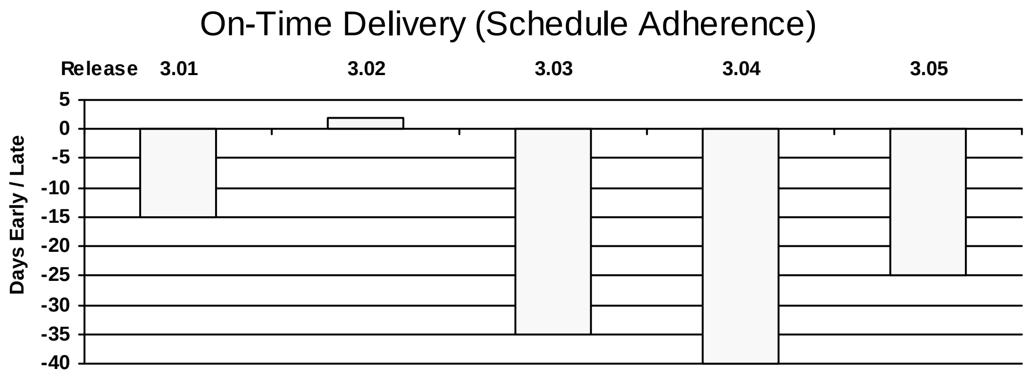


On-Time Delivery (and Requirements Volatility)

The software organization is required to have an effective project planning and tracking system capable of recording plan-vs.-actual performance against the committed schedule and the planned and actual contents (new features and defect fixes) of the release. The following key indicators are to be measured:

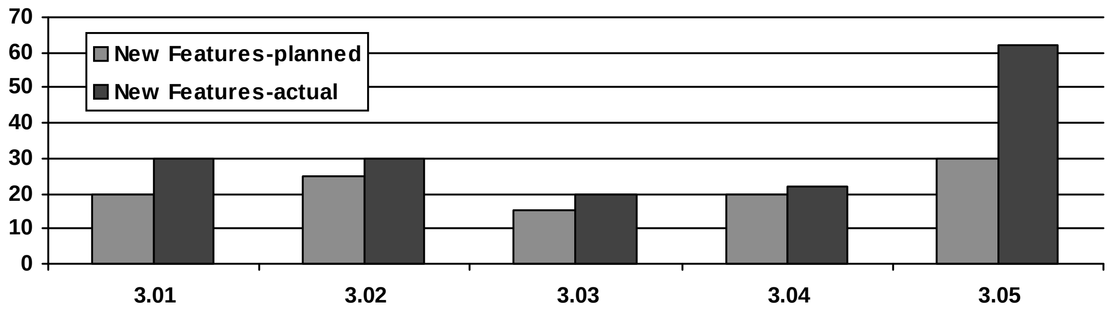
On-Time Delivery

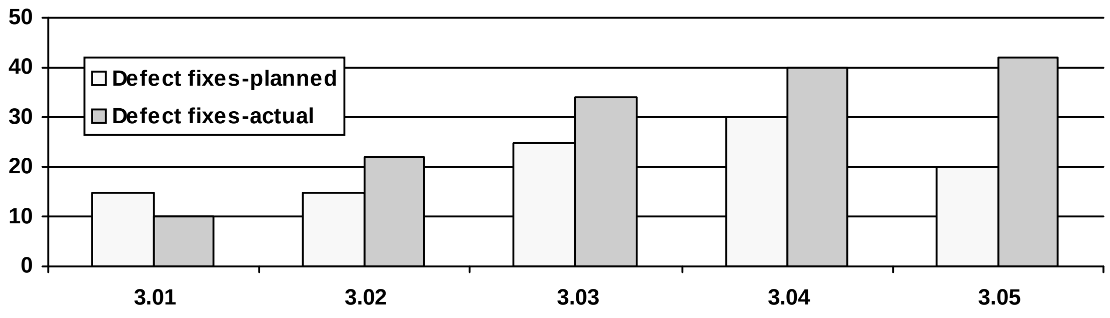
Number of days early (+) or late (-) for several recent software releases, measured at the ECO release date.



Requirements Volatility

Release contents: new features and defect fixes, planned-vs.-actual, measured at the ECO release date:





Software Process Maturity (SEI/CMM Levels 2-3)

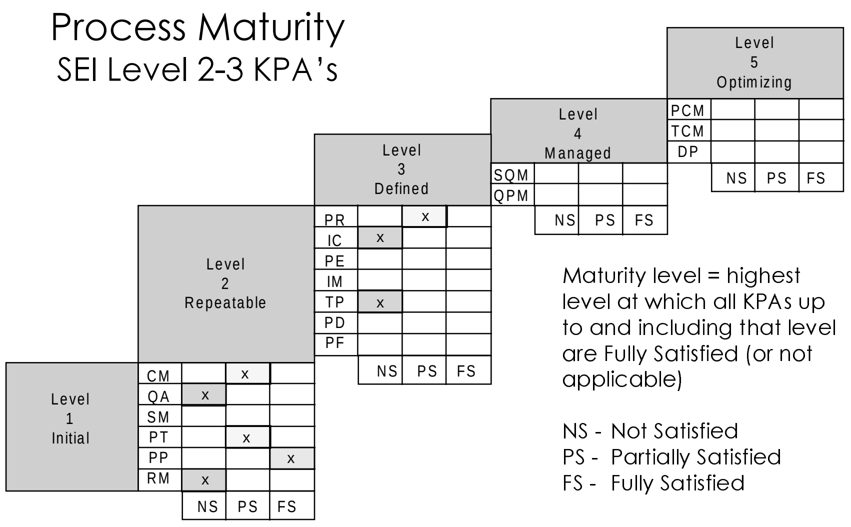
The Software Engineering Institute (SEI) Capability Maturity Model for Software (CMM) defines five Levels of software process maturity. Each level has several key process areas (KPAs) that must be satisfied to reach that level. Each KPA has a set of goals and activities associated with it which must be satisfied to reach that level. To reach SEI Level 2, all of the Level 2 KPAs must be fully satisfied.

SEI Level 2 KPAs

- Requirements Management
- Project Planning
- Project Tracking And Oversight
- Subcontract Management
- Software Quality Assurance
- Configuration Management

SEI Level 3 KPAs

- Organization Process Focus
- Organization Process Definition
- Training Program
- Integrated Software Management
- Software Product Engineering
- Intergroup Coordination
- Peer Technical Reviews



The software organization is required to perform an assessment of its software engineering practices (the assessment to be performed by a qualified SEI assessor) and document the results using the SEI/CMM method at least once a year. The results must be presented as an achievement of SEI Levels 2 or 3, and/or the satisfaction of the SEI Level 2 and 3 KPAs as shown above.

1 Capability Maturity Model for Software, Version 1.1 (see References, Section A.1.)

Software Process Maturity (SEMATECH SSQA Module 3)

SEMATECH Standardized Supplier Quality Assessment (SSQA) Module 3 defines 12 requirements:

1.1	Documented process	1.7	Independent testing of software
1.2	Software project planning and control	1.8	Software quality goals
1.3	Software development system	1.9	Software quality organization
1.4	Documented requirements	1.10	Continuous software quality improvement
1.5	Configuration management and change control	1.11	Software capability improvement
1.6	Software development security	1.12	Software subcontractors

Each of these has an overall statement of the requirement, followed by several detailed items, for example:

1.1 Documented Process: An approved, documented process shall be used to guide the development and maintenance of all software that impacts total customer satisfaction.

Indicate below what you are doing to satisfy this requirement:

a. A documented process for software development has been approved by senior management.

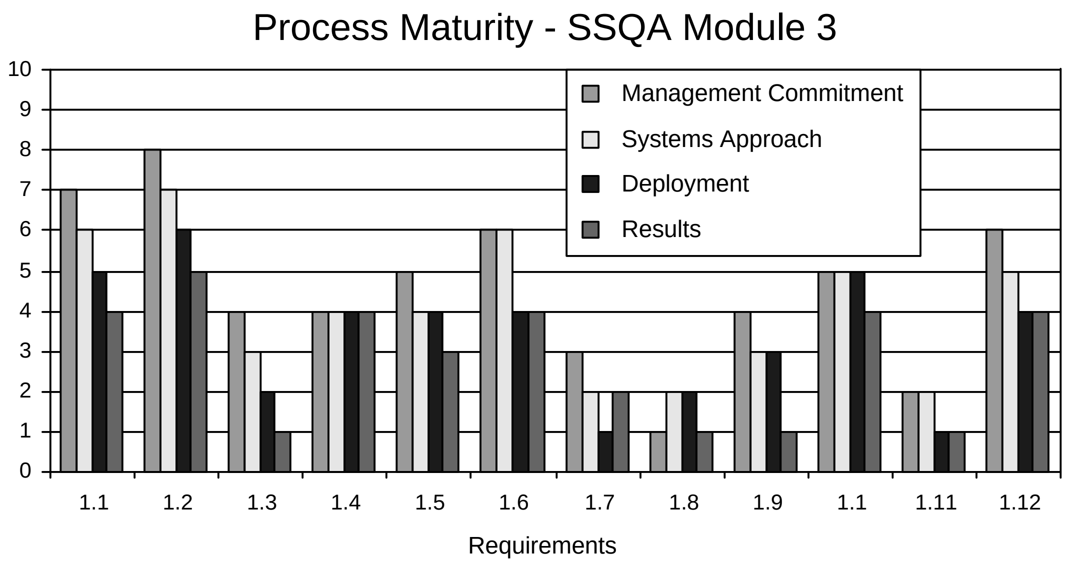
b. The process is followed with measurable and observable milestones established.

c. Training is in place to educate people on the process.

-etc.-

Each requirement is scored on a scale of 0-10 on each of four factors, using the SSQA scoring matrix:

Management Commitment	Is management committed to this requirement being satisfied?
Systems Approach	Is there a systematic approach in place that defines the process?
Deployment	Is the process deployed and being used throughout the organization?
Results	Are there measurable results that can be attributed to the process?



The software organization is required to perform a self-assessment of its practices (validated by a SEMATECH/member company assessment team) and document the results using the SSQA method at least once a year. The results must be presented as the scores for each of the twelve SSQA requirements, as shown above.

2 The SEMATECH SSQA Workbook and Scoring Matrix are available to SEMATECH and SEMI/SEMATECH member companies. Call the SSQA Program Manager at SEMATECH for ordering information.

A.1 References

Software Reliability

Software Reliability, J.D. Musa, A. Iannino & K. Okumoto, McGraw-Hill, 1990.

Ensuring Software Reliability, A.M. Neufelder, Marcel Dekker, 1993.

Tactical Software Reliability Guidebook, Technology Transfer #95092967A-GEN, 1995.

IRONMAN Methodology Manual, Technology Transfer #94082509A-XFR, 1994.

Standard for Definition and Measurement of Equipment Reliability, Availability, and Maintainability (RAM), SEMI E10-96.

Software Defect Trend

Software Quality Measurement: A Framework for Counting Problems and Defects, CMU/SEI-92-TR-22, 1992.

Failure Reporting, Analysis, and Corrective Action System (FRACAS), Technology Transfer #94042332A-GEN.

IEEE Standard Classification for Software Anomalies, IEEE Std 1044-1993.

Software Process Maturity- SSQA/QSR

SSQA Workbook (Standardized Supplier Quality Assessment), SEMATECH.

Quality System Review (QSR) Guidelines, Subsystem 10 – Software Quality Assurance, Motorola, Inc.

Software Process Maturity- Software Engineering Institute Capability Maturity Model for Software

Capability Maturity Model for Software, Version 1.1, CMU/SEI-93-TR-24, 1993.

Key Practices of the Capability Maturity Model, CMU/SEI-93-TR-025, 1993.

Maturity Questionnaire, CMU/SEI-94-SR-7, 1994.

On-Time Delivery (Estimating, Project Planning, and Tracking, etc.)

Software Effort and Schedule Measurement: A Framework for Counting Staff-hours and Reporting Schedule Information, CMU/SEI-92-TR-21, 1992.

Other Related Software Quality Topics

Managing the Software Process, W.S. Humphrey, Addison-Wesley, 1990.

Software Measurement for Semiconductor Manufacturing Equipment, Technology Transfer #95012684A-TR, 1995.

Software Measurement for DOD Systems: Recommendations for Initial Core Measures, CMU/SEI-92-TR-19, 1992.

APPENDIX B
The Customer: Actions Consistent With The Policy

A semiconductor manufacturer buys and uses massive amounts of software from many suppliers. Some elements are acquired as standalone systems (e.g., the manufacturing execution system), but most are embedded in the many tools installed throughout the factory.

A manufacturer’s executives, managers, and staff make decisions and perform roles that impact the quality of software they acquire. Many of these people are not software specialists. This material is for them. It offers 24 guides-to-action for staff in non-software roles. The guidelines point to actions that ensure high quality for factory software.

These guidelines were developed by representatives from several of the SEMATECH member companies, supplier companies, staff from the Software Process Improvement Project, and other members of the Project Technical Advisory Board that serves as an SPI steering committee. They are consistent with the Capability Maturity Model for Software Acquisition, developed by the Software Engineering Institute [10].

The guidelines are for people in three main categories:

1. Executives and senior managers
2. People responsible for developing and operating a semiconductor factory
3. People with significant responsibility for interacting with internal or external suppliers

Within the three categories, the 24 guidelines are grouped according to eight areas of activity. Twelve of the 24 are designated the core group. Section 3.2 lists this core group; in this Appendix they are italicized. It is recommended that an organization apply these guidelines first. The other 12 guidelines, also important, supplement and support the core group.

You apply the 24 guides-to-action and institutionalize the practices you define by doing the following:

- Map the guidelines to your organization’s departments, functions, and/or business processes.
- Identify specific roles/individuals with responsibilities related to each guideline.
- Translate the guideline into specific practices appropriate for the roles/individuals identified.
- Ensure that each practice you define supports the organization’s business plan, and is consistent with the Policy for Software Quality Improvement and with the organization’s corporate culture/approach to doing business.
- Institutionalize the practices defined: make them mandatory and train people to use them.
- Assure that the practices are regularly and effectively performed.
- From time to time assess your software practices to evaluate their effectiveness; add and adjust practices as necessary.

To continuously improve, set quality goals for factory software and metrics for measuring the level of quality you achieve. The two product metrics specified in the SQI Policy—reliability and defect trends for delivered software—are an excellent place to start. Applying these guidelines in the context of such goals and metrics provides a base on which to develop the well-rounded competence in software that a semiconductor manufacturer needs today to be competitive.

24 Guides-To-Action that Improve Software Quality for Manufacturers in the Semiconductor Manufacturing Industry

- Executive Management:

Area 1 Sponsoring Software Quality Improvement

1.1 Deploy organization-wide a policy on software quality improvement for suppliers.
1.2 Ensure the organization's own practices are consistent with the policy for suppliers.
1.3 Allocate adequate resources to the work of software quality improvement.

- Factory Development and Operation:

Area 2 Factory Planning, Engineering, Specifying

2.1 Develop requirements/specifications for software, including quality specifications.
2.2 Document and manage the requirements/specifications for software.
2.3 Define software quality and put priorities on the attributes selected to define it.

Area 3 Procuring Factory Equipment and Systems

3.1 Use your policy on software quality improvement in the procurement process.
3.2 Base all procurements that include software on stated requirements/specifications.
3.3 Continuously improve how you procure products that include software.

Area 4 Integrating Equipment and Systems, Testing, Qualifying

4.1 Share with suppliers data about software's performance during integration.
4.2 Coordinate with suppliers their responsibilities for software testing prior to delivery.
4.3 Put all software that has been accepted for use in a factory under version control.
4.4 Define software-related roles/responsibilities/processes for suppliers during integration.

Area 5 Operating the Production-Factory

5.1 Collect software performance data for production operations; share it with suppliers.
5.2 Deploy the Overall Equipment Effectiveness metric.
5.3 Relate software quality improvement issues to existing factory priorities and goals.

- Supplier Relations:

Area 6 Providing Leadership by Requiring Software Improvement

6.1 Require software quality improvement by reviewing quality indicators (the 4-Ups).
6.2 Show suppliers the results you have achieved in your internal improvement effort.

Area 7 Developing and Supporting Software and Systems (Internal Groups)

7.1 Apply your software quality improvement policy to all internal software groups.
7.2 Educate internal software groups/managers about the importance of and need for quality.
7.3 Use with internal suppliers the software practices used with external suppliers.

Area 8 Accelerating External Suppliers' Progress in Software Improvement

8.1 Do software quality improvement projects with key suppliers to accelerate progress.
8.2 Share lessons learned from improvement projects you have completed with suppliers.
8.3 Establish/strengthen management-to-management relations with all suppliers.

24 Guides-To-Action for SC Manufacturers with Notations

<><><><>

Organized by Area within Work Group

- **Executive Management:**

Area 1 Sponsoring Software Quality Improvement

1.1 Deploy organization-wide a policy on software quality improvement for suppliers.

Establish official company policy on quality in software acquired from suppliers, and deploy throughout your organization the authorities/responsibilities required to implement and enforce it. Use the Policy on Improving Software Quality (Appendix A) that has already been distributed to the members of SEMI/SEMATECH, or your own adaptation of it. Give priority to implementing your policy within the procurement/product-acceptance functions.

1.2 Ensure the organization's own practices are consistent with the policy for suppliers.

Use this document and its guidelines to define/establish software practices throughout your organization that collectively form a mature, well-rounded software capability to complement the mature approach to software you require of suppliers. A mature software capability also equips you to effectively expedite the progress suppliers make improving software quality.

1.3 Allocate adequate resources to the work of software quality improvement.

Allocate the staff and budget resources needed to move forward with the work of improving software quality, including the following:

- Charter and assign the responsibilities of an 'advocate' for factory software quality along the lines recommended in this document.
- Authorize the staff time required to apply the guides-to-action this document presents and to perform the software practices developed from the guidelines.
- Provide the staff time and other resources needed to produce the data needed for baselines, benchmarks, and progress reports – again along the lines recommended in this report.
- Take an active role in the work of the SPI Project's PTAB.

- **Factory Development and Operation:**

Area 2 Factory Planning, Engineering, Specifying

2.1 Develop requirements/specifications for software, including quality specifications.

Include software requirements, along with quality requirements, in the specifications for products that you deliver to suppliers.

- Establish and use well defined, effective lines of communication with suppliers for all aspects of software, including your requirements, delivery expectations, and acceptance/support criteria.
- Through these lines of communication deliver software requirements early and completely, and be sure they are correctly understood.
- Host visits by supplier software engineers to enrich their understanding of your needs and mode of operation.

2.2 Document and manage the requirements/specifications for software.

Document requirements completely, and manage changes with well defined configuration management disciplines. Also,...

- Prioritize requirements to show what is mandatory, and what is less important.
- Identify incompletions in requirements and flag what is most likely to change.
- Keep track of each requested special, avoiding requests for specials when possible, thereby keeping them to a minimum.
- Label each version of your requirements and use these labels when communicating requirements to suppliers; use a proven tool for configuration management.

2.3 Define software quality and put priorities on the attributes selected to define it.

Define software quality in terms of its key attributes, and then clearly state the priorities of these attributes. (The SQI Policy puts first priority on reliability; connectability is also important.) For your definition of software quality to have an impact, you must...

- Prioritize such attributes as reliability, connectability, usability, maintainability, etc.
- Have buy-in on the importance of software quality from senior managers, equipment/process engineers, and buyers.
- Ensure that statements about 'required quality' includes software.

Area 3 Procuring Factory Equipment and Systems

3.1 Use your policy on software quality improvement in the procurement process.

Implement in your product procurement process all aspects of your policy on software quality, including...

- Requiring suppliers to provide indicators of quality (e.g., the 4-Ups Metrics) that build your confidence in the software they deliver.
- Providing templates for the test plans, project plans, and other software quality indicators you think are important – templates that eliminate vagueness and ambiguity.
- Rating objectively the capability of each supplier to develop and support the software you acquire from them, including their software quality improvement effort, and evaluating objectively how their software meets the requirements.

3.2 Base all procurements that include software on stated requirements/ specifications.

Base all acquisitions of software, including embedded software, on documented requirements. Based on these requirements...

- Be clear about what testing you expect/require suppliers to do.
- Have an agent with factory-software experience involved in accepting/validating equipment with embedded software and new releases of software.
- Make sure a supplier knows how your payment schedule is affected by the product's satisfaction of your quality requirements, e.g., some percentage of the final payment is withheld until all software issues are closed.

3.3 Continuously improve how you procure products that include software.

Continuously improve how you procure software using...

- Best practices collected from throughout the industry and the lessons learned from evaluations of your own experience.
- SEI's Capability Maturity Model for Software Acquisition [10].

Area 4 Integrating Equipment and Systems, Testing, Qualifying

4.1 Share with suppliers data about software's performance during integration.

Share information with a supplier about all aspects of your work with a product during integration/test/qualification. From your work with their product, communicate performance data, lessons learned, and suggestions for software improvement.

4.2 Coordinate with suppliers their responsibilities for software testing prior to delivery.

During equipment and system integration/testing/qualification, coordinate with a supplier their responsibilities for software testing along these lines:

- Clearly define the test results you require for product acceptance.
- Provide suppliers with scenarios from the factory for their use in the environmental and final tests that they perform.
- Require suppliers to do extensive verification before delivering software, including (for key products) interactions with your shop floor-control-system.
- Provide the platform of testing a 'unique' feature or capability you have requested/required.
- Be responsible for testing/qualifying any specials and custom-features that you have requested/required.
- For all changes to software, review test results from the supplier and perform integration and regression tests as needed.

4.3 Put all software that has been accepted for use in a factory under version control.

Once software is accepted for use in a factory (including software embedded in a tool that has been accepted) put the software under version control that manages and tracks any and all changes to it.

4.4 Define suppliers' software-related roles/responsibilities/processes during integration.

To minimize the time and effort used, it is important to define the supplier's role(s), explain your work process, and schedule effort expected of a supplier in areas like assigning priorities to requirements, configuration management, managing specials, software release installation/qualification, problem reporting, and equipment performance.

Area 5 Operating the Production-Factory

5.1 Collect software performance data for production operations; share it with suppliers.

Assess the performance of supplier's product during production operations and communicate the findings to the supplier, together with any software improvement suggestions and the lessons learned during production.

5.2 Deploy the Overall Equipment Effectiveness metric.

Support the deployment of OEE throughout the factory, and relate software quality improvement to it by doing such things as...

- Requiring suppliers to comply with the E10-96 standard and E-58 ARAMS (Automated, Reliability, Availability, and Maintainability Standard).
- Improving and automating the techniques used to capture and report status and performance data.
- Improving the error detection, classification, and reporting practices used.
- Providing the training that prepares equipment operations to diagnose problems and use recovery options correctly.
- Reporting all equipment failures, even minor ones like "resets" and "reboots."

5.3 Relate software quality improvement issues to existing factory priorities and goals.

Recognize how software quality contributes to achieving factory priorities and goals and then be sure the software practices needed to make those contributions are in place and are effective.

- **Supplier Relations:**

Area 6 Providing Leadership by Requiring Software Improvement

6.1 Require software quality improvement by reviewing quality indicators (the 4-Ups).

Enforce the SQI Policy (or your own version of it) with each of your suppliers that deliver software by taking the following steps:

- Communicate the policy from your executive level to the supplier’s executive level.
- Require each of your suppliers to establish an organized and deliberate effort to improve software quality in a manner that is responsive to the Policy.
- Ensure that the entire policy is well understood, especially what evidences of software quality you expect the supplier to provide.
- Require that the supplier provide you quality indicators (4-Ups Metrics) at regular intervals, reviewing with the supplier how they use the measurements to drive an effort that is continuously improving quality.

6.2 Show suppliers the results you have achieved in your internal improvement effort.

Make available to external suppliers software quality indicators produced by ‘internal suppliers’ to show what your organization has accomplished, thereby indicating how serious you are about software quality improvement.

Area 7 Developing and Supporting Software and Systems (Internal Groups)

7.1 Apply your software quality improvement policy to all internal software groups.

Apply your policy on software quality improvement to all ‘internal suppliers’ by...

- Getting an explicit buy-in from the executive sponsor of software improvement.
- Educating the staff on the CMM for Software (from the SEI) [5] as a model of software process maturity (SSQA [7] and ISO [6]are alternatives), and on IDEAL [9] as a model for the continuous improvement cycle.
- Periodically assessing the maturity of your organization’s software capability, using the findings (together with the IDEAL model) as a basis for improvement plans and actions.
- Produce at regular intervals objective indicators of software quality, using them to drive and track the progress made toward the quality goals you have set.

7.2 Educate internal software groups/managers about the importance of/need for quality.

Advocates for software quality talk with all “internal suppliers” of software about the importance of improving software quality and what it takes to do it. For example, they...

- Recommend periodic assessments of their software capabilities.
- Encourage that they participate in the SPI Project’s workshops and other activities.
- Provide access to the SPI Guidelines, training courses, case studies, and process asset library.

7.3 Use with internal suppliers the software practices used with external suppliers.

Apply the disciplines used with the external suppliers of software to all of the internal groups who provide and support factory software.

Area 8 Accelerating External Suppliers’ Progress in Software Improvement

8.1 Do software quality improvement projects with key suppliers to accelerate progress.

Team up with key suppliers to do tactical projects that improve software quality in such areas of work as the following:

- Configuration management and release control
- Defect reporting and tracking
- Pre-release testing
- Testing the “specials” a manufacturer requests
- Quality and completeness indicators for tests of interface software, e.g., SECS/GEM[2]/SEM[3]/HSMS[4]
- The content and quality of release notes with major software releases
- The on-time delivery of products, with all of the functionality requested
- Monitoring equipment performance/reliability, especially in production

8.2 Share lessons learned from improvement projects you have completed with suppliers.

After completing tactical projects with suppliers, share the lessons learned about the approach used and the results achieved, doing it a form that is useful to other suppliers and manufacturers.

8.3 Establish/strengthen management-to-management relations with all suppliers.

Establish a strong relationship with each of your suppliers at the executive management level by doing the following:

- Tell supplier executives that software quality improvement is important and that measured progress is expected.
- Visit supplier sites and participate in reviews and assessments of their software quality improvement efforts and results.
- Provide incentives to compliant suppliers, recognizing and rewarding progress in software quality improvement.

[2] Generic Equipment Model
[3] Specific Equipment Model
[4] High Speed Message Services

APPENDIX C
The Suppliers: Progress Through Policy Compliance

Section 4 charts the path to software excellence [4]. The path is an elaboration of the Policy on Software Quality Improvement (see Appendix A). It consists of a starting point and six stages or categories of work that culminate in "industry leadership." Collectively, these seven elements show what software excellence requires. The outline below recaps the highlights.

- Starting Point. Your organization's software capability is clearly described by:
  - Assessments—the description is objective and recent
  - Baselines—it is stated in terms of a recognized metric or measurement standard
  - Benchmarks—it shows where you are relative to an appropriate standard of excellence
- Stage #1. Management is "on-board" an initiative to improve software quality:
  - Senior management sponsor—active participant
  - Influential champion and process manager—visionary leader and change agent
  - Project managers—plan, track, and control (risk management) projects
- Stage #2. An organization-wide software process is established and institutionalized:
  - Defined process
  - Staff training program (initial and continuing)
  - Continuous preservation and improvement infrastructure
- Stage #3. You change how you develop/support software effectively and with measured benefits:
  - Tool changes
  - Method changes
  - Process changes
- Stage #4. Your infrastructure of software specialties is mature and includes the following:
  - Change control (for both products and processes)
  - Configuration management (for all software)
  - Verification and validation (of all software products and processes)
- Stage #5. You are able to reuse software because you have done the following:
  - Trained your staff in software reuse
  - Increased software quality to a point that supports reuse
  - Established a reuse library that is operational and available
- Stage #6. It is evident that you are an industry leader, measured in terms of the following:
  - Quality of the products you deliver and customer satisfaction
  - Maturity and efficiency of the processes you use
  - Your staff's capabilities and the technologies they use

Objective indicators that show increasing product quality and progress toward software excellence are the lifeblood of any quality improvement initiative. Management that is truly committed to improving software quality recognizes this and supports production of such indicators to the same degree it supports the production of software itself. An organization uses such indicators first and foremost to drive its own improvement effort, and as a basis for incentives that it awards to the technical staff and managers. Such indicators also are used to show customers that software quality is improving.

In the semiconductor manufacturing industry, organizations use three different methods to measure their progress toward software excellence:

- *The Capability Maturity Model (CMM) for Software*, developed and supported by the Software Engineering Institute at Carnegie Mellon University [5].
- The *ISO 9000* series of standards, developed and supported by the International Organization for Standardization [6].
- The *Standardized Supplier Quality Assessment* method, developed and supported by the SEMATECH member companies [7].

The material in this appendix shows:

- How the *CMM for Software* indicates progress along the path to software excellence
- How the other two methods relate to the *CMM for Software*

Additional information about each of the three methods is available in the reference materials cited.

The Capability Maturity Model (CMM) for Software [5]. *The CMM for Software* measures progress toward software excellence in terms of maturity levels. Table 6 profiles the five maturity levels of the CMM. These levels correspond with the Path to Software Excellence in the manner indicated below:

Path to Software Excellence	***CMM for Software***
Starting Point..	Maturity Level 1—Initial
Stage 1—Management Issues	
Stage 2—Structured Methods...................................	Maturity Level 2—Repeatable
Stage 3—New Tools and Approaches	
Stage 4—Infrastructure..	Maturity Levels 3 & 4—Defined & Managed
Stage 5—Reusable Software	
Stage 6—Industry Leadership..................................	Maturity Level 5—Optimizing

The CMM for Software consists of 18 areas of software related activity, called key process areas. Each key process area has a list of goals and representative practices for accomplishing the goals. Each key process area is assigned to a specific maturity level. Figure 10 lists the key process areas by maturity level. An organization is rated at Maturity Level 2 if it has mastered the key process areas on Level 2, and likewise for Levels 3, 4, and 5. An organization masters a key process area by accomplishing its goals. An organization is rated Level 1 until it masters all key process areas at Level 2. Following is a summary of how organizations function at each level of maturity.

Table 6 Profiles of the Five Maturity Levels of the CMM for Software

• **Maturity Level 1: Initial** This first level of the CMM is unique; it has no model of maturity—no areas, goals, or practices. That void means that a software process which fails to qualify at any of the other four levels—those that are unpredictable and poorly controlled, even ad hoc and chaotic—are referred to as Level 1 processes.
• **Maturity Level 2: Repeatable** At this level the model's focus is on basic management control of software operations. Goals and practices in six key process areas identify what is required to repeat previously mastered tasks or projects in a way that gets similar results. For that reason, this level is referred to as "repeatable." By effectively planning and managing each of its software projects, an organization controls this most basic level of its software process.
• **Maturity Level 3: Defined** At this level, the model focuses on an established software process that is used throughout the organization. Goals and practices in seven areas clarify and characterize the entire process, making it understood and regularly used organization-wide. Once established, the process is institutionalized: it is documented, staff and managers are trained in all aspects of it, and it is verified that projects are conducted in a manner suitably consistent with it.
• **Maturity Level 4: Managed** At this level, the model's focus is on mastering techniques for measuring the software process itself, and the quality of software produced by the process. Goals and practices in two key process areas add instrumentation to the organization-wide process that was established at Level 3, allowing that process to be quantitatively understood and controlled, and also facilitating fact-based decision-making and priority-setting by managers.
• **Maturity Level 5: Optimizing** At this final level, the model's focus is on full control of the software process, including the ongoing improvement of both software quality and process maturity. Goals and practices in three key process areas identify important aspects of defect prevention and continuous improvement. At this level, the organization's software process routinely operates well, freeing a larger share of the effort for continuous improvement and process adaptation.

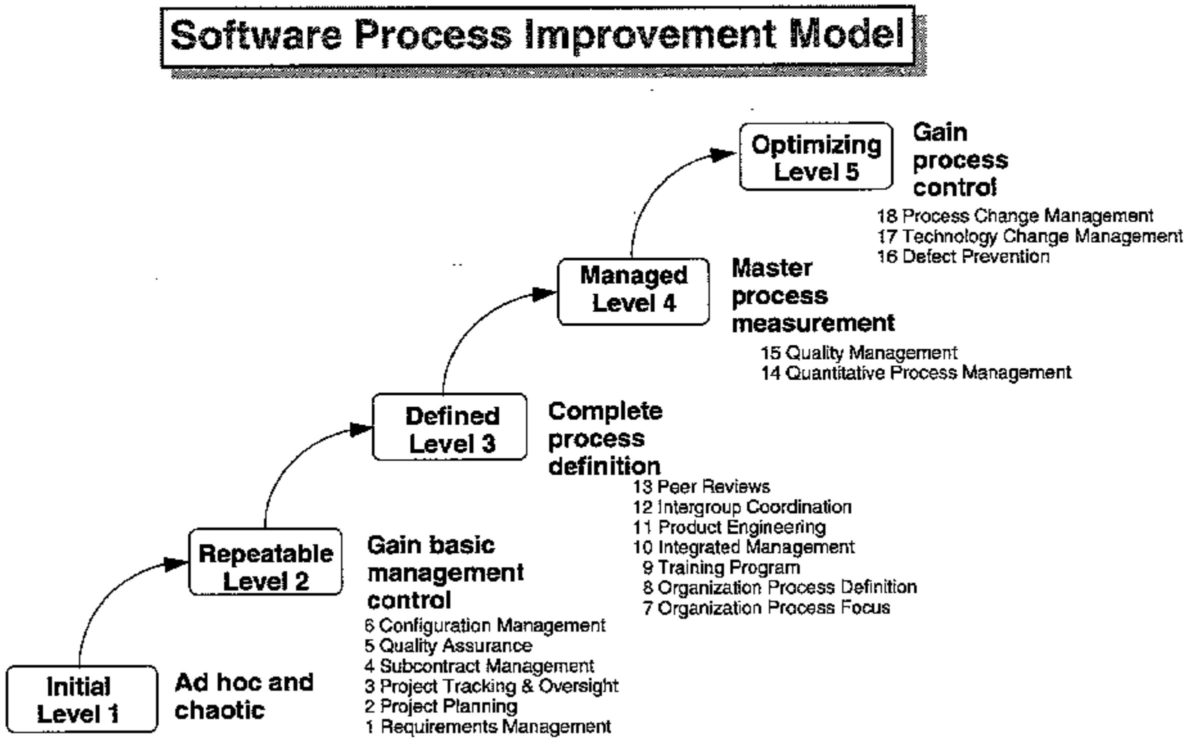

Software Process Improvement Model
Optimizing Level 5
Gain process control
18 Process Change Management
17 Technology Change Management
16 Defect Prevention
Managed Level 4
Master process measurement
15 Quality Management
14 Quantitative Process Management
Defined Level 3
Complete process definition
13 Peer Reviews
12 Intergroup Coordination
11 Product Engineering
10 Integrated Management
9 Training Program
8 Organization Process Definition
7 Organization Process Focus
Repeatable Level 2
Gain basic management control
6 Configuration Management
5 Quality Assurance
4 Subcontract Management
3 Project Tracking & Oversight
2 Project Planning
1 Requirements Management
Initial Level 1
Ad hoc and chaotic

Figure 10 Improvement Strategy and Key Process Areas of the CMM for Software

Level 2—Repeatable: Gaining Basic Management Control. When an organization has a Maturity Level 2 rating, its software process is repeatable and under basic management control. This means that project managers are able consistently to make reasonable estimates and plans, and track and control project performance via these plans. The organization's best software practices are accumulating at the project level, and there is an easy-to-recognize improvement in the style of work. Successful projects—in terms of cost, schedule, and requirements—are the norm. The disciplines in place track cost, schedule, and functionality, and allow new projects to draw on earlier, similar projects in ways that repeat their success.

Maturity at this level and beyond is implemented within norms that the organization establishes for software projects, norms that define the business settings for software projects and how these projects relate to other aspects of product production and support.

Representative accomplishments at this level of maturity include the following:

- Policies for managing projects and procedures that implement the policies are established.
- Planning and managing new projects is based on experience with similar projects.
- Successful practices developed on earlier projects are reused on subsequent projects.
- Project commitments are realistic, based on results from previous projects and requirements of the current project
- Software project managers track software costs, schedules, and functionality.
- Problems in meeting commitments are identified when they arise.

- Software requirements and the work products developed to satisfy them are baselined, and their integrity is controlled.
- Software project standards are defined, and the organization ensures they are faithfully followed.
- The software project establishes strong customer-supplier relationships with subcontractors.
- The project process is institutionalized, meaning it is practiced consistently, documented, trained, and measured, and can improve.

For an organization to be rated at this level of maturity, it must be producing results that satisfy the 20 goals listed for the six key process areas in Table 7.

Table 7 Key Process Areas and Goals for Maturity Level 2

1. Requirements Management (KPA 1)	An established baseline of software requirements exists and is used.
	Software (all aspects of it) is kept consistent with the requirements.
2. Software Project Planning (KPA 2)	Software estimates are documented for use in planning and tracking.
	Software project activities and commitments are planned and documented.
	Groups and individuals agree to the commitments that impact them.
3. Software Project Tracking and Oversight (KPA 3)	Actual results and performances are tracked against software plans.
	Corrective actions are taken that deal with significant deviations from plans.
	Changes to commitments are agreed to by those affected by the change.
4. Software Subcontract Management (KPA 4)	The subcontractors that are selected are qualified.
	The contractor and subcontractor agree to their commitments.
	The contractor and subcontractor maintain ongoing communications.
	The contractor tracks actual results and performance against commitments.
5. Software Quality Assurance (KPA 5)	The SQA activities are planned.
	Adherence to standards, procedures, and requirements is objectively verified.
	Affected groups and individuals are informed of SQA activities and results.
	Senior management handles noncompliance issues not resolved in projects.
6. Software Configuration Management (KPA 6)	Software CM activities are planned.
	Selected work products are identified, controlled, available.
	Changes to identified work products are controlled.
	Affected groups and individuals are informed about software baselines.

Level 3—Defined: Establishing Your Process. At Maturity Level 3, the best practices from projects have been generalized for use by the organization as a whole. That means the organization has found a systematic way to share best practices, namely codifying them as the organization's standard software process. Both management and engineering activities that develop or maintain software are documented, generalized, and integrated into a standard process for the organization.

The organization's defined process does not restrict projects, because there is also a systematic way to adapt the accumulated process knowledge to the needs of each new software project that comes along. That means each software project follows an approved, tailored version of the standard process.

Representative accomplishments at this level of maturity include the following:

- There is an organization-wide understanding of the roles and responsibilities for all participants in the software process.
- An organization-wide training program is implemented for managers and technical staff, ensuring they have the knowledge and skills their roles require.
- Projects tailor the organization's standard software process to develop their own defined software process.
- The software process is defined in terms of readiness criteria, required inputs, standards and procedures for performing the work, verification mechanisms (such as peer reviews), expected outputs, and completion criteria.
- Management has good insight into technical progress on all projects.
- Within established product lines, cost, schedule, and functionality are controlled, and software quality is tracked.

For an organization to be rated at this level of maturity, it must be producing results which satisfy the 17 goals listed for the seven key process areas in Table 8.

Table 8 Key Process Areas and Goals for Maturity Level 3

7. Organization Process Focus (KPA 7)
Process development and improvement is coordinated across the organization.
Strengths/weaknesses of the software process are identified by a process standard.
Organization-level process development/improvement activities are planned.
8. Organization Process Definition (KPA 8)
A standard software process for the organization exists and is maintained.
Information about process use by projects is collected, reviewed, maintained.
9. Training Program (KPA 9)
Training activities are planned.
Training is provided for both management and technical staff.
Individuals involved with software receive the training they need in their roles.
10. Integrated Software Management (KPA 10)
Each project's defined process is tailored from the organization's process.
Each project is planned and managed according to the project's defined process.
11. Software Product Engineering (KPA 11)
Software engineering tasks are defined, integrated, and consistently performed.
Software work products are kept consistent with each other.
12. Intergroup Coordination (KPA 12)
The customer's requirements are agreed to by all affected groups.
Commitments between the engineering groups are agreed to by affected groups.
The engineering groups identify, track, resolve intergroup issues.
13. Peer Reviews (KPA 13)
Peer review activities are planned.
Defects in the software work products are identified and removed.

Level 4—Managed: Mastering Process Measurement. At Level 2, the best practices tend to be in projects. By Level 3, the organization has mastered the technique of spreading the best practices across the organization in the form of a standard process. Now, at Level 4, all the process assets accumulated from Level 2 and Level 3 practices are used by the Level 4 organization to support projects with a quantitatively understood, stable process. Detailed measures of the software process and product quality are collected and used in managing the projects.

Representative accomplishments at this level of maturity are as follows:

- The organization sets quantitative quality goals for both software products and processes.
- Productivity and quality are measured for important activities across all projects as part of an organization-wide measurement program.
- Both the software process and its products are quantitatively understood and controlled.
- An organization-wide database is used to collect and analyze data reflecting uses of the defined software process within projects.
- All software projects report prescribed measurements that are well defined and consistent, and provide a basis for evaluating the process used and products produced.
- Projects exhibit control over their products and process by narrowing their performance variations, causing the performance to fall within acceptable quantitative boundaries.
- The risks involved in moving up the leaning curve in a new type of software project are known and carefully managed.
- The organization is able to predict trends in process maturity and product quality.
- When the limits set on product quality are exceeded, corrective actions are taken.
- Software products are of predictably high quality.

For an organization to be rated at this level of maturity, it must be producing results that satisfy the six goals listed for the two key process areas in Table 9.

Table 9 Key Process Areas and Goals for Maturity Level 4

14. Quantitative Process Management (KPA 14) The quantitative process management activities are planned. The performance of the project's defined process is controlled quantitatively. The capability of the organization's standard process is known quantitatively.
15. Software Quality Management (KPA 15) The software quality management activities are planned. Measurable goals for product quality and their priorities are defined. Actual progress made to achieving the quality goals is quantified and managed.

Level 5—Optimizing: Gaining Full Process Control. By the time Level 5 is reached, an organization's process mechanisms operate routinely to deliver its software products. These mechanisms work so reliably that the process of producing software is virtually automatic. There is even a process for handling deviations in processes. Everyone knows what to do next and whose job it is to do it.

So, what do the people in a Level 5 company do? They produce software that is reliable, and they do it with high productivity. The process is so effective they can readily adapt it for strategic business reasons, e.g., to gain competitive advantage in quality, in productivity, and/or in timely delivery—or in combinations of these three. When a process is effective, efficient, and under control, people can focus on improving it and adapting it to new situations.

Representative accomplishments at this level of maturity are as follows:

- The organization has the means to identify weaknesses and strengthen the process proactively, with the goal of preventing the occurrence of defects.
- Continuous process improvement is enabled by quantitative feedback from the process and from piloting innovative ideas and technologies.
- Data on software process effectiveness is used to perform cost benefit analyses of new technologies and proposed changes to the organization's software process.
- Innovations that exploit the best software engineering practices are identified and transferred throughout the organization.
- Software project teams analyze defects to determine their root causes.
- A project's software process is evaluated to prevent reuse of any deficiencies that are recognized.
- Lessons learned during a project are disseminated to other projects.
- Improvement occurs both by incremental advancements in the existing process and by innovations using new technologies and methods.

For an organization to be rated at this level of maturity, it must be producing results that satisfy the nine goals listed for the three key process areas in Table 10.

Table 10 Key Process Areas and Goals for Maturity Level 5

16. Defect Prevention (KPA 16) Defect prevention activities are planned. Common causes of defects are sought out and identified. Common causes are also prioritized and systematically eliminated.
17. Technology Change Management (KPA 17) Incorporation of technology changes are planned. Effects of new technologies on quality/productivity are evaluated. Appropriate new technologies are put into use across the organization.
18. Process Change Management (KPA 18)

Continuous process improvement is planned.

Participation in the organization's SPI activities is organization-wide.

The organization's standard and defined software are improved continuously.

A company that continuously improves software quality has changed its organizational culture. Its software process is established organization-wide and is owned and managed by its senior management sponsor. The chief change agent inspires, drives, and coordinates the ongoing work of improvement and adaptation to change. Software engineers throughout the organization are empowered within the "continuous improvement cycle" to propose and evaluate change, deploying and institutionalizing those changes that have merit. Changes made to the software process are evaluated in terms of their impact on the quality indicators selected. Improvement goals are expressed in terms of these qualitative indicators, and managers and staff are both incentivised to achieve the goals that are set.

ISO 9000 for Software [6]. There are 20 clauses in ISO 9001 which can be compared to the key process areas, goals, and practices of the CMM for Software. The 20 clauses are:

1. Management Responsibility—requires that a quality policy be defined, documented, understood, implemented, and maintained; that responsibilities and authorities for all personnel specifying, achieving, and monitoring quality be defined; and that in-house verification resources be defined, trained, and funded. A designated manager ensures that the quality program is implemented and maintained.
2. Quality System—requires that a quality system, including procedures and instructions, be established as an integrated process throughout the entire life cycle of the product.
3. Contract Review—requires that contracts be reviewed to determine whether the requirements are adequately defined, agree with the bid, and can be implemented.
4. Design Control—requires that procedures to control and verify the design be established. This includes planning design activities, identifying inputs and outputs, verifying the design, and controlling design changes.
5. Document Control—requires that the distribution and modification of documents be controlled.
6. Purchasing—requires that purchased products conform to their specified requirements, including the assessment of potential subcontractors and verification of purchased products.
7. Purchaser-Supplied Product—requires that any purchaser-supplied material be verified and maintained.
8. Product Identification and Traceability—requires that the product be identified and traceable during all stages of production, delivery, and installation.
9. Process Control—requires that production processes be defined and planned, including carrying out production under controlled conditions, according to documented instructions; special processes that cannot be fully verified after the fact are continuously monitored and controlled.

10. Inspection and Testing—requires that incoming materials be inspected or verified before use and in-process inspection and testing be performed; final inspection and testing are performed prior to release of finished products.
11. Inspection, Measuring, and Test Equipment—requires that equipment used to demonstrate conformance be controlled, calibrated, and maintained; when test hardware or software is used, it is checked before use and rechecked at prescribed intervals.
12. Inspection and Test Status—requires that the status of inspections and tests be maintained for items as they progress through various processing steps.
13. Control of Nonconforming Product—requires that nonconforming product be controlled to prevent inadvertent use or installation.
14. Corrective Action—requires that the causes of nonconforming product be identified, potential causes of the nonconformity be eliminated by changes in procedures as appropriate.
15. Handling, Storage, Packaging, and Delivery—requires that procedures for handling, storage, packaging, delivery, installation, and acceptance be established and maintained.
16. Quality Records—requires that quality records be collected, maintained, and dispositioned.
17. Internal Quality Audits—requires that audits be planned and performed, the results are communicated to management, and any deficiencies found are corrected.
18. Training—requires that training needs be identified and that training be provided, since selected tasks may require qualified personnel; records of training are maintained.
19. Servicing—requires that servicing activities, including maintenance, be performed as specified.
20. Statistical Techniques—requires that, where appropriate, adequate statistical techniques are identified and used to verify that process capability and product characteristics are acceptable.

There is a strong correlation between ISO 9001 and the *CMM for Software*. However, some issues in ISO 9001 are not covered in the CMM, and some issues in the CMM are not addressed in ISO 9001. The major similarity is that the bottom line for both is “Say what you do and do what you say.” The fundamental premise of ISO 9001 is that every important process should be documented and every deliverable should have its quality checked through a quality control activity. ISO 9001 requires documentation that contains instructions or guidance on what should be done or how it should be done. The CMM shares this emphasis on processes that are defined and documented.

The biggest difference between these two methods is the emphasis of the CMM on continuous process improvement. The CMM also emphasizes the need to record information for later use in the process and for improvement of the process. ISO 9001 addresses the minimum criteria for an acceptable quality system. For example, the first two columns of Table 11 illustrates the CMM-based profile of an organization that is ISO 9001-compliant and has no quality practices beyond those directly called out in ISO 9001.

Table 11 Relationships of ISO 9001 and SSQA to the CMM for Software

CMM for Software	ISO 9001		SSQA
Key Process Areas	Not Satisfied	Fully Satisfied	Requirements
18. Process Change Management			
17. Technology Change Management			
16. Defect Prevention			
15. Software Quality Management	▨▨▨▨		8
14. Quantitative Process Management			
13. Peer Reviews	▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨		3
12. Intergroup Coordination	▨▨▨▨▨▨▨		3
11. Software Product Engineering	▨▨▨▨▨▨▨▨▨▨▨▨▨▨		
10. Integrated Software Management			
9. Training Program	▨▨▨▨▨▨▨▨▨▨▨▨		11
8. Organization Process Definition			1
7. Organization Process Focus			10
6. Software Configuration Management	▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨		5, 6
5. Software Quality Assurance	▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨		7, 9
4. Software Subcontract Management	▨▨▨▨▨▨▨		12
3. Software Project Tracking & Oversight	▨▨▨▨▨▨▨		2
2. Software Project Planning	▨▨▨▨▨▨▨▨▨▨▨		2
1. Requirements Management	▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨▨		4

Standardized Supplier Quality Assessment (SSQA) [7]. The SSQA is a standardized assessment for suppliers in the semiconductor manufacturing industry. The assessment consists of a self-assessment by the supplier followed by a customer validation of the findings. This process helps prepare an organization for ISO certification. It also encourages customer/supplier communications on a wide range of important matters.

The SSQA method of assessment contains three modules:

- The first addresses the quality of systems and is structured to answer ISO 9001 requirements.
- The second has a Malcolm Baldrige National Quality Award orientation, but also addresses business systems and organizational capabilities that are critical to long term success (but are nor included in either ISO or Baldrige).
- The third addresses software processes and capabilities, using the requirements of Motorola, Inc.'s Quality System Review.

The software module consists of the following 12 requirements:

1. Documented Process—An approved, documented process shall be used to guide the development and maintenance of all software that impacts total customer satisfaction.
2. Software Project Planning and Control—Such mechanisms shall be in place and followed.
3. Software Development System—Software shall be developed as part of a total system using a phased development approach, intermediate deliverables, and review and approval shall be based on criteria for entry and exit from each phase.

4. Documented Requirements—Software shall be developed in support of documented (formal, written, approved, available) requirements, with conformance to the requirements verified.
5. Configuration Management and Change Control—Software shall be maintained under documented plans for configuration management and change control, including installation and customer configurations.
6. Software Development Security—Software shall be developed using proper tools and shall be documented, with approved procedures for security and information recovery, including disaster protection.
7. Independent Testing of Software—Software shall undergo system/acceptance testing by individuals or organizations not directly involved in the design or implementation or the product being tested. Testing shall reflect customer usage.
8. Software Quality Goals—Goals shall be established for software quality that are supported by measurement systems, with provisions for tracking progress toward these goals and highlighting issues from a customer perspective.
9. Software Quality Organization—Such an organization shall act as a customer advocate in software matters by assuring conformance to customer requirements and specifications, and proper execution of the approved development process.
10. Continuous Software Quality Improvement—A mechanism shall be used to ensure continuous quality improvement of software and of the software development process.
11. Software Capability Improvement—A program that includes deployment and assessment of training shall be in place for all software organizations.
12. Software Subcontractors—The processes used by software contractors shall be controlled, and conformance to requirements of subcontracted software is verified.

With each requirement are examples of actions that satisfy the requirement. Guided by these examples, an organization is scored (on a scale of 0–10) for how well it satisfies the requirement. This scoring is done for each of four factors:

- Management Commitment—Is management committed to this requirement?
- Systems Approach—Is there a systematic approach in place that defines the process?
- Deployment—Is the process deployed and being used throughout the organization?
- Results—Are there measurable results that can be attributed to the process?

The first and third columns in Table 11 above show the pattern of relationships between the 12 SSQA requirements and the 18 key process areas of the CMM for Software.

SEMATECH Technology Transfer
2706 Montopolis Drive
Austin, TX 78741

http://www.sematech.org