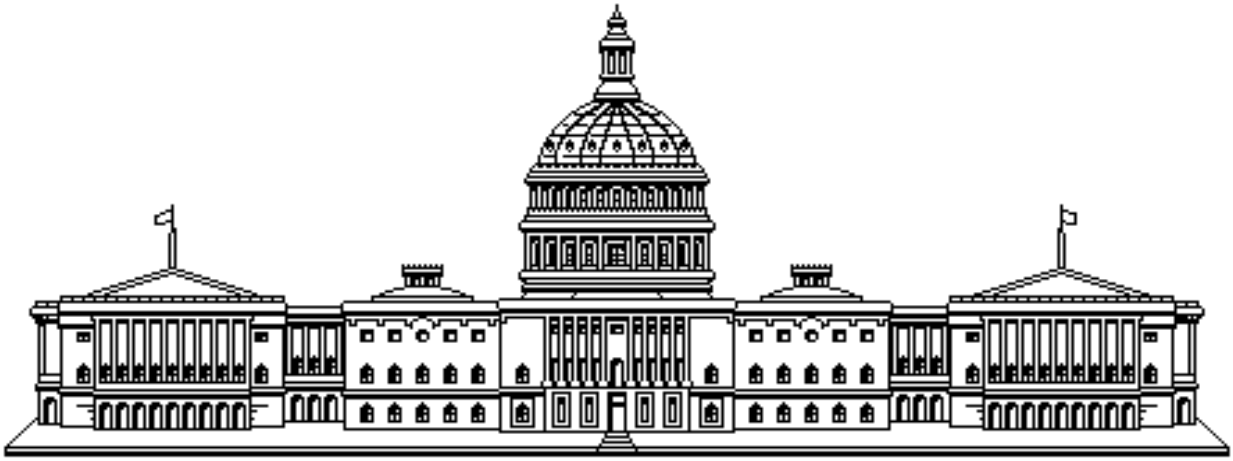


NATIONAL SOFTWARE QUALITY EXPERIMENT
A LESSON IN MEASUREMENT
1992-2002

Don O'Neill
Independent Consultant
(301) 990-0377
ONeillDon@aol.com

FEATURING

Experiment Motivation and Organization

Software Inspections Mechanism

Experiment Results

Conclusion

NATIONAL SOFTWARE QUALITY EXPERIMENT
A LESSON IN MEASUREMENT 1992-2001

KEY WORDS

Analysis Bins
Common problems
Core samples
Critical defect prediction
Defect types
Experimentation
Experiment participants
Organization type
Software process maturity level
Standard of excellence Product type
Product type
Programming language
Software Inspection Lab
Return on investment

FIGURES

9d Lines Per Conduct Hour by Product Type
9e Defects Per Session by Product Type
9f Preparation Per Conduct Effort by Product Type
9g Return on Investment by Product Type
9h Defect Type Distribution by Product Type

10 NSQE Metrics Statistics

11 NSQE Metrics Sorted
11a Preparation Per Defect Sorted
11b Preparation Per Major Defect Sorted
11c Major Defects Per Thousand Lines Sorted
11d Minor Defects Per Thousand Lines Sorted
11e Lines Per Conduct Hour Sorted
11f Defects Per Session Sorted
11g Preparation Per Conduct Effort Sorted
11h Lines Per Session Sorted
11i Return on Investment Sorted
11j Defect Type Distribution

12 Control Panels
13 Return on Investment Sorted
14 Defect Leakage Model
15 Major Defects Per Thousand Versus X10 Objective

NATIONAL SOFTWARE QUALITY EXPERIMENT
A LESSON IN MEASUREMENT 1992-2001

PROLOGUE

The nation's prosperity is dependent on software. Increasingly software is depended on to deliver value in industries of all kinds. The competition to meet time to market demands has resulted in the practice of shipping software products with defects, and customers have learned to accept this. The quality shortfall in the trustworthiness of the nation's software systems is now imposing limits on the security of the nation's software infrastructure. Consequently the nation's software industry is slipping, and it is slipping behind other countries. The National Software Quality Experiment is riveting attention on software product quality and revealing the patterns of neglect in the nation's software infrastructure.

ABSTRACT

In 1992 the DOD Software Technology Strategy set the objective to reduce software problem rates by a factor of ten by the year 2000. The National Software Quality Experiment is being conducted[1] to benchmark the state of software product quality and to measure progress towards the national objective.

The National Software Quality Experiment is a mechanism for obtaining core samples of software product quality. A micro-level national database of product quality is being populated by a continuous stream of samples from industry, government, and military services. This national database provides the means to benchmark and measure progress towards the national software quality objective and contains data from 1992 through 2002.

The centerpiece of the experiment is the Software Inspection Lab where data collection procedures, product checklists, and participant behaviors are packaged for operational project use. The uniform application of the experiment and the collection of consistent measurements are guaranteed through rigorous training of each participant. Thousands of participants from dozens of organizations are populating the experiment database with thousands of defects of all types along with pertinent information needed to pinpoint their root causes.

To fully understand the findings of the National Software Quality Experiment, the measurements taken in the lab and the derived metrics are organized along several dimensions including year, software process maturity level, organization type, product type, programming language, global region, and industry type. These dimensions provide a framework for populating an interesting set of analysis bins with appropriate core samples of software product quality and for deriving the statistical process control limits of each metric.

[1] The National Software Quality Experiment is an entrepreneurial activity.

EXPERIMENT MOTIVATION AND ORGANIZATION

Executive Summary

The National Software Quality Experiment (NSQE) is riveting attention on software product quality and revealing the patterns of neglect in the nation's software infrastructure. In 1992 the DOD Software Technology Strategy set the objective to reduce software problem rates by a factor of ten by the year 2000. The National Software Quality Experiment is being conducted to benchmark the state of software product quality and to measure progress towards the national objective.

A micro-level national database of product quality is being populated by a continuous stream of samples from industry, government, and military services. Over three thousand participants from over sixty organizations have inspected over a million lines of code and populated the experiment database with nearly fifteen thousand defects of all types along with pertinent information needed to pinpoint their root causes.

The results of the NSQE show no systematic movement towards fulfilling the national goal. The quality goal has not been met and was missed by a wide margin. Instead of a factor of ten reduction, problem rates are being pushed higher:

- With the emphasis on quicker, better, and cheaper.
- With the trend towards code and upload practice as the life cycle model.
- With the struggle to improve software process maturity and master the management track practices found in level 2 of the Software Capability Maturity Model, an obstacle to many [Paulk 95].
- With the downsizing of middle management and senior technical staff who often held the line on product quality through push back.

These NSQE metrics reveal patterns of neglect in the nation's software infrastructure. With 2.46 major defects per thousand lines of code and a major defect detected every 72.24 minutes of preparation effort, software organizations should treat these findings as a wakeup call. Nearly 41% of all defects are related to documentation particularly the lack of traceability from code to requirements. In fact, many software systems have no requirements baseline; the code simply does what the code does. Over 23% of all defects are related to compliance with programming standards and style.

Hidden in the NSQE metrics are clues and pointers to better practice. For example:

- The software inspection teams that look harder for defects are the ones that find more. Looking harder is accomplished by reducing the size of the artifacts inspected and increasing preparation effort.
- Program size matters. Defect density decreases with increasing size... to a point. Starting, finishing, and fitting in are all more error prone than the body of the program which gives it size.
- The data suggests that increased software process maturity results in increased defect detection, with the result perhaps being lower defect leakage into the field.
- In addition the data suggests that the organization's neglect of its software process exceeds the poor workmanship of individual programmers as the source of errors. Documentation and standards defect types account for nearly two-thirds of all defects, and these are the responsibility of the organization and its process.

W. Edwards Deming taught us that there is no substitute for profound knowledge. We know that software quality is one of the fault lines of the software crisis. It is an important leading indicator of global competitiveness. The National Software Quality Experiment

provides a framework for obtaining a profound knowledge of software quality on the project, within the organization, and across the nation. This knowledge will contribute to the understanding of the quality fault line.

Projects seek to achieve predictable performance. Organizations build on project predictability and seek to be competitive. The nation builds on organizational competitiveness in achieving prosperity. Today we find software as a key integrating element in every industry. If projects do not achieve predictable performance and if organizations are not competitive, the prosperity of the nation may be impacted.

Mr. Dooley[2] said, “It ain’t what you don’t know that hurts you. It’s what you know that ain’t so.” Many managers make decisions based on hearsay or myth. The National Software Quality Experiment is helping organizations to substitute facts for myths. For example, consider the question, “Which organizations detect more defects in software inspections, level 1, level 2, or level 3?” The answer is found in the measured results of the National Software Quality Experiment.

Lord Kelvin said it best, "When you can measure what you are speaking about and express it in numbers, you know something about it; when you can not measure it... your knowledge is meager and unsatisfactory." When it comes to software which must be bit-perfect, Lord Kelvin's hard edged conclusion is especially warranted.

Ralph Waldo Emerson wrote, “The years teach us things the days never knew.” The National Software Quality Experiment has operated for the past decade and is committed to run through the year 2000. True to Emerson's adage, interesting trends are beginning to reveal themselves. These trends may help us to better understand the practice of software engineering.

Robert McNamara said during the Vietnam War, "If you don't watch the periphery, it will soon become the center." The National Software Quality Experiment provides the means to keep an eye on the periphery... and the center of software product quality.

[2] Mr. Dooley was a turn of the century newspaper man in Chicago

OVERVIEW

The National Software Quality Experiment is doing more measurement at the national level than many organizations are doing at the project level! One observer said, *"All we do here is talk about it; you are actually doing it."*[3] Participants are attracted to the experiment as a place where they can calibrate their software quality against appropriately selected industry core samples. Here they can jump-start the organization's quality measurement program on the shoulders of uniform Software Inspection Lab procedures. These procedures are operationally packaged for project use and include well defined processes, industrial strength product checklists, participant roles and behaviors, and standard forms and reports.

The National Software Quality Experiment provides the framework to pose important quality questions. Its micro-level national quality database provides the measurements to answer them. Similarly, the extent of certain common risks can be quantified. As a participant in the experiment, an organization can characterize the effectiveness of its software quality process. At the industry level, progress towards the national software quality objective can be benchmarked.

Participants in the experiment benefit in several ways. They are able to characterize the maturity of their software quality process. With this understanding, they are able to establish goals for improving the process and to set priorities for immediate action. Beyond that, these organizations are able to promote a vision for excellence based on perfection not management slogans in their software products and to calibrate their progress towards the national software quality goal.

MOTIVATION

The motivation for the National Software Quality Experiment is found in my industry mission statement for strategic software improvement which includes the following:

1. To obtain the deepest possible understanding of software engineering in all its dimensions
2. To arrive at a realistic expectation for its application at every level
3. To communicate this knowledge and expectation to both producers and consumers

Software engineering has a multiplicity of dimensions. These include engineering, management, operations, product, process, business, and human resources to name several. The application of software engineering can be viewed at various levels including the nation, industry, the organization, the project, and the individual practitioner. The experiment and its micro-level national database is structured to service particular needs at all these levels and in all these dimensions.

The Department of Defense Software Technology Strategy was drafted for the Director of Defense Research and Engineering in December 1991 [DOD STS 91]. Three important national objectives were established to be achieved by the year 2000:

1. Reduce equivalent software life-cycle costs by a factor of two
2. Reduce software problem rates by a factor of ten
3. Achieve new levels of mission capability and interoperability via software

Every software organization should treat the national objective to improve software product quality by a factor of ten as a wake-up call. Are organizations planning to reduce software problem rates by a factor of ten? Do they know what these rates are now? Industry

[3] Comment made by participant in National Software Quality Experiment Seminar in 1993

problem rates ranged from 1 to 100 defects per thousand lines of source code inserted and .1 to 10 defects per thousand lines of source code fielded. Meeting the objective would shift the range to .1 to 10 defects per thousand lines of source code inserted and .01 to 1 fielded.

MEASUREMENT BEST PRACTICE

The strategy for obtaining the deepest possible understanding of software in an application domain is strongly determined by the experimental nature of software and the need to discover information. However, software engineers and computer scientists need to conduct more experimentation using systematic and repeatable processes and need to perform the collection and analysis of data in the confirmation of theory and hypotheses [Tichy 98]. Field studies quantifying software practice are rare. In 612 articles in the software literature during 1985-1995, only seven (7) were classified as field study. Half were classified as assertions, lessons learned, and project monitoring; another third employed no experimentation or were not applicable [Zelkowitz 98].

Although measurement is needed to derive effective policy governing acquisition, development, and operations, there is not yet an industry consensus on the wisdom of creating a national database for software engineering. The issue centers on the use of the data, not on its collection. The worry is that the industry is not ready to use the database appropriately. Clearly the industry can learn to use the database appropriately once it exists. If there are national goals set for software engineering, there must also be a national measurement program and database to track progress and refine goals. Carnegie Mellon University's Software Engineering Institute produced "A Concept Study for a National Software Engineering Database" in July 1992 [Van Verth 92]. The study points out that there are many users for such a database, but few suppliers. The study offers the following observations and advice on establishing a national database:

1. Wide variance may exist in the collection process
2. Common data definitions are needed
3. Goals and questions should precede data collection
4. Confidentiality of the data must be protected

In designing, implementing, and operating the National Software Quality Experiment, it is recognized that the prescription for achieving lasting value in measurement depends on the successful integration of measurement concepts, operationally defined and packaged processes, effective technology transition and adoption including the training of participants and the dissemination of results, and hands-on oversight of the experiment. The prescription for lasting value in measurement revolves around four driving measurement concepts. First, measurement must be aligned with business needs. Second, metrics must be carefully pinpointed and rigorously defined. Third, measurement activities must be built into the normal operation of the organization. Finally, extraordinary steps must be applied to obtain consistency and uniformity in data collection.

Finally, Dr. Vic Basili of the University Maryland provides the following important guidelines on measurement [Wallace 97], [Basili 02]:

1. Establish the goals of the data collection
2. Develop a list of questions of interest
3. Establish data categories
4. Design and test the data collection form
5. Analyze data

While there are numerous problems associated with industrial software measurement, two

problems deserve mention here. One of the difficulties in measurement is that while the design and planning of a measurement program or experiment may be straightforward, its implementation can be fraught with difficulties. An important consideration is the ease of availability of the data to be collected. When the data is easily accessible as a by product of an essential activity within the normal operation of the organization, measurements can be greatly facilitated. When the data is difficult to obtain or requires extra steps, the measurement program may be threatened at the outset. For example, collecting inspection data is straightforward, collecting failure data from testing is more difficult, and collecting failure data from the field is increasingly more difficult. A second important difficulty is sustaining a persistent data collection long enough for interesting trends to reveal themselves. With the fluctuating commitment to measurement as business needs change and new leaders enter the scene, sustaining a measurement program for ten years is a challenge.

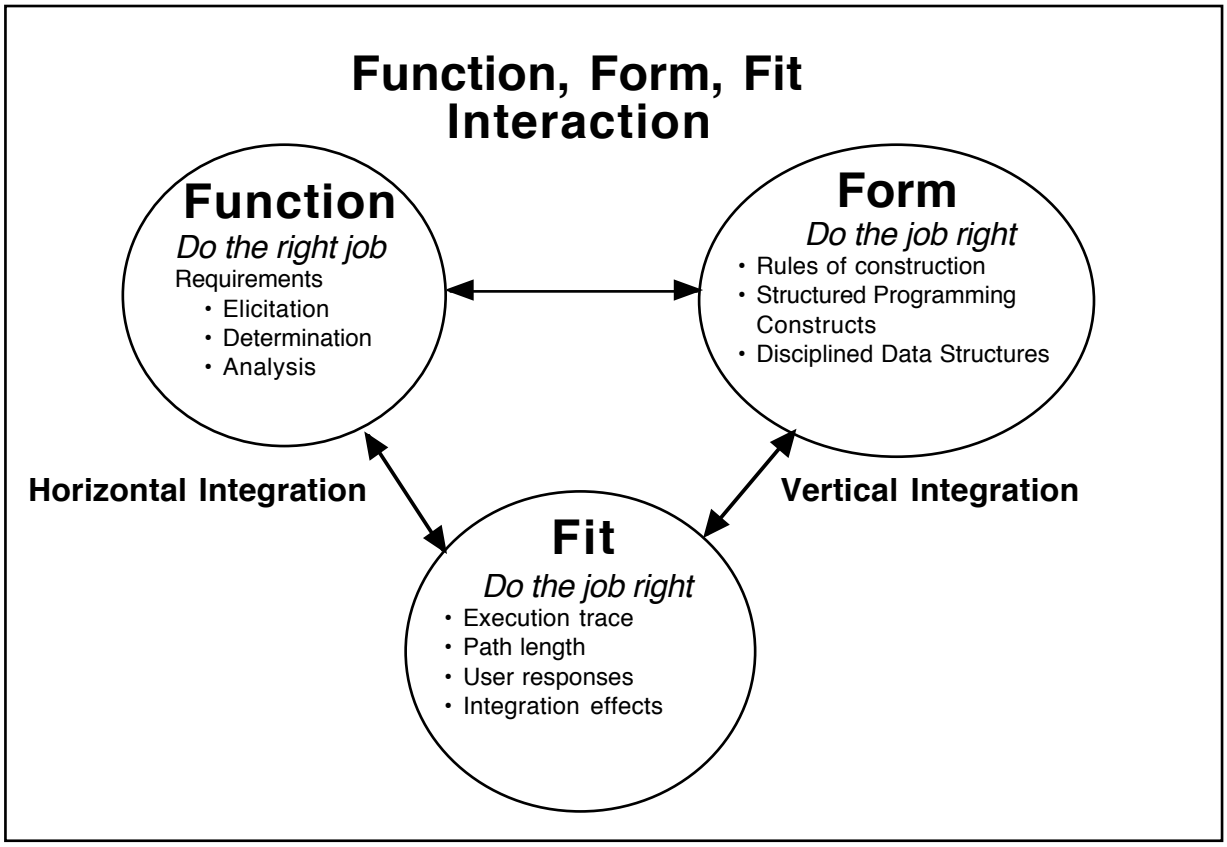
EXPERIMENTAL NATURE OF SOFTWARE

Large scale software development is research in the experimental or laboratory sense operating within a process of experimentation where the hypotheses are organized around function, form, and fit and the inherent uncertainty in specifying, designing, and developing software that will operate harmoniously on the computing platforms intended for the benefit of expected users and their enterprises. The process of experimentation is based on the nature of the life cycle activities; their organization for iteration, prototyping, and incremental development; and the strategies for validating and verifying the artifacts they produce.

Software development involves the discovery of information that is technological in nature, in particular, performance in terms of execution traces, path lengths, user response times, and integration effects both vertical and horizontal. This critical information permits the selection of the best mix of algorithms and data structures for a particular software component and the software system as a whole.

In developing large scale software systems, risk and uncertainty occur when the information available to the project team is insufficient to plan or carry out fully the next step(s) in the development process and the design and programming of the software system. These risks stem from technical choices associated with the function, form, and fit of the software product. And so function, form, and fit become the hypotheses in the process of experimentation used to discover technological information needed to progress from one life cycle activity to another:

- Function is determined through understanding the application domain and interaction with users and the activities of requirements elicitation and determination where the goal is to do the right job. Not all capabilities and features needed to do the right job are known ahead of time. After the start of the project, function may be increased by important new capabilities and features.
- Form is determined by the application domain architecture and its templates, screens layouts, and data models organized to house the capabilities and features that comprise the function in a way that fits the constraints of computer resources.
- Fit is determined by the supply of computer resource timing and capacity and the demand in the load imposed by the use of the capabilities and features of the software system and their resultant execution traces, path lengths, memory utilization, system resource queues, and integration effects. The behavior of the system is influenced by the interaction of this supply and demand and is evidenced by the timeliness of user response times. Doing the job right requires obtaining the best fit which is a complex activity requiring specialized tools capable of identifying integration bottlenecks that can then be reprogrammed to utilize less time or memory.



EXPERIMENT DESIGN

Experienced software practitioners and managers understand that software development is a process of experimentation involving the continuous discovery of technical information associated with the hypotheses of function, form, and fit of the software product as it moves through the requirements, specification, design, code, test, and maintenance activities of the life cycle. Reasoning about, understanding, and predicting the behavior of defects experienced at different stages is the basis for managing software risk and uncertainty.

Several indicators characterize the organization capability to perform effective experimentation. When these indicators are present and embedded in risk management practice, the enterprise is a learning organization.These indicators include setting goals; defining, collecting, and analyzing measurement and metrics; maintaining a repository of results; providing feedback on results to participants and stake holders; packaging experiment artifacts for consistent repetition; and combining multiple experiments.

The design of an experiment includes the hypothesis and questions to be answered, factors that characterize and distinguish the context or domain of study, response variables in the form of measurements and derived metrics, and results that correspond to the hypothesis and address the questions.

- Some hypotheses and questions may be known early; others may be invented as results and studies dictate and suggest.
- Factors define or characterize analysis bins individually or in combination.
- Response variables are the measurements and metrics associated with Software Inspection Lab Operations and defect type distributions.
- The results are the association of the response variables and their measurements and

metrics to the hypotheses and questions and the judgments that are made.

As a software manager on several large scale software projects, there were many questions to which I sought answers. Some of these questions are asked and answered in the experiment including:
1. To what extent is there a continuing stream of requirements changes?
2. What are the leading types of errors?
3. Are errors traced to people or process?
4. Is a standard development process followed?
5. To what extent are wrong software functions being developed?
6. To what extent are there shortfalls in real time performance?
7. Is gold plating a problem?

Other questions of interest are not answered in the experiment. For example:
1. What types of defects are present in fielded software products?
2. To what extent do users encounter defects in fielded software products?
While a study focusing on delivered defects would be useful and interesting, this is beyond the scope of the National Software Experiment which depends solely on the engine of software inspections for its data collection and does not utilize defect data from testing or field experience.

NATURE AND ROLE OF THE EXPERIMENT

Experiment is defined as, "*A test made to demonstrate a known truth, to determine the validity of a hypothesis"* [AHDEL 76]. The DOD Software Technology Strategy has provided the basis for the hypothesis that should be tested for validity: "That software problem rates shall be reduced by a factor of ten by the year 2000".

A process of experimentation involves alternatives and choices. Over time, experiment findings are reported and may be acted upon, software practice improvements may be made, and the experiment may become closed loop.

In the practice of software engineering, managers are guided more by myth than by measurement. The experiment provides the framework for measuring critical aspects of software product quality practice. The framework supplies the ingredients needed to install a uniform and consistent measurement methodology. These are thoroughly described in the Software Inspections Mechanism. The predictability of the measurements taken in conducting the experiment provides the basis for assessing the validity of the hypothesis. This is discussed in Experiment Results.

Some of the questions asked and answered in the experiment are:
1. To what extent is there a continuing stream of requirements changes?
2. What are the leading types of errors?
3. Are errors traced to people or process?
4. Is a standard development process followed?
5. To what extent are wrong software functions being developed?
6. To what extent are there shortfalls in real time performance?
7. Is gold plating a problem?

Software inspections are an essential ingredient in fact-based software management. They utilize a reasoning process for conducting a fine-grained, deep-probing evaluation. When combined with automation-based quick-look evaluations, the best balance between efficiency and insight can be obtained. Once installed in the organization, the software

inspection process yields core samples of software product quality. These can be used to benchmark problem rates by defect type among major product areas within the organization. With the benchmark measurements in place, the software inspections process provides a stable, uniform, and persistent mechanism for measuring improvement progress toward the software product goals of the organization.

The National Software Quality Experiment is an ambitious program to define, create, and mange a National Software Engineering Database of software product quality core samples. This micro-level national database is being populated using data collection procedures packaged for operation in the Software Inspection Lab. These core samples of software problem rates provide the foundation to benchmark and measure progress towards the national software quality objective. The comprehensive structure and rigorous definition of the experiment are needed to provide a stable and persistent mechanism for measuring progress to the year 2000 objective. The experiment itself is a lesson in measurement.

Bridging the gap of prediction practice among defects, faults, and failures remains an unsolved problem. Defects are detected early using software inspections as exit criteria for activities in the software life cycle. Faults are detected later through exercise during integration and system test. Failures occur even later during system operation. Yet there is no accepted method for using defect data available early to predict faults and failures that occur later.

The National Software Quality Experiment with its Software Inspection Lab and its repository of core samples uses defect detection to derive metrics capable of calibrating defect leakage prediction and defect leakage type distribution. The question is, “To what extent are Software Engineering Error Prediction Models capable of utilizing defect leakage prediction and defect leakage type distribution to predict faults and failures?” These models include both error count models and time between models.

Verification of Method and Data

The software inspections method used to collect the NSQE data is detailed in the Wiley Publishing Encyclopedia of Software Engineering [O'Neill 02]. The verification of methods used, data collected, results analyzed is accomplished in he following ways:

1. I have attended every inspection session included in the experiment.
2. Organizations utilize the upper and lower control limits derived from the NSQE metrics in their software inspection operations on the factory floor to guide and control the inspections practice.
3. Analysis results and reports are presented at various professional meetings and conferences and in professional journals. The most recent example is the "Return on Investment Using Software Inspections" study presented at the 11th ICSQ in Pittsburgh just this week [O'Neill 01] and the
4. The National Software Quality Experiment results were shared with the research community associated with the Center for Empirically based Software Engineering (CeBASE).

NSQE Seminar History

The results of the National Software Quality Experiment have been reported to industry in dozens of seminars over the past decade. The venues used for disseminating these results have included the Association for Software Quality Control, the Association for Software Quality, the International Conference of Software Quality, the Software

Technology Conference, the Quality Week Conference, the Quality Week Europe Conference, the NASA Software Engineer Workshop, the International Conference on Software Process Improvement, the Software Developer's Conference, and many Software Process Improvement Network sessions.

QAAM '93	Columbia, Md.
Process Conference '94	Washington, D.C.
ASQC International '94	Washington, D.C.
ICSQ International '94	Washington, D.C.
STC '95	Salt Lake City, Utah
STC '96	Salt Lake City, Utah
QW '97	San Francisco, Ca.
QWE '97	Brussels, Belgium
NRC '98	Rockville, Md.
HRSPIN '98	Virginia Beach, Va.
NASA SEW '98	Greenbelt, Md.
CrossTalk, 12/98- Web version	Salt Lake City, Utah
AAQ '99 (Backup)	Washington, D.C.
ISACC'99	Chantilly, Virginia
SSQ 11/99	Herndon, Virginia
ASQ '00	Rockville, Md.
STC '00	Salt Lake City, Utah
DC SPIN '00	Washington, DC
NJ SPIN '00	Rutgers University, NJ
Greater B'more SPIN '01	UMBC, Baltimore, Maryland
11th ICSQ	Pittsburgh, Pennsylvania
Software Education Associates Ltd. '02	Wellington, New Zealand
Software Education Associates Ltd. '02	Melbourne, Australia
1st ICSPI, '02	Washington, D.C.
New Jersey SPI Conference, '03	Woodbridge, NJ

CeBASE Collaboration

The National Software Quality Experiment results were shared with the research community associated with the Center for Empirically based Software Engineering (CeBASE). This is an National Science Foundation (NSF) sponsored initiative consisting of the University of Maryland, University of Southern California, Fraunhofer Center Maryland, University of Nebraska Lincoln, Mississippi State University.

CeBASE seeks to achieve an empirically based process and experience base that contains validated guidelines for selecting techniques and models. One focus of this research is defect reduction techniques. CeBASE is co-directed by Dr. Victor R. Basili and Dr. Barry Boehm who jointly authored the article on the ten questions on software defects that formed the basis for the study [Basili/Boehm 01]. The CeBASE web site (http://www.cebase.org) contains resources for empirical researchers and practitioners, such as tools, data, reports and experimental results.

Is the Fundamental Thesis of the Paper Flawed?

A reviewer of the paper asserted that the fundamental thesis of the paper was flawed. I would like to air the reviewer's concern and my response to it because it may help others to avoid the same misunderstanding. Here is what the reviewer said, "The fundamental thesis of the paper that there is a national quality objective and progress towards achieving that

objective can be tracked via results from inspection training is flawed. A DOD objective is not a national objective. By definition training is provided to people who are unfamiliar with a skill, so it is unreasonable to assume that improvement could be detected through results of subsequent training sessions. The paper could be useful as a benchmark of inspection process performance and should be re-focused in that direction." The 1992 DOD objective, of course, is to reduce software problem by a factor of ten by the year 2000.

In response, I must acknowledge that a DOD objective is not be a national objective. When it comes to software there is no national software policy, there is no national software authority, there are no national software goals, and in fact there is even an absence of consensus among the leaders of profession on the principles of software engineering and its university curriculum. Consequently when it comes to software, a DOD objective is the closest thing to a national objective. But in the interest of correctness, I have conceded to the use of "DOD objective" instead of "national objective".

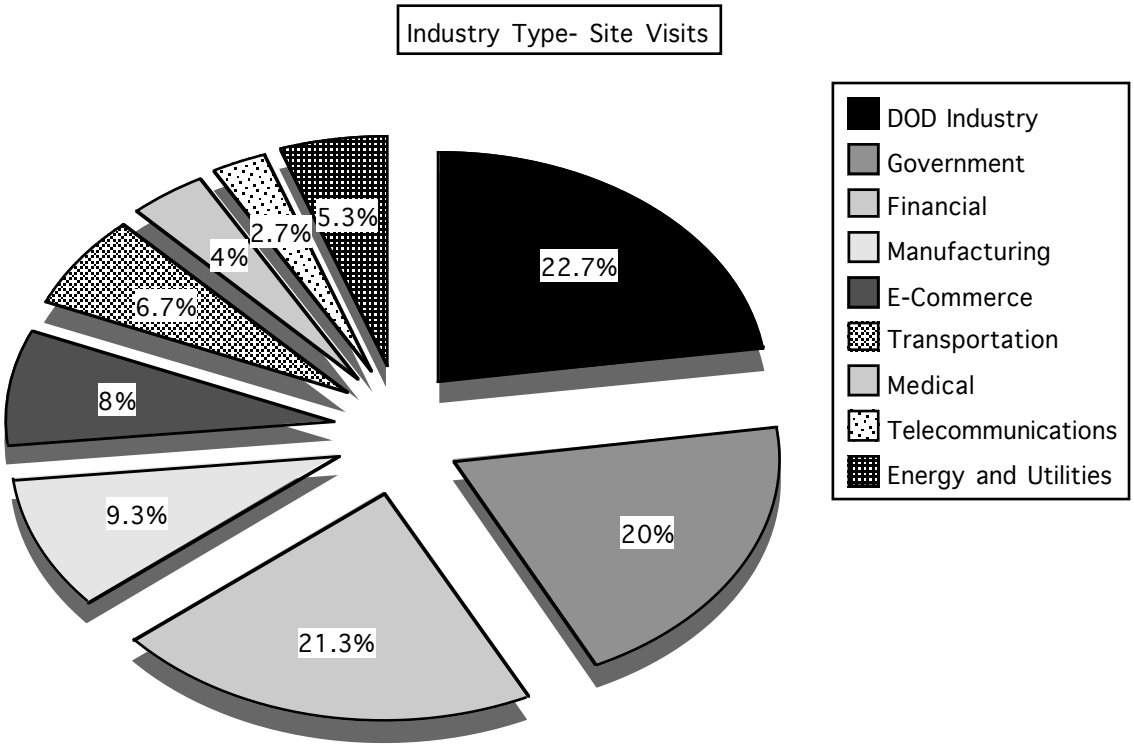
Now to respond to the substance of the reviewer's concern which represents a basic misunderstanding of what is being measured and what is doing the measurement, the improvement being tracked by the National Software Quality Experiment is software product engineering improvement, not improvement in the practice of software inspections. Defects are inserted in the practice of software product engineering; defects are detected in the practice of software inspections. Therefore, it is the hoped for improvement in software product engineering practice that is being measured, and it is software inspections practice that is the measurement mechanism.

The widespread collection of core samples from over sixty organizations spanning over a decade is expected to reveal industry improvement as evidenced by fewer defects inserted during the practice of software product engineering. If fewer defects are inserted, then fewer defects would be detected during software inspections assuming that a uniform process for software inspections [O'Neill 88] is followed and that participants received the same training [O'Neill 89]. This commonality in software inspections practice is in fact true. The same industrial strength software inspections process [O'Neill 02] and the same training program with the same instructor were carried out for over a decade. Exactly 157 course sessions trained over 3,308 participants who conducted 3,040 software inspection sessions producing the data for the experiment. However, the hoped for order of magnitude reduction in problems from software product engineering practice did not materialize. Nevertheless, the software inspections process metrics collected and analyzed in the National Software Quality Experiment are providing a useful performance benchmark for those engaged in software inspections practice.

EXPERIMENT PARTICIPANTS

The participants of the National Software Quality Experiment have been trained in the Software Inspections Course and Lab [O'Neill 89]. Experiment results are drawn from these Inspection Lab sessions. Over fifty participating organizations span government, DOD industry, and commercial sectors and represent a wide range of application domains and product types. The industry types represented include telecommunications, transportation, financial, manufacturing, medical systems, utilities and energy, defense, and e-commerce.

- Accounting, personnel, administration
- Administrative and management decision support
- Airline operations support
- Aircraft jet engine diagnostics, logistics, and maintenance
- Air travel reservations
- Artillery fire control system
- Avionics flight on-board control
- CIO Support
- Control devices for avionics applications
- Credit card application
- Department of State embassy support
- e-commerce
- Electronic warfare
- Energy operations management
- FAA communications
- Factory line support
- Finance and accounting services
- Global positioning system user sets
- Government payment system
- Information and accounting
- Insurance and medical information
- Insurance brokering
- International banking
- Joint Chiefs of Staff support
- Medical devices and diagnostics
- Medical information system
- Naval surface weapons system
- Pre and post flight space application
- Securities trading
- Stock market back office operations
- Telecommunications
- Small tool manufacturing



The participants and their applications are listed below:

A	Communications
B	Finance, personnel, administration
C	Command and control center
D	Pre and post flight space application
E	Command and control center
F	Avionics flight on-board control system
G	Administrative and management decision support
H	Medical information system
I	Global Positioning System user set
J	Joint Chiefs of Staff support
K	Avionics flight on-board control system
L	Artillery fire control system
M	Surface ship command and control
N	FAA communications
O	Communication command and control
P	Naval surface weapons system
Q	Control devices for avionics applications
R	Control devices for commercial applications
S	Aircraft jet engine diagnostics and maintenance
T	Financial services
U	Insurance and medical information systems
V	Government accounting

W	Aircraft logistics and maintenance
X	Telecommunications
Y	Aircraft jet engine diagnostics a
Z	FAA Air traffic control
AA	Financial services
AB	Naval surface weapons system
AC	International banking
AD	Electronic commerce
AE	Credit card application
AF	Department of State embassy s
AG	Factory line support
AH-AJ	Credit card application
AK	Factory line support
AL	Electronic warfare
AM	Medical diagnostics
AN	Medical devices
AP	Financial systems
AQ	Information and accounting
AR	Manufacturing
AS	International banking
AT	Insurance brokering
AU	Air travel reservations
AV	Finance and accounting services
AW	International banking
AX	Securities trading
AY	Airline operations support
AZ	e-commerce
BA	CIO support
BB	Energy operations management
BC	Information Systems
BD	Small tool manufacturing
BE	FAA communications
BF	Government payment system

National Software Experiment Participants

SEPARATING SIGNAL FROM NOISE

While the National Software Quality Experiment faithfully conforms to the well defined Software Inspections Process [O'Neill 02] which has achieved stability through long term use, the Software Product Engineering (SPE) processes [Paulk 95] that produce the artifacts being inspected may not be stable and may lack faithful conformance.

The variation in process performance includes both process noise and process anomalies. The degree to which the SPE processes are stable and conforming is the degree to which process noise is minimized [SEI 97].

ORGANIZATION OF FINDINGS

The findings of the National Software Quality Experiment are organized along several dimensions including year, software process maturity level, organization type, product type, programming language, and global region. These dimensions provide a framework for populating an interesting set of analysis bins with appropriate core samples of software product quality.

SOFTWARE INSPECTIONS MECHANISM

Setting the Standard of Excellence

The industry continues to evolve the definition of quality and the indicators of a mature quality process. Early on, conformance to requirements was recognized as an important quality characteristic. The software product must satisfy every "shall" in the requirements document. To this has been added the characteristic of defect-free [Joyce 89]. A software product is considered defect-free when it attains Six Sigma quality, which is three to four errors per million opportunities to fail. Conformance to requirements and defect-free characteristics are necessary but not sufficient conditions. More is needed.

Customer satisfaction is the quality characteristic that now occupies center stage. Customer surveys and feedback cover all aspects of the software product including engineering, construction, operations, and support. To this is being added the characteristic of value. Value is applying the best capability of the organization to what the customer needs most. Where the organization's strategic planning process operates to continue to align its capabilities with customer needs, the maturity of its quality process is ranked high.

Measuring the quality of an evolving software product can be accomplished by conducting strict and close examinations of its requirements, specification, architecture, design, code, and test artifacts. The quality characteristics measured using this software inspection mechanism include conformance to requirements and defect detection and leakage among baseline artifacts. Other mechanisms are needed to measure customer satisfaction and value characteristics.

In producing software systems that meet customer needs, it is necessary to do the right job and to do the job right. Software inspections address the issues of construction and workmanship needed to do the job right. This is done by setting the standard of excellence and then disciplining the organization to meet the standard set. Simply setting the standard changes the calculus of software product quality. The attention of the organization's practitioners, project manager, and senior managers will be riveted on software product quality.

Software inspections practice employs the strongly preferred indicators from the standard of excellence spanning completeness, correctness, style, rules of construction, and multiple views.

- Completeness is based on traceability among the requirements, specifications, designs, code, and test procedures.
- Correctness is based on reasoning about programs through the use of informal verification and correctness questions derived from the prime constructs of structured programming, their composite use in proper programs, and the disciplined data structures they manipulate [Linger 79].
- Style is based on project specified style guidance for block structuring, naming conventions, commentary, alignment, and templates for repeating patterns.
- Rules of construction are based on the software architecture and the specific protocols, templates, and conventions used to carry it out.
- Multiple views are based on the various perspectives required to be reflected in the product including the programmer, tester, user, computer resource loading as well as

safety, security, and the earlier year 2000 problem [Basili 96].

VALUE OF SOFTWARE INSPECTIONS

Software inspections[4] provide an immediate and concrete step that every organization can take to improve its process maturity. They provide a powerful mechanism for improving software product quality by detecting and correcting defects early and preventing their reoccurrence. Software inspections accomplish this by conducting close and strict examinations of software requirements, specification, design, code, and test artifacts. They provide a vantage point for fact based software management, one not occupied by designers, programmers, and testers.

Organizations are increasingly using software inspections as an integral process in the development of quality software. The installation of a software inspections process initially results in detecting 50% of the inserted errors. As an organization acquires skill and refines its process, the detection rate increases to 60-90% within eighteen months. The cost to correct defects found in testing greatly exceeds the correction cost following an inspection. In fact, IBM Rochester, winner of the Malcolm Baldrige Award, reported that defects leaking from code to test cost nine times more to detect and correct, and defects leaking from test to the field cost thirteen times more [Lindner 91]. Consequently, the application of an industrial strength software inspections process reduces development costs, shortens delivery schedules, and promotes higher reliability of operational software products in the field.

The measurement mechanism used in the National Software Quality Experiment adapts and packages the defined processes, product checklists, participant behaviors, defect type definitions, and reports found in software inspections into an integrated operation. The result is a measurement tool that can be applied uniformly and consistently using participants from projects trained for their roles in the Software Inspection Lab.

An example may help illustrate why a leaked defect costs more. A code defect that leaks into testing may require multiple test executions to confirm the error and additional executions to obtain debug information. Once a leaked defect has been detected, the producing programmer must put aside the task at hand, refocus attention on correcting the defect and confirming the correction, and then return to the task at hand. The corrected artifact must then be reinserted into the software product engineering or product release pipeline [O'Neill 03].

ELEMENTS OF SOFTWARE INSPECTIONS

The elements of the Software Inspections Process include a structured review process, defined roles of participants, system of checklists, and forms and reports [O'Neill 02]. These are fully described in the article on “Peer Reviews” found in the Encyclopedia of Software Engineering Second Edition, John Wiley &Sons, Inc., January 2002. This article can be viewed at http://members.aol.com/ONeillDon/nsqe-assessment.html.

- A structured review process is a systematic procedure integrated with the activities of the life cycle model selected. The process is composed of planning, preparation, entry criteria, conduct, exit criteria, reporting, and follow up [Fagan 76], [Gilb 93].

- The role of each participant in the structured review process is defined. These roles include the moderator, producer, reader, recorder, reviewer, and manager. Each role is characterized by particular skills and behaviors [Freedman 90] .

[4] Software inspections were pioneered by Michael Fagan at IBM in the 1970's

- A system of checklists govern each step of the structured review process and the review of the product itself, objective by objective. Process checklists are used as a guide for each activity of the structured review process. Product checklists house the strongly preferred indicators that set the standard of excellence for the organization's software products [O'Neill 88].

- Forms and reports provide a uniformity in recording issues at all software inspections, reporting the results to management, and building a data base useful in process management. Data collection utilizes three recording instruments: Inspection Record, Inspection Reporting Form, and Report Summary Form. [Ebenau 94].

STRUCTURED REVIEW PROCESS

Planning is done by management early in the project. Planning identifies the life cycle activities and product artifacts including top level designs, detailed designs, and code to be inspected. The schedule for each software inspection is recorded in the project's software plan. A moderator is assigned to each software inspection, and moderator training is scheduled as necessary.

Preparation is done by the moderator a few weeks before the inspection. The readiness of the product for inspection is assessed. The moderator obtains the reader, recorder, and reviewers and instructs them on their roles. The moderator ascertains the status of the baseline change activity. The overview session is conducted and the review materials are distributed to all participants.

Entry Criteria are checked by the moderator on the day of the review. Before the software inspection begins, the moderator must be certain that the product is ready to be inspected and that the participants are ready to inspect it. The moderator verifies that trained and briefed participants are in place that all participants have received the product and checklists. The recorder notes the preparation time spent by each participant. Finally any changes to the baseline are identified. Where the entry criteria are not met satisfactorily, the moderator may reschedule the inspection.

The inspection is **conducted** by the moderator, recorder, producer, reader, and reviewers; the manager and the consumer do not attend. Some key principles govern participant behavior during the inspection conduct:

1. The product is reviewed, not the person.
2. The inspection is limited to periods of peak concentration, usually 2-4 hours.
3. Issues are identified, not proposed solutions.

Each product component is inspected using each checklist. Participants have prepared for the inspection. Each participant in turn is asked whether there is an issue for the product component and checklist now before the group. If so, the issue is stated and recorded. The producer may wish to obtain clarification of the issue at the time it is raised, but there is no need for the producer to defend or even explain the approach taken. The producer will have the opportunity to resolve the issue during the fillip activity.

Exit Criteria are checked by the moderator at the close of the inspection. The moderator verifies that all product components and checklists have been reviewed. Reviewers are asked if there are any additional issues to be raised. The recorder then reads all the issues raised. The producer is given the opportunity to make any closing comments. This concludes the participation of the reader and reviewers.

The moderator, with the help of the recorder, **reports** the results to management within a week or so. This report provides a review summary, statistics on the inspection process, key issues, and fillip recommendations. This concludes the participation of the moderator and recorder.

The **followup** rework on the product is performed by the producer. The process is managed by the manager in the usual way.

DEFINED ROLES OF PARTICIPANTS

The **manager** is active in the planning, preparation, reporting, and fillip activities. In planning, the manager identifies and schedules all software inspections in the project's software plan. The manager identifies resource needs and allocates them to each inspection. Moderators are assigned, and moderator training is arranged. The manager does not attend the conduct activity. After the software inspection is conducted, the manager receives the report and oversees any fillip.

The **producer** is active during the preparation, entry criteria, conduct, exit criteria, and fillip activities. The producer is responsible for creating the materials to be inspected. The producer attends the inspection as a reviewer and is expected to raise
issues. From time to time the producer may offer a technical explanation of the product. The producer expects criticism of the product but does not offer any defense as issues are raised. Where an issue is surfaced that is not understood by the producer, a dialogue may be needed to obtain clarification. At the conclusion of the conduct activity, the producer is afforded the opportunity to comment on the inspection. The producer performs the fillip actions resulting from the inspection.

The **moderator** is the keystone of the software inspection process and is active in the preparation, entry criteria, conduct, exit criteria, and reporting activities. The moderator directs the activities of the software inspection. The moderator briefs all participants of the inspection on their roles in the structured review process and administers the preparation activity including an overview meeting. The moderator directs the entry criteria, conduct, and exit criteria activities, facilitating interaction among the participants. The moderator intervenes as little as possible but as much as necessary to ensure an effective and efficient software inspection. A skillful moderator recognizes the needs of the various participants and restrains any intervention until it is clearly required. Moderators who limit intervention enhance the feeling of responsibility of each participant and give real meaning to the term "peer review". The moderator collaborates with the recorder in preparing the report

The **recorder** is active in the preparation, entry criteria, conduct, exit criteria, and reporting activities. During the entry criteria activity the recorder notes the preparation time of each participant. The recorder notes all issues and concerns raised by the participants of the inspection. For each issue, the recorder captures a description of the issue, the location in the product, the checklist and entry that prompted the issue, and other defect and resolution attributes. The role of recorder is to be transparent to the process of conducting the inspection in recording all issues completely and accurately. This requires a high degree of judgment and technical knowledge.

The **reader** is active in the preparation, entry criteria, conduct, and exit criteria activities. Where necessary, the reader may read parts of the the product aloud to the other participants of the inspection. This permits the producer to assume a low profile and

minimizes the need for producer-reviewer interaction, thereby promoting ego less programming. The reader helps the group to focus attention on the relevant parts of the product. The reader is not expected to read line by line. This may be necessary at times when difficulties are encountered in a section of the product artifact. The reader may instead direct attention to a program unit or a construct.

The **reviewers** are active in the preparation, entry criteria, conduct, and exit criteria activities. A reviewer is expected to spend sufficient time preparing and to raise issues and concerns about the product. Reviewers accept the discipline imposed by the round robin, checklist structure of the inspection. In return for accepting these responsibilities and disciplines, reviewers are assured of an uninterrupted opportunity to raise issues.

SYSTEM OF CHECKLISTS

Completeness is based on traceability among the requirements, specification, design, code, and test artifacts. Completeness analysis reveals what predecessor artifact sections have not been satisfied as well as the inclusion of extra fragments.

1. Has traceability been assessed?
2. Have all predecessor requirements been accounted for?
3. Were any product fragments revealed not to have traceability to the predecessor requirements?

Correctness is based on reasoning about programs through the use of informal verification and correctness questions derived from the prime constructs of structured programming and their composite use in proper programs. Input domain and output range are analyzed for all legal values and all possible values. Adherence to project specific disciplined data structures is analyzed.

1. Is the function commentary satisfied?
2. Does the loop terminate?
3. Is a one time loop acceptable?
4. Is the control variable modified in the loop?
5. Is the loop initialized and terminated properly?
6. Is the domain partitioned exclusively and exhaustively?
7. Does the input domain span all legal values?

Style is based on project specified style guidance based on block structured templates. Semantics of the design and code are analyzed for correspondence to the semantics used in the requirements, specifications, and design. Naming conventions are checked for consistency of use; and commentary, alignment, upper/lower case, and highlighting use are checked.

1. Are style conventions for block structuring followed?
2. Are naming conventions followed?
3. Do the semantics of the product correspond with the requirements?
4. Are style conventions for commentary followed?

Rules of construction are based on the software architecture and its specific protocols, templates, and conventions used to carry it out. For example, these include interprocess communication protocols, tasking and concurrent operations, program unit construction, and data representation.

1. Are guidelines for program unit construction followed?
2. Is the interprocess communication protocol followed?
3. Are data representation conventions followed?

Multiple views are based on the various perspectives required to be reflected in the product. During execution many factors must operate as intended including initialization, timing of processes, memory use, input and output, and finite word effects. In generating the software, packaging considerations must be coordinated including program unit construction, program generation process, and target operations. Product construction disciplines of systematic design and structured programming must be followed as well as interfaces with the user, operating system, and physical hardware.

1. Have execution considerations been assessed including timing, memory use, input and output, initialization, and finite word effects?
2. Have packaging considerations been assessed including program unit construction, program generation process, and target operations?
3. Have user interface considerations been assessed?

FORMS AND REPORTS

All data collected and reported during the Software Inspections Process is recorded by the recorder. This includes data about the product being inspected and about the inspection process itself. The value of the experiment hinges on the uniformity and consistency of the recording process. The requirements for data collection are carefully defined and are an important part of the training for the Software Inspection Lab. There are three recording instruments: Inspection Record, Inspection Reporting Form, and Report Summary Form.

The **Inspection Record** gathers data about the conduct of the process in the Software Inspection Lab. The name of the project, the specific product component, and the date of the inspection session are recorded. The size of the product is recorded. This metric is defined as the number of non-blank lines. Where an organization has an established definition for a line of code, this metric is recorded. Each participant is listed by name and identified by role including moderator, recorder, reader, reviewer, and producer. During the entry criteria process, the recorder asks each participant to state the number of minutes of inspection preparation effort expended, and this is recorded. Each product checklist selected for use in this session is noted. The time spent in the inspection conduct is recorded as the wall clock time for the start and end of the session.

The **Inspection Reporting Form** gathers data about each issue raised in the inspection session. In addition to a description of the issue, important information about the issue is collected . Each issue is assigned a sequence number. In inspecting a product component, several units may be inspected in a session. The component unit name is recorded. The page and line number pinpointing the issue is entered. Where page and line numbers are not present, the pages are numbered manually; and page position is identified as top, middle, bottom. The checklist name and entry number most closely corresponding to the issue are entered, for example, completeness and correctness. A defect category is assigned as missing, wrong, or extra. A defect severity is assigned as major or minor. The appropriate defect type is assigned.

Defect type definitions include:

Interface: error in parameter list

Data: error in data definition, initial value setting, or use of disciplined data structures and their operations

Logic: error revealed through informal correctness questions spanning prime constructs of structured programming

I/O: error in formatting, commanding, or controlling I/O operations

Performance: error in managing or meeting constraints in computer resource allocations and capacities for CPU, memory, or I/O

Functionality: error in stating intended function or in satisfying intended function through refinement and elaboration

Human Factors: error in externally visible user or enterprise interface or interaction

Standards: error in compliance with product standards for construction or integration including programming style guidelines, open system interfaces, or guidelines for the application domain architecture

Documentation: error in guidance documentation

Syntax: error in language defined syntax

Maintainability: Error in good practice impacting the supportability and evolution of the software product

Other: any other error

Defect types discontinued include:

Test Environment: Limitation, incompatibility, or error in test bed

Test Coverage: Shortfall in requirements or functionality covered by test exercisers

The **Report Summary Form** is constructed at the close of the inspection session. It is a frequency count of issues presented as a matrix of defect types by defect severity (major, minor) and defect category (missing, wrong, extra). This form serves several purposes. Since it cannot be constructed unless the recorder has completed the Inspection Reporting Form, it serves as an on the spot check of the recorder. Once completed, weaknesses are highlighted and some opportunities for defect prevention suggest themselves. When the results of enough inspection sessions are overlayed on the Report Summary Form, these frequency counts divided by the total defects can serve as the probability of occurrence for each defect type, defect severity, and defect category.

SOFTWARE INSPECTION LAB

The centerpiece of the National Software Quality Experiment is the Software Inspection Lab [O'Neill 89]. Here data collection procedures, product checklists, and participant behaviors are packaged for operational use. In order to ensure the uniform application of the experiment and the collection of consistent measurements, each participant is trained in the Software Inspections Process. This course is composed of one day of lecture, prerecorded video, and student participation focusing on the elements of software inspections followed by the Inspection Lab.

Learning to organize and conduct software inspections involves learning both knowledge and skills. Basic knowledge of software engineering, of models of software engineering processes, and of how programs can be verified within them is essential to understanding why and how software inspections can be effective. Knowledge of the steps and elements of the software inspection process as well as the skills involved in performing them form the basis for organizing the process in a specific organization and project. Finally, various performance and human relations skills are involved in actually conducting software inspections. Such skills are best learned through a sequence of verbal understanding, modeling, and practice.

To apply the behavior, skills, and knowledge acquired during day one, the Inspection Lab provides the opportunity for each participant to play each defined role. In this way, the concepts learned are put to immediate use in a realistic situation, difficulties are encountered and overcome, and the confidence to reapply these skills on a real project is developed. The realism of the Inspection Lab is achieved by requiring each participant to bring five to ten pages of actual detailed design or code to be inspected.

Planning for the lab takes place at the close of the day one session. The outcome of this planning session is the assignment of a defined role for each participant in each inspection session. Everyone is expected to prepare for the lab by reviewing the detailed design or code to be inspected for compliance with product checklists.

In the conduct of the Inspection Lab, inspections are conducted on each artifact. Each participant plays the defined role assigned for each inspection. The recorder collects all the data using the standard forms and reports. This includes an identification of each participant by name and role, the preparation effort applied, the wall clock time of the inspection session, the product component name, and size of the artifact being inspected. During the conduct process, the recorder describes each issue along with information on defect type, category, severity, and origin. The issue data collected is tabulated into a summary matrix at the close of the inspection session.

The moderator is assisted in the operation of the Inspection Lab by the checklist entries for entry criteria, conduct, and exit criteria. The checklist entries defining the conduct of the lab include:

Entry Criteria

1. Has the preceding life cycle been concluded?
2. Are review participants in place and briefed?
3. Have all participants received all the review materials and checklists?
4. How many minutes of preparation did each participant perform?
5. Are there any changes to the baseline?

Conduct

1. Are there any issues in completeness?
2. Are there any issues in correctness?
3. Are there any issues in style?
4. Are there any issues in rules of construction?
5. Are there any issues in multiple views?
6. Are there any issues in technology?

Exit Criteria

1. Have all product elements been inspected?
2. Have all checklists been processed?
3. Have the inspection results been recorded?
4. Would the recorder read back the issues?
5. Have metrics been collected?
6. What should be the disposition of the inspection?
7. Would the producer like an opportunity to comment?

After all the inspections have been conducted and the data collected, participants may analyze the results to identify root causes of defects discovered in order to prevent their reoccurrence. The ranking of defect types by defect frequency provides a clear indication of any patterns of neglect.

DATABASE DESIGN

The database for the National Software Quality Experiment has been created and is being sustained as a steady stream of core samples of software quality is entered. Findings are analyzed and reported on an annual basis.

The database is organized by organization, year, software process maturity level, organization type, product type, programming language, global region, and industry type.

Database Field Definitions

1. Organization
    - Anonymous identifier
    - Organization name
2. Year [pre-1992-2000]
3. Software Process Maturity Level [Level 1-3]
4. Organization Type
    - Commercial
    - DOD Industry
    - Government
5. Product Type
    - Embedded
    - Organic
    - Packaged
6. Programming Language Type
    - Old Style
    - Modern
7. Global Region
    - North America
    - Asia Pacific
    - Europe
    - Latin America
8. Industry Type
    - Telecommunications
    - Transportation
    - Financial
    - Manufacturing
    - Medical Systems
    - Utilities and Energy
    - Defense
9. Application
10. Process Measurement
    - Preparation Effort in Minutes
    - Conduct Time in Minutes
    - Major Defects
    - Minor Defects
    - Size in Lines of Code
    - Size in Pages
    - Number of Participants
11. Process Metrics
    - Preparation Effort Per Defect
    - Preparation Effort Per Major Defect
    - Major Defect Per Thousand Lines
    - Minor Defects Per Thousand Lines

- Size Per Conduct Hour
- Defects Per Session
- Preparation Effort/Conduct Effort
- Lines Per Session
- Return on Investment

12. Defect Type
    - Interface
    - Data
    - Logic
    - I/O
    - Performance
    - Functionality
    - Human Factors
    - Standards
    - Documentation
    - Syntax
    - Maintainability
    - Other

Spreadsheet Expressions

1. Consider the raw data being provided as proprietary
2. Defect type data is missing on early instances and a few later ones
3. Some changes in defect type definitions
   - -Test Environment and Test Coverage defect types fall out of usage
   - -Maintainability defect type was added
4. Spreadsheet road map and expressions:

A	Title	analysis bin name
B	Prep Effort	minutes
C	Conduct Time	minutes of wall clock time
D	Major Defects	affecting execution
E	Minor Defects	not affecting execution
F	Lines of Code	non-blank lines
G	Pages of Doc	pages
H	Sessions	number of inspections
I	Prep/Defect	"=B/(D+E)"
J	Prep/Major	"=B/D"
K	Major/Size	"=D/(F/1000)"
L	Minor/Size	"=E/(F/1000)"
M	Size/Conduct	"=F/(C/60)"
N	Defects/Session	"=(D+E)/H"
O	Prep/Conduct	“=B/(C*4)”
P	Return on Investment	“=((D*9)+E)/((B+(C*4))/60)”

Notes:

1. B through H, General numeric format
2. I through P, Fixed Numeric format, Precision 2
3. I through P, Graphs using Bar Format using Labels shown
4. I through P cell expressions include row number, ie, =B10/(D10+E10)

EXPERIMENT RESULTS

RESULTS SUMMARY

Ralph Waldo Emerson said, "The years teach us things the days never knew". The National Software Quality Experiment has been accumulating a steady stream of core samples for its micro-level national database. These results have provided a benchmark of software product quality measurements useful in assessing progress towards the national software quality objective for the year 2000. These results are highlighted below in the discussion of the common problems pinpointed, defect category and severity data summary, Inspection Lab operations, and the ranking of defect types.

Common Problems

Analysis of the issues raised in the experiment to date has revealed common problems that reoccur from session to session. Typical organizations which desire to reduce their software problem rates should focus on preventing the following types of defects:

1. Software product source code components are not traced to requirements.
   As a result, the software product is not under intellectual control, verification procedures are imprecise, and changes cannot be managed.
2. Software engineering practices for systematic design and structured programming are applied without sufficient rigor and discipline.
   As a result, high defect rates are experienced in logic, data, interfaces, and functionality.
3. Software product designs and source code are recorded in an ad hoc style.
   As a result, the understandability, adaptability, and maintainability of the software product are directly impacted.
4. The rules of construction for the application domain are not clearly stated, understood, and applied.
   As a result, common patterns and templates are not exploited in preparation for later reuse.
5. The code and upload development paradigm is becoming predominant in emerging e-commerce applications.
   As a result, the enterprise code base services only the short term planning horizon where code rules and heroes flourish, but it mortgages the future where traceable baseline requirements, specification, and design artifacts are necessary foundations.

Defect Category and Severity

An earlier analysis was conducted on defect category and defect severity. The defect severity metric revealed that 14.27% of all defects were major, and 85.73% minor. Defect category distinguishes missing, wrong, and extra. For major defects, 7.44% were missing, 5.95% wrong, and .88% extra. For minor defects, 49.76% were missing, 27.63% wrong, and 8.32% extra.

Defect Severity and Category Summary				
	Missing	**Wrong**	**Extra**	**Total**
Major	7.44	5.95	.88	14.27
Minor	49.76	27.63	8.32	85.73
Total	57.20	33.60	9.20	100.00

Inspection Lab Operations

Through 2002 there have been 157 Inspection Labs in which 3,324 participants were trained and conducted 3,040 inspection sessions. A total of 1,020,229 source lines of code have received strict and close examination using the packaged procedures of the lab. There have been 181,471 minutes of preparation effort and 71,283 minutes of conduct time expended to detect 14,903 defects.

Of these 14,903 defects, 2,512 were classified as major and 12,391 as minor. A major defect effects execution; a minor defect does not. It required 12.18 minutes of preparation effort on the average to detect a defect. To detect a major defect required 72.24 minutes of preparation effort on the average. On the average, 858.74 source lines of code were examined each inspection conduct hour. There were 2.46 major defects detected in each thousand lines, and 12.15 minor defects. There were 4.90 defects detected in inspecting 335.60 lines per session. The preparation effort was 0.64 of conduct effort. The Software Inspection Labs produced a return on investment of 4.50.

INSPECTION LAB OPERATIONS

Sessions	Prep Effort	Conduct Time	Major Defects	Minor Defects	Size in Lines
3,040	181,471	71,283	2,512	12,391	1,020,229

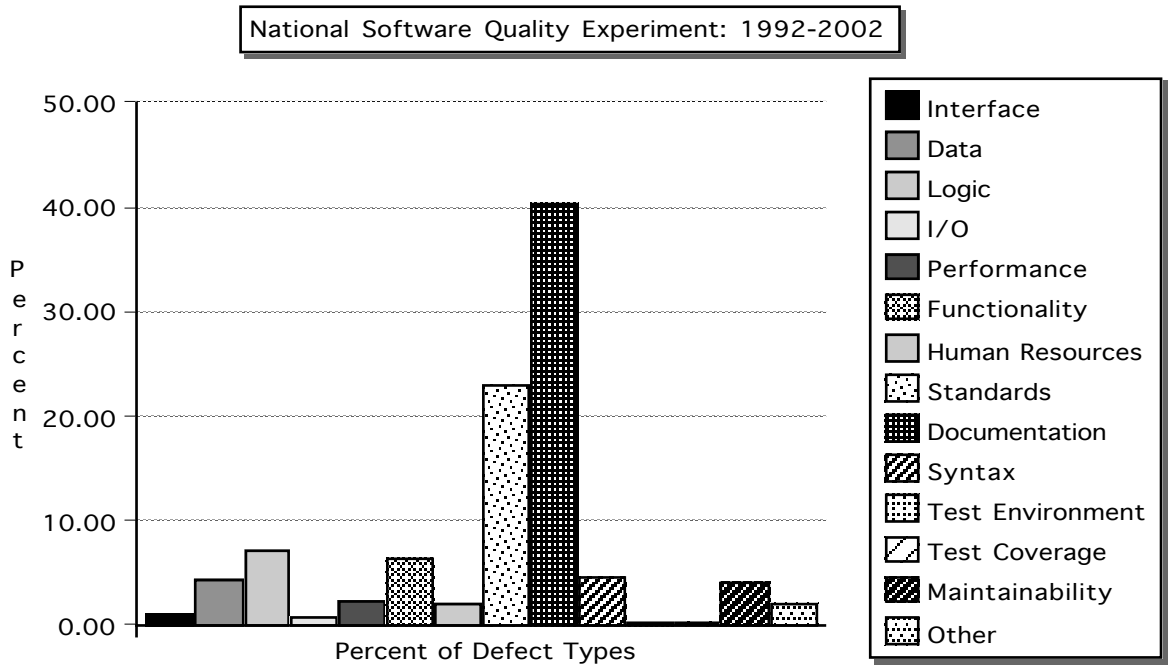
Metrics:

1.	12.18	minutes of preparation effort per defect
2.	72.24	minutes of preparation effort per major defect
3.	2.46	major defects per thousand lines
4.	12.15	minor defects per thousand lines
5.	858.74	lines per conduct hour
6.	4.90	defects per session
7.	0.64	preparation/ conduct effort
8.	335.60	lines per session

Defect Type Ranking

The foremost defect types that accounted for 91% of all defects detected are shown below. Documentation, specifically lack of traceability, accounts for 40.51%of all defects. Standards accounts for about 23.20%. Both of these are examples of organizational neglect. These are followed by logic, functionality, syntax, data, and maintainability defect types which are examples of programmer neglect.

Documentation	40.51%	error in guidance documentation
Standards	23.20%	error in compliance with product standards
Logic	7.22%	error revealed through informal correctness questions
Functionality	6.57%	error in stating or meeting intended
Syntax	4.79%	error in language defined syntax compliance
Data	4.62%	error in data definition, initial value setting, or use
Maintainability	4.09%	error in good practice impacting the supportability and evolution of the software product

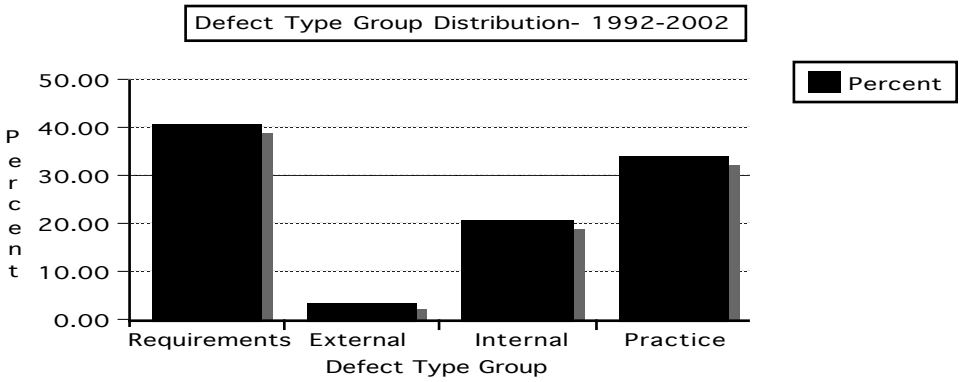


Defect Type Groups

Defect types can be organized for analysis as follows:

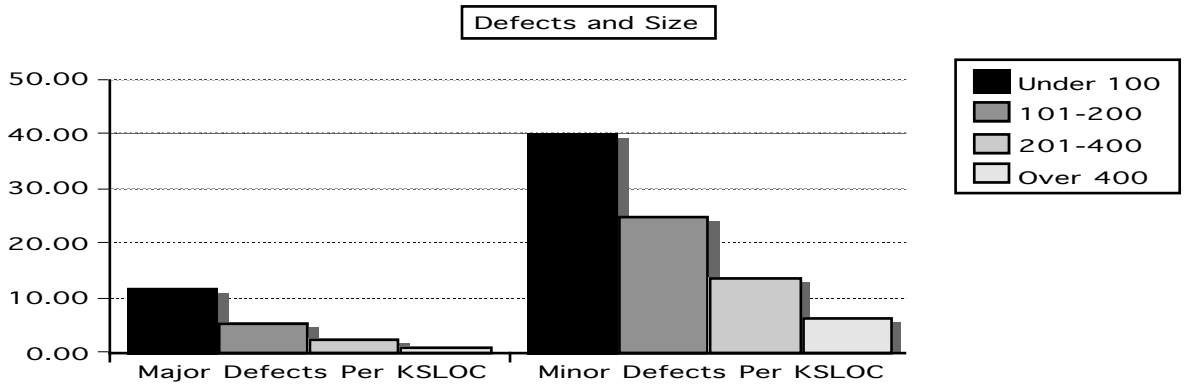
- Requirements
  - ◇ Documentation
- External
  - ◇ Interface, human factors, I/O
- Internal
  - ◇ Functionality, logic, data, performance
- Software practice

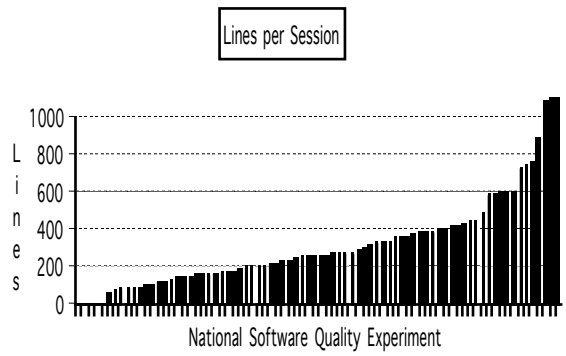
◇ Syntax, standards, maintainability, other

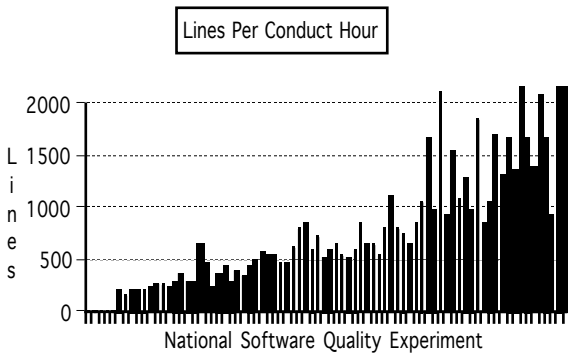


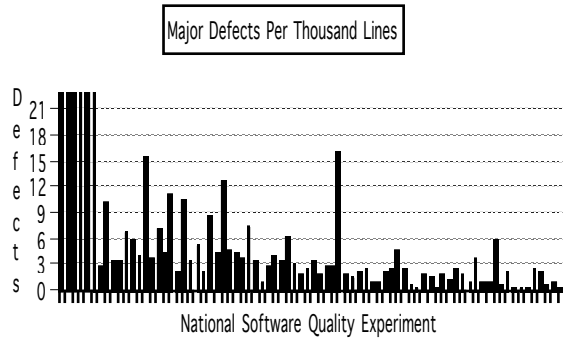
DEFECT RATES AND PRODUCT SIZE

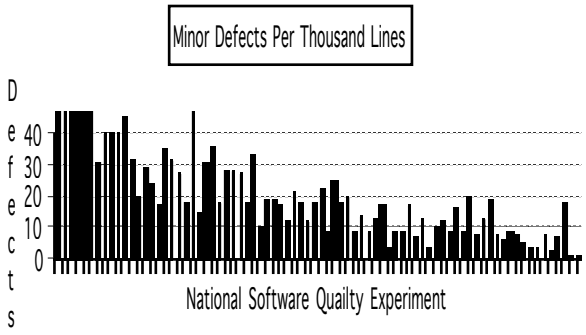
An analysis was conducted on defect rates and product size. The rate of defects detected in the Software Inspection Lab reveals an inverse relationship to product size. All programs contain a beginning, an end, and a context for operation within the larger system. Starting, finishing, and fitting in are all more error prone than the body of the program which gives it size.











A Risk Management Application

The probability of occurrence of defect types has application in managing software risks on the project. Here it is useful to make a careful distinction between sources of risk, risks, and problems. A problem is a previous risk whose consequences are being played out.

The **documentation** defect type accounted for 40.51% of all defects. The lack of a requirements traceability mechanism is a principal source of risk contributing to the documentation defect type. Left unattended, this source of risk will continue at a high probability of occurrence, and there will be problems in baseline and change management and in ease of maintenance and adaptability.

The **standards** defect type accounted for 23.20% of all defects. The lack of a programming style guide or the lack of enforcement of the style guide is a principal source of risk contributing to the standards defect type. Left unattended, this source of risk will continue at a moderately high probability of occurrence, and there will be problems in ease of maintenance and adaptability.

The **logic** defect type accounted for 7.22% of all defects. The lack of rigor in applying systematic design, structured programming, and disciplined data structures is a principal source of risk contributing to the logic defect type. Left unattended, this source of risk will continue at a moderate probability of occurrence, and there will be problems in reliability and ease of maintenance and adaptability.

The **functionality** defect type accounted for 6.57% of all defects. The lack of shared vision between producer and consumer and the lack of experience with the application domain are the principal sources of risk contributing to the functionality defect type. Left unattended, this source of risk will continue at a low probability of occurrence, and there will be problems in end user satisfaction.

The **data** defect type accounted for 4.62% of all defects. The lack of experience with the application domain, the use of disciplined data structures, and the hardware/software platform are the principal sources of risk contributing to this defect type. Left unattended, this source of risk will continue at a low probability of occurrence, and there will be problems in reliability.

The **syntax** defect type accounted for 4.79% of all defects. The lack of experience with the rules of syntax for recording the artifact and the fact that people make mistakes sometimes are the principal sources of risk contributing to the syntax defect type. Left unattended, this source of risk will lead to ambiguity and errors in logical expression,

problems in end user satisfaction, and problems in end user satisfaction and adaptability.

The **maintainability** defect type accounted for 4.09% of all defects. The extreme emphasis on time to market and the widespread adoption of the code and ship/upload life cycle has impacted the use of good software engineering practice impacting the supportability and evolution of the software product. Left unattended, the occurrence of the maintainability defect type will increase, and there will be problems in end user satisfaction and the cost of maintenance.

The **performance** defect type accounted for 2.30% of all defects. The lack of experience with the application domain and the hardware/software platform is a principal source of risk contributing to the performance defect type. Left unattended, this source of risk which applies to a small percentage of the code will continue at a low probability of occurrence, and there will be problems in end user satisfaction as a result.

Questions Answered in the Lab

The micro-level national database on software product quality can be used to answer important software engineering questions. When appropriately selected core samples are accumulated in the Report Summary Form and the probability of occurrence is computed for each defect type, defect severity, and defect category, these probabilities can be used to construct answers to questions. Five of Boehm's top ten risks are answered below using the 1992-2002 data from the experiment:

To what extent were the wrong software functions being developed?
Functionality errors accounted for 6.57% of all errors.
To what extent were the wrong user interfaces developed?
Interface errors accounted for 1.18% of all errors.
Human Factors accounted for 1.98% of all errors.
To what extent was there gold plating?
9.20% of all errors were classified as extra.
To what extent was there a continuing stream of requirements changes?
Documentation errors accounted for 40.51% of all errors.
To what extent was there a shortfall in real time performance?
Performance errors accounted for 2.30% of all errors.

Questions Not Yet Answered

There is interest in defect leakage and ways to measure and reason about it. The Software Inspection Lab includes a mechanism to collect data on defect leakage and to reason about the results. This reasoning process crosses over into defect prevention... and fault and failure prediction.

Defect leakage was introduced into the National Software Quality Experiment in 1995, and the data on this is starting to build up. The defect leakage data needs to populate each analysis bin in sufficient quantity before these results are usable. With this data it will be possible to conduct special studies on defect leakage to augment the core analyses done continuously.

Questions asked but not yet answered include:

1. To what extent is defect leakage occurring?
2. What is the frequency distribution of defect types that leak?
3. What is the frequency distribution of defect types for each life cycle activity?

The mechanism used to gather defect leakage involves identifying the life cycle activity for each software inspection and the defect origin for each defect. Each software inspection is considered an exit criteria for a software product engineering activity. Each defect is characterized by category, severity, type, ... and defect origin. Defect origin is the software product engineering activity where the defect was inserted. Where defect origin does not match the software product engineering activity for which this inspection serves as an exit criteria, defect leakage has occurred.

Separating signal from noise is a challenge in the analysis of defect leakage. As I rollout software inspections in organizations, I know that the data collection mechanism is consistent and well defined and that the data collection practice is reasonably consistent. The inconsistency of the software product engineering activity is a source of noise. With the software industry practice at a low level of software process maturity, noise dominates signal. Learning more about measuring defect leakage and methods to reason about these measurements are prerequisites to serious work in the estimation of defect leakage.

ANALYSIS OF RESULTS

As software inspections practice takes hold on the project, measured results accumulate with each inspection session. Naturally the defects detected in each session are corrected and defect leakage is reduced. More can be done with the project's inspection measurement data.

These measurements provide the means to achieve defect prevention. This is accomplished by coordinating a monthly project analysis of these results. The project manager convenes a meeting of staff members dedicated to the analysis of software inspection results. This analysis spans both process and product measurements.

The process measurements and their derived metrics are assessed. The project measurements taken are compared to the project history, the organization measurement, and the industry average taken from the National Software Quality Experiment. Where the project measurement is at substantial variance, the project team reasons about the cause and the process changes it may suggest.

The product measurements are collected on the Inspection Summary Form which organizes the frequency of defect types by category and severity. Each defect type is assessed, and the following questions are asked:

1. Why does the defect type occur?
2. What management action can be taken?
3. What technical action can be taken?
4. What process change can be made?
5. Is the action worth taking?

At the conclusion of the session the project team recaps the actions worth taking. These analysis results may be shared with senior management and made available to the coordination infrastructure.

MEASUREMENT RESULTS BY ANALYSIS BINS

When organized into analysis bins, the information may suggest interesting trends. The analysis bins are used to organize the findings into collections of data that reveal distinctions. The types of bins selected are year, software process maturity, organization type, product type, programming language, global region, and industry type. As data for each year is

collected, the overall results become more interesting, and the population of analysis bins becomes more robust.

YEAR: 1992-2002

The thesis for the experiment was stated in 1992, the year the Department of Defense made a commitment to reduce software problems by a factor of ten by the year 2000. Accordingly, the experiment systematically tracks software inspection measurements and metrics each year beginning with 1992.

The data collected in the Software Inspection Lab are well defined measurements. Ralph Waldo Emerson observed, "The years teach us things the days never knew." In fact this adage holds true for the experiment.

SOFTWARE PROCESS MATURITY

The Software Engineering Institute's Capability Maturity Model identifies five levels of software process maturity. These levels motivate and guide an organization from commitment management to the achievement of predictable results through measurement. Accordingly, the experiment organizes the measurements and metrics by software process maturity. The organizations participating in the experiment were assessed at level 1, level 2, and level 3 during their involvement .

ORGANIZATION TYPE

There is interest in the effect that organization type has on on a variety of software engineering outcomes including quality. A useful distinction among organizations is to separate Government, DOD Industry, and Commercial organizations. A Government organization draws its performance team from the ranks of civil servants and military personnel. A DOD Industry organization produces software under contract to the Government or other DOD Industry organization. A commercial organization produces software to support its business enterprise or to sell to consumers.

PRODUCT TYPE

There is interest in the effect that product type has on a variety of software engineering outcomes including quality. Some software products are embedded in complex hardware/software systems and may operate in real time. These tend to operate within tight constraints and place a high premium on meeting schedule. Other software products are organically entwined with the people and processes of the enterprise they serve. These are produced by relatively small teams that possess a thorough understanding of how the system contributes to the organization's objectives.

- Embedded software might be found in the guidance system for a cruise missle or a collision avoidance system on a railroad engine. This operates in real time and must be safety critical. Consequently, embedded software is produced using disciplined software engineering. These organizations may operate as SEI CMM level 4-5.
- Organic software might be found in a business supporting accounts receivable, invoicing, and payments. This operates in quick time and must be trustworthy. Consequently, organic software is produced using structured software engineering. These organizations may operate as SEI CMM level 3.
- In my experience, eCommerce applications, which are driven by time to market demands, are produced using ad hoc programming best described as code and upload.

PROGRAMMING LANGUAGE

There is interest in the effect programming language selection has on a variety of software engineering outcomes including quality. To construct programming language analysis bins with sufficient samples and sample sources, it was necessary to group programming languages into old style and modern. The old style bin includes Cobol, Fortran, CMS-2, Jovial and assembly language. The modern bin includes Ada, C, C++, Java, HTML, Perl, Mumps, SQL, and Visual Basic.

GLOBAL REGION

As global competitiveness moves to the forefront, understanding distinctions among global regions is of interest. Analysis bins exist for North America, Latin America, and Asia Pacific.

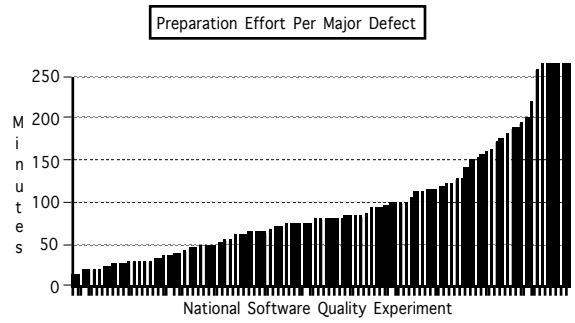
INDUSTRY TYPE

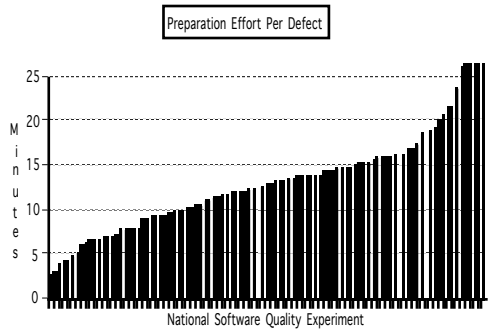
As the usage of software increases, the number of industries dependent on software increases. Essential industries which derive significant value from software include telecommunications, transportation, financial, manufacturing, medical systems, utilities and energy, defense, and e-commerce. Note: e-commerce results are reported in the industry group of which they are a part.

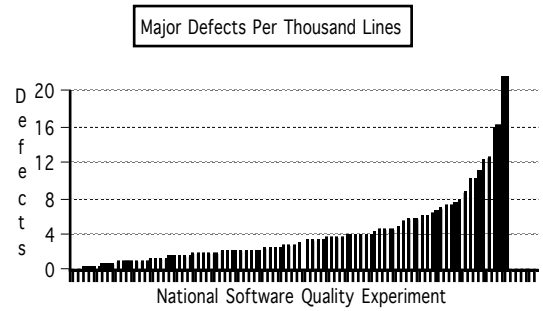
SOFTWARE PRODUCT ENGINEERING MODE

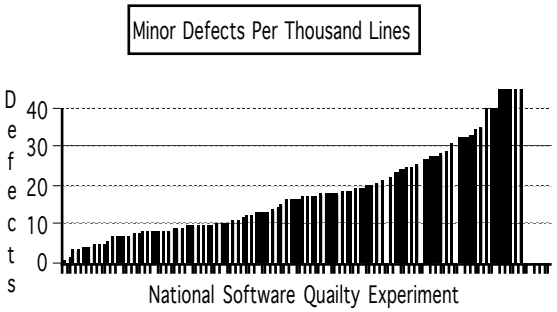
Three modes of software product engineering practice are identified:

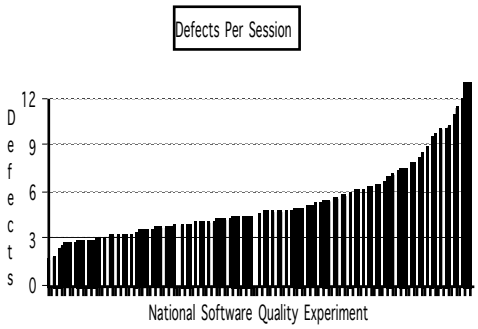
1. Ad hoc programming is characterized by a code and upload life cycle and a hacker coding style. The result is spaghetti bowl coding lacking in order and consistency. This is common in low software process maturity organizations especially those facing time to market demands. Its practitioners are expected to experience high defect insertion and low defect detection rates.
2. Structured software engineering employs structured programming, modular design, and defined programming style and pays close attention to establishing and maintaining traceability among requirements, specification, architecture, design, code, and test artifacts. The result is well structured, consistently recorded components with organized relationships among modules and traceability among life cycle artifacts.
3. Disciplined software engineering is more formal and might be patterned after Clean Room software engineering [Prowell 99], Personal and Team Software Process [Humphrey 97], and Extreme Programming techniques [Wells 01]. The result is well specified, professionally engineered, expertly architected with source code components organized and made understandable through templates for repeating patterns whose completeness, correctness, style, and rules of construction can be reasoned about with confidence.

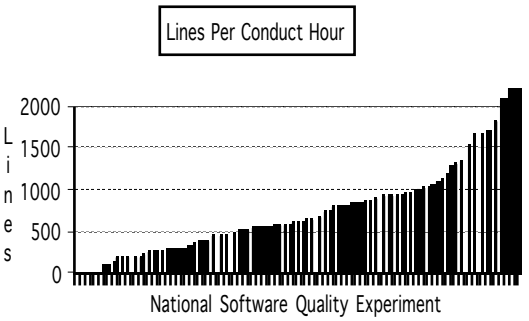

Preparation Effort Per Major Defect
Minutes
250
200
150
100
50
0
National Software Quality Experiment

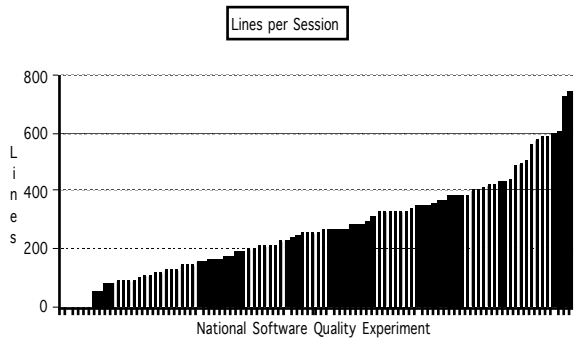

Preparation Effort Per Defect
Minutes
25
20
15
10
5
0
National Software Quality Experiment

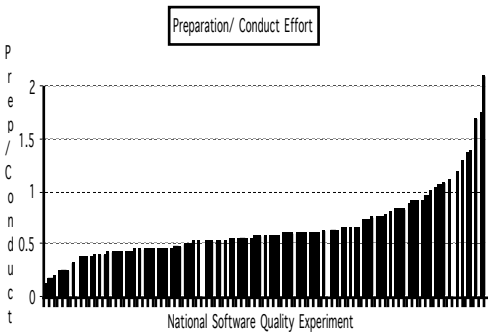

Major Defects Per Thousand Lines
Defects
20
16
12
8
4
0
National Software Quality Experiment

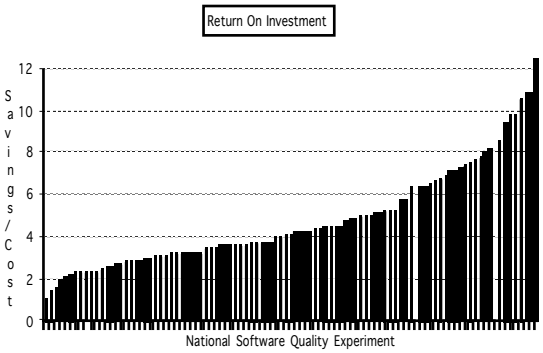

Minor Defects Per Thousand Lines
Defects
40
30
20
10
0
National Software Quailty Experiment

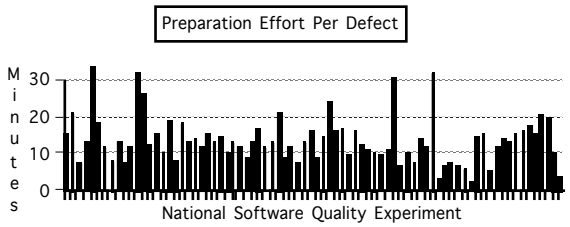

Defects Per Session
Defects
12
9
6
3
0
National Software Quality Experiment

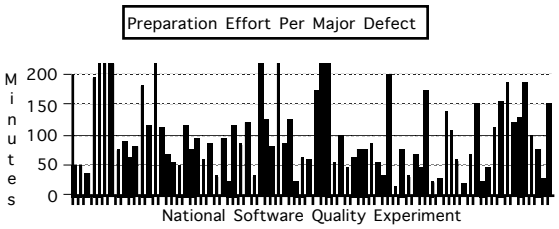

Lines Per Conduct Hour
Lines
2000
1500
1000
500
0
National Software Quality Experiment

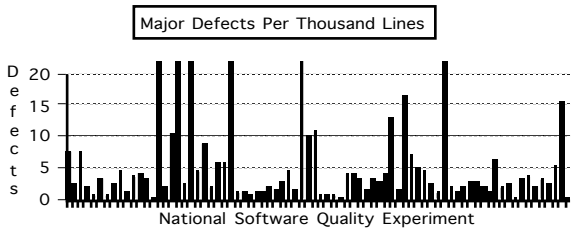

Lines per Session
Lines
800
600
400
200
0
National Software Quality Experiment

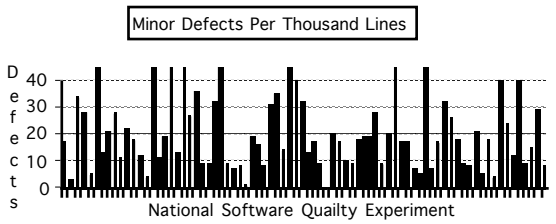

Preparation/ Conduct Effort
Prep/Conduct
2
1.5
1
0.5
0
National Software Quality Experiment

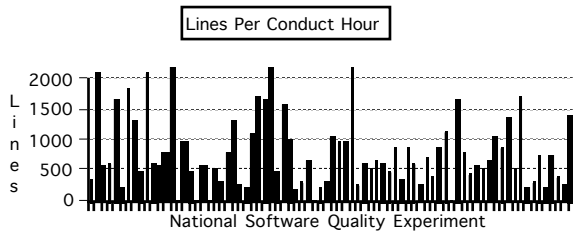

Return On Investment
Savings/Cost
12
10
8
6
4
2
0
National Software Quality Experiment

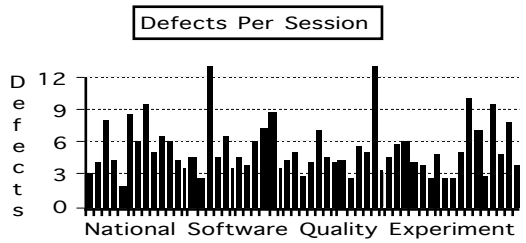

Preparation Effort Per Defect
Minutes
30
20
10
0
National Software Quality Experiment

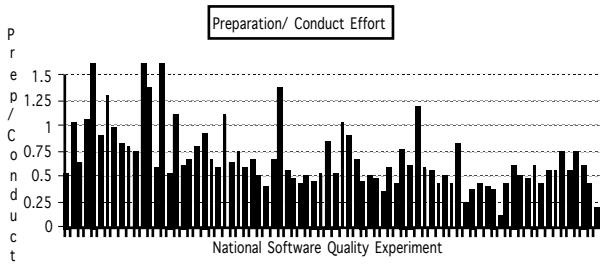

Preparation Effort Per Major Defect
Minutes
200
150
100
50
0
National Software Quality Experiment

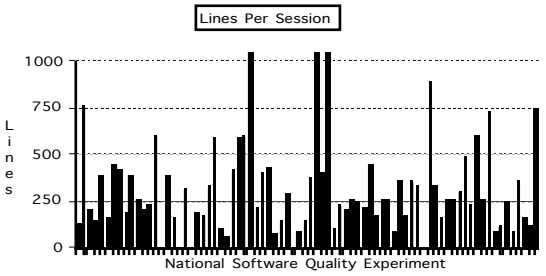

Major Defects Per Thousand Lines
Defects
20
15
10
5
0
National Software Quality Experiment

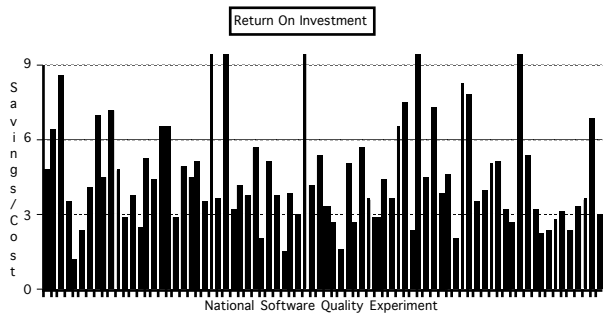

Minor Defects Per Thousand Lines
Defects
40
30
20
10
0
National Software Quailty Experiment

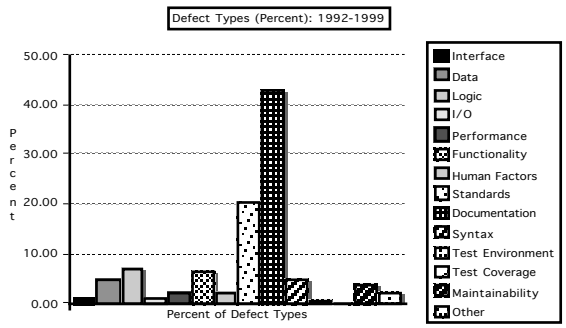

Lines Per Conduct Hour
Lines
2000
1500
1000
500
0
National Software Quality Experiment


Defects Per Session
Defects
12
9
6
3
0
National Software Quality Experiment


Preparation/ Conduct Effort
Prep/Conduct
1.5
1.25
1
0.75
0.5
0.25
0
National Software Quality Experiment


Lines Per Session
Lines
1000
750
500
250
0
National Software Quality Experiment

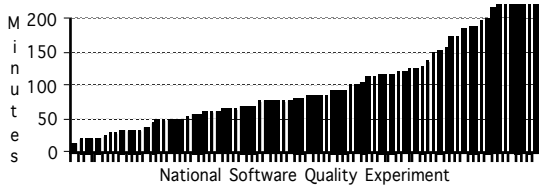

Return On Investment
Savings/Cost
9
6
3
0
National Software Quality Experiment


Defect Types (Percent): 1992-1999
Percent
50.00
40.00
30.00
20.00
10.00
0.00
Percent of Defect Types
Interface
Data
Logic
I/O
Performance
Functionality
Human Factors
Standards
Documentation
Syntax
Test Environment
Test Coverage
Maintainability
Other

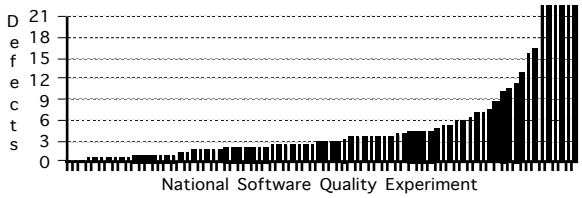
National Software Quality Experiment Metrics: All Participants by Time

Preparation Effort Per Defect

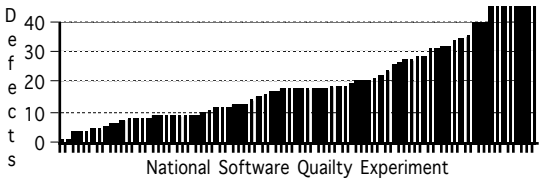
Preparation Effort Per Major Defect



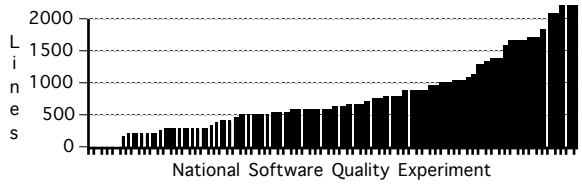
Major Defects Per Thousand Lines



Minor Defects Per Thousand Lines

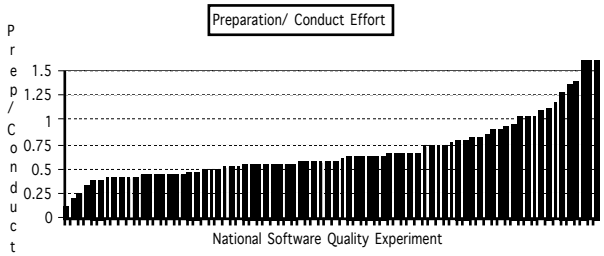


Lines Per Conduct Hour

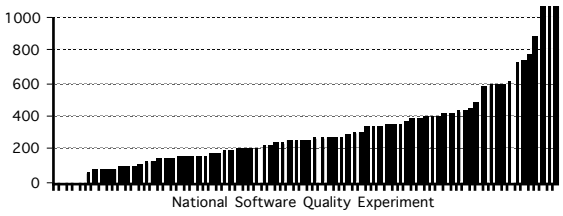


Defects Per Session

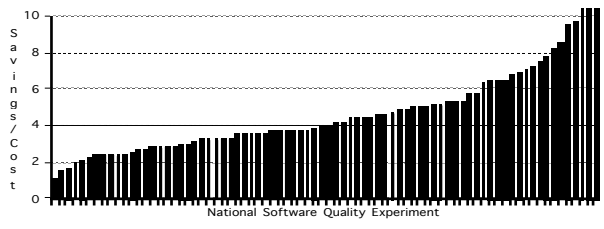
Preparation/ Conduct Effort



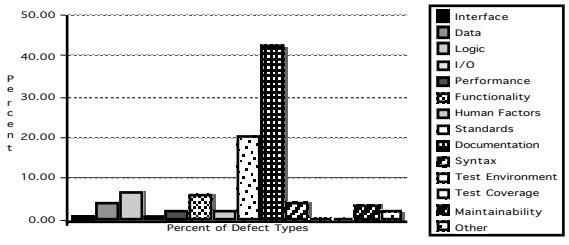
Lines per Session



Return On Investment

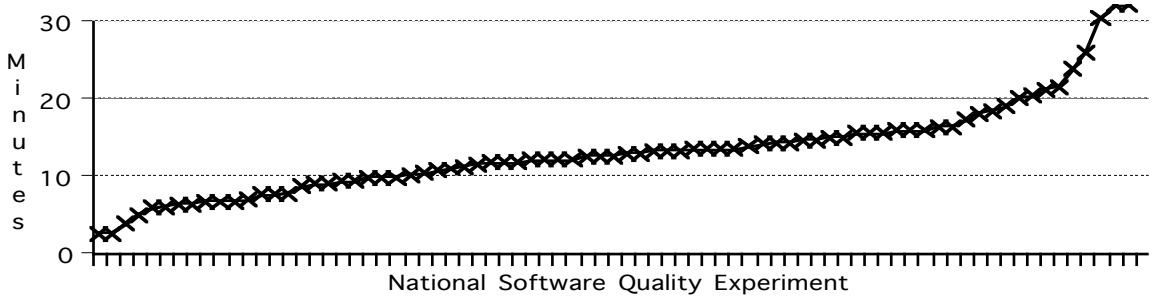


Defect Types (Percent): 1992-1999

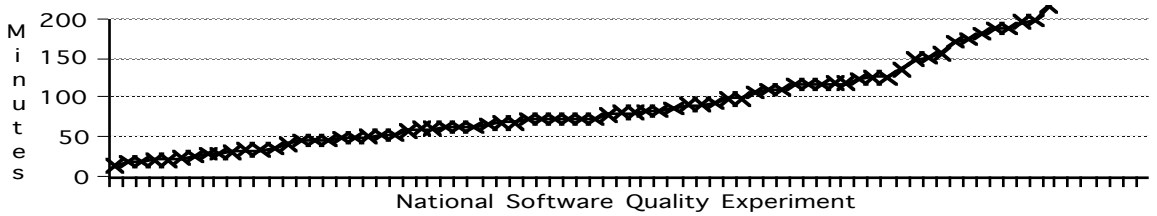


National Software Quality Experiment Metrics: All Participants Sorted

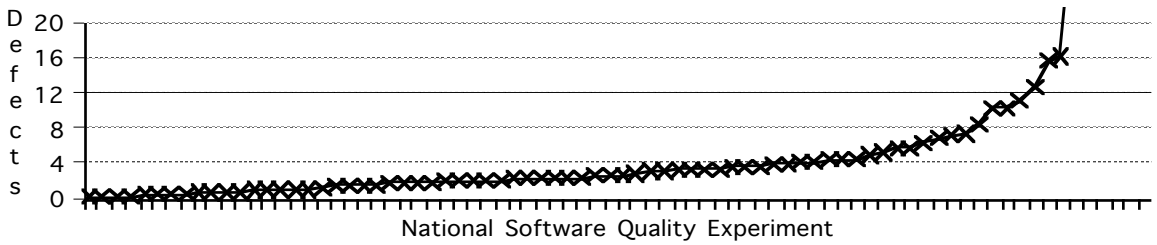
Preparation Effort Per Defect

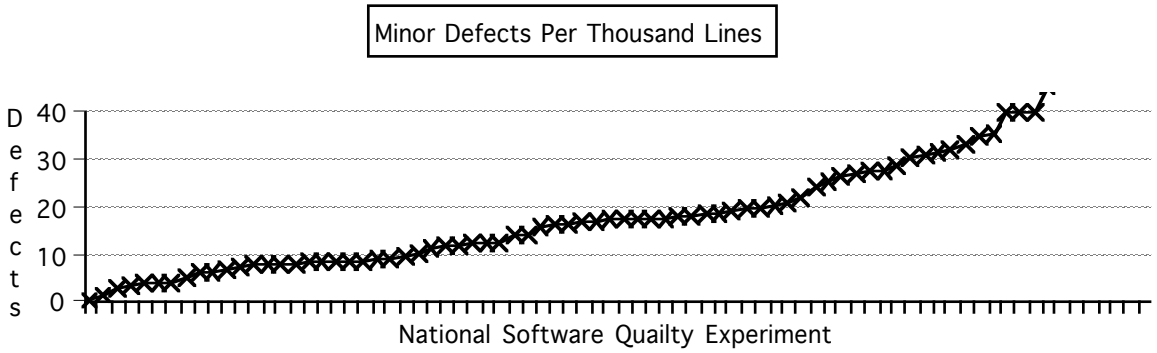


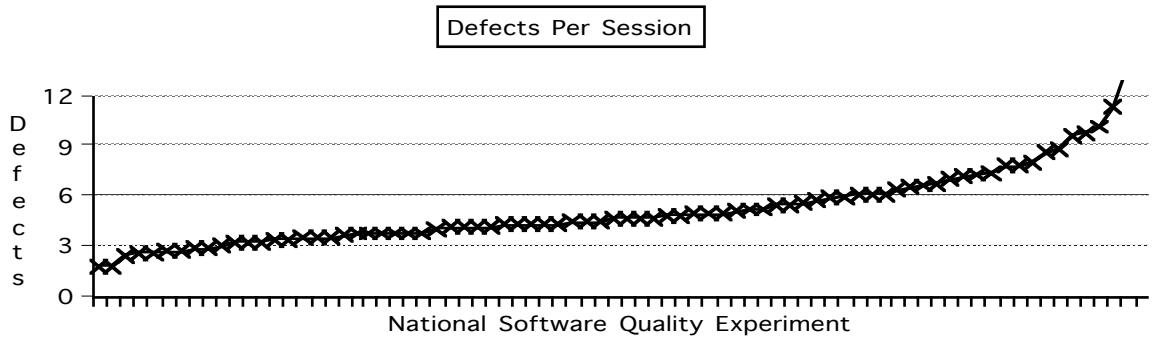
Preparation Effort Per Major Defect

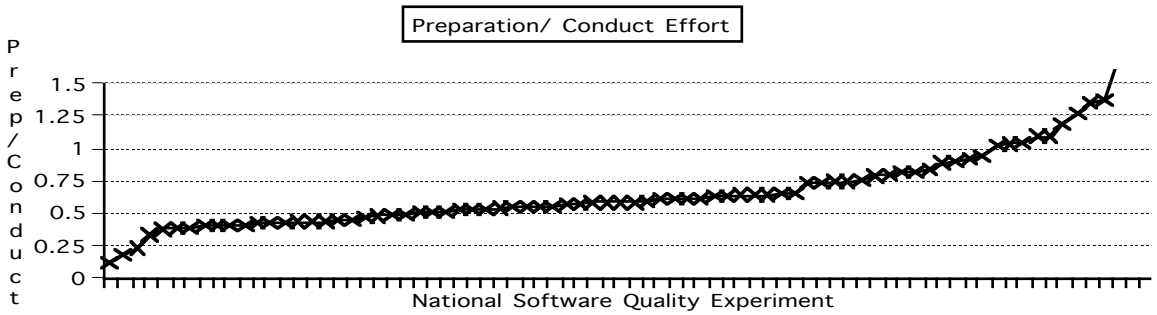


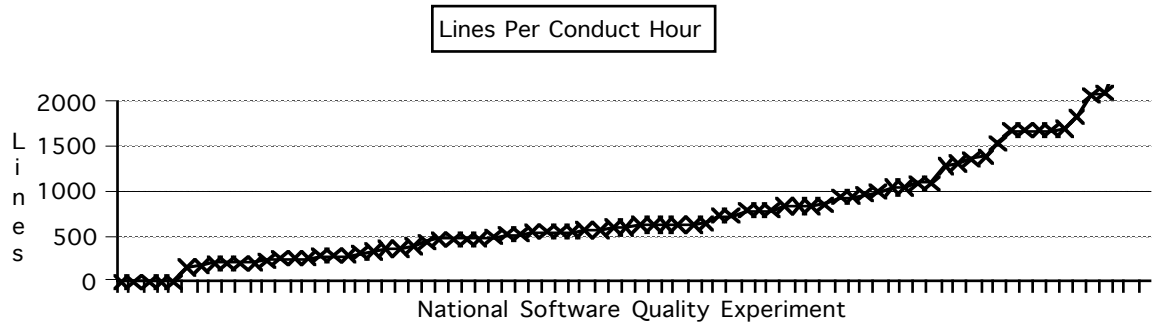
Major Defects Per Thousand Lines

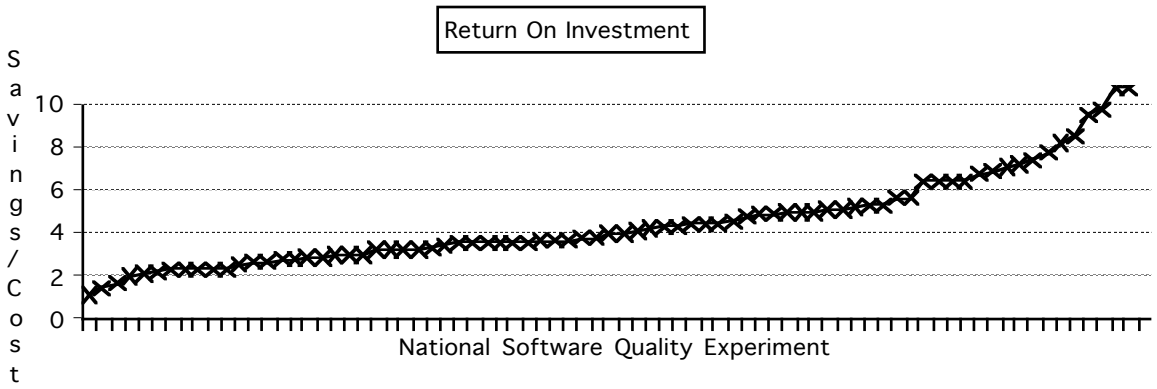



Minor Defects Per Thousand Lines
Defects
40
30
20
10
0
National Software Quailty Experiment

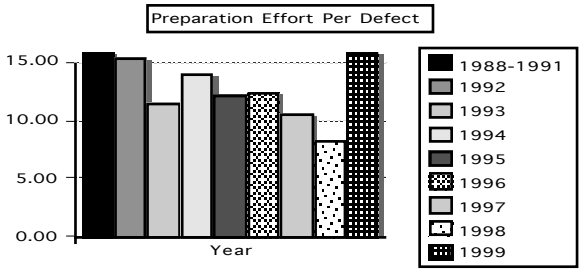

Defects Per Session
Defects
12
9
6
3
0
National Software Quality Experiment

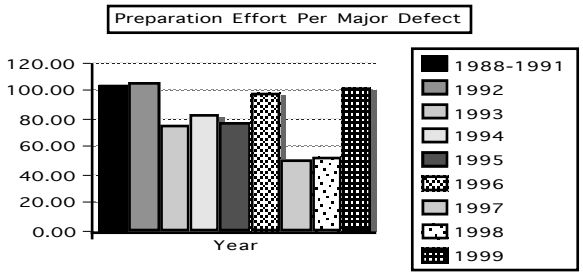

Preparation/ Conduct Effort
Prep/Conduct
1.5
1.25
1
0.75
0.5
0.25
0
National Software Quality Experiment

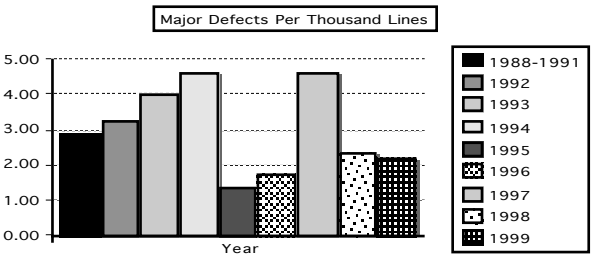

Lines Per Conduct Hour
Lines
2000
1500
1000
500
0
National Software Quality Experiment

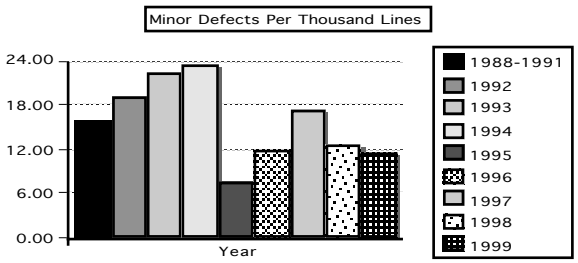

Return On Investment
Savings/Cost
10
8
6
4
2
0
National Software Quality Experiment

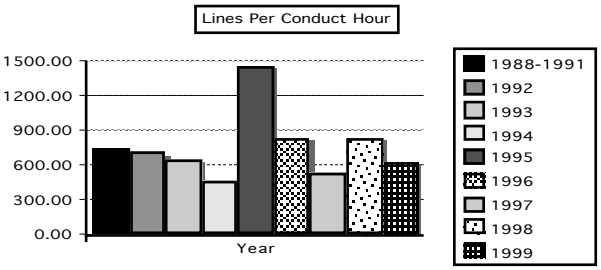
National Software Quality Experiment Metrics: Annual Results

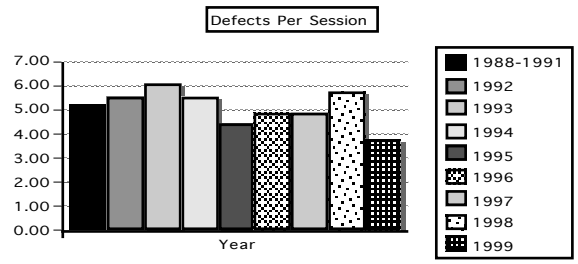


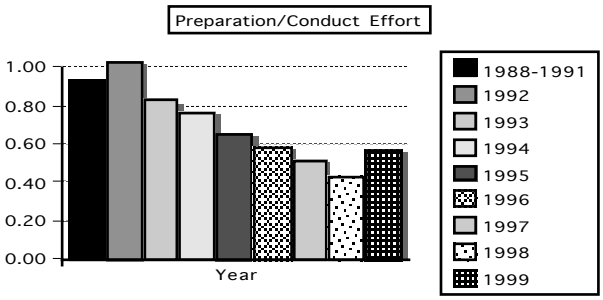


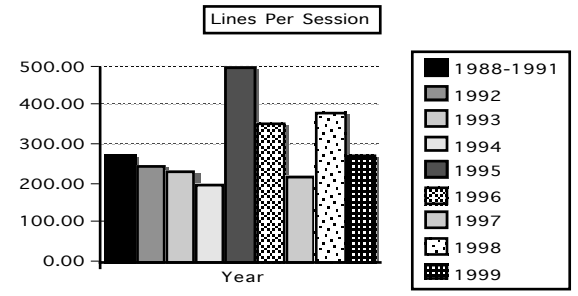


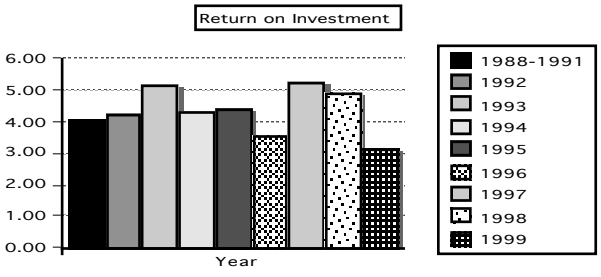


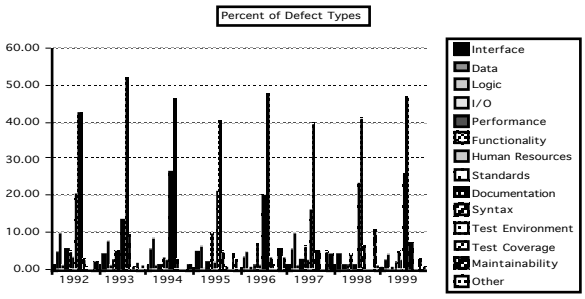




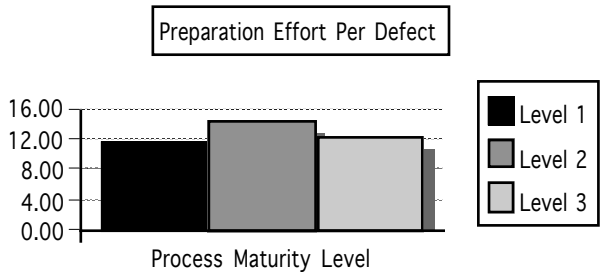


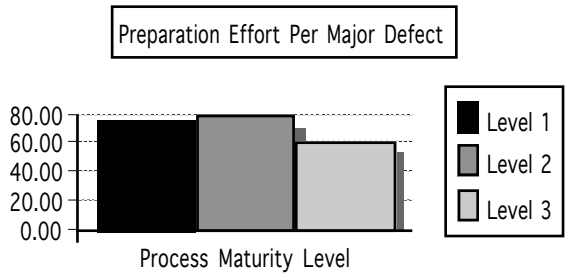


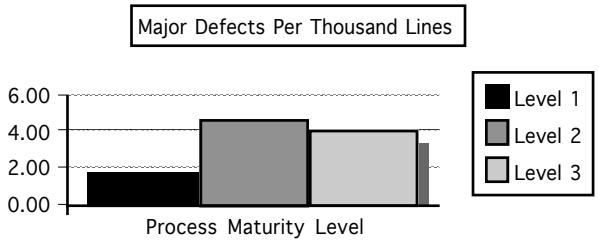


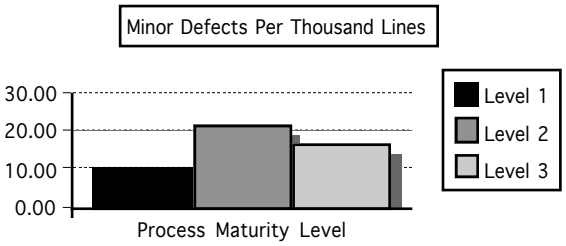


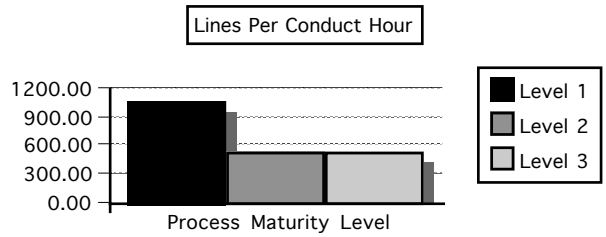
National Software Quality Experiment Metrics: Software Process Maturity

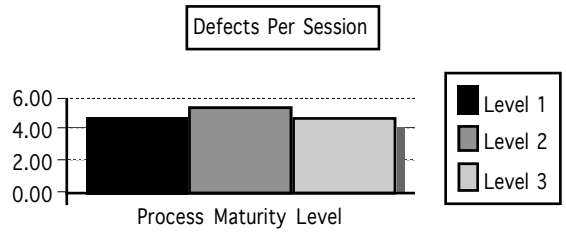


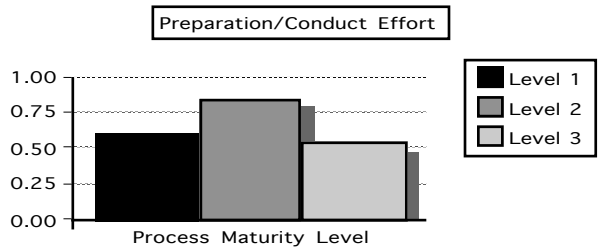


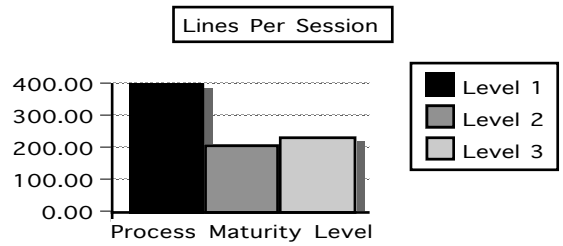


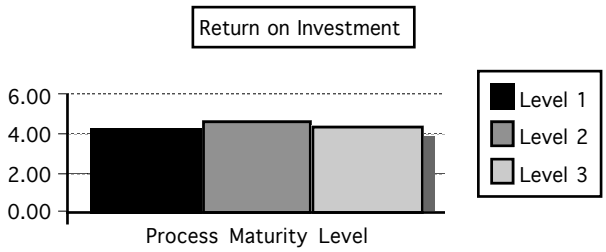


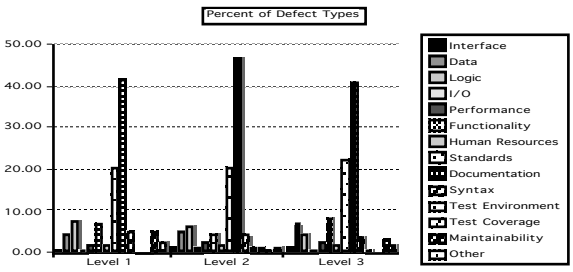












Analysis of Annual Averages
The 1992 hypothesis being investigated in the National Software Quality Experiment is whether software problems are being reduced by a factor of ten by the year 2000. An analysis of annual averages suggests that a moderately stable process is in operation, and that there is little pressure to reestablish performance at an improved level of practice.

Prep/Defect- The average prep/defect has operated in a stable manner in the early years with less predictability in the later years. A lower prep/defect may suggest:
1. There may be more inserted defects to find.
2. Practitioners may be getting better at detecting defects.

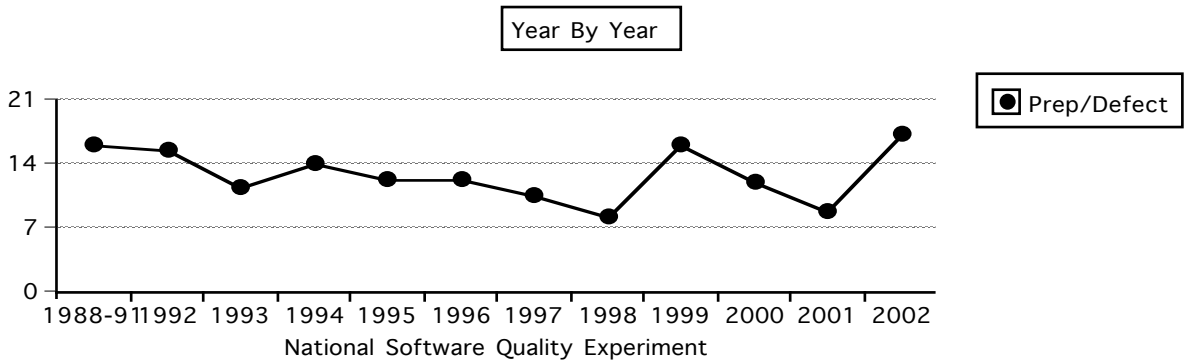


Figure 6a Preparation Per Defect by Year

Prep/Maj- The average prep/maj has operated in a stable manner almost throughout the period. A lower prep/defect may suggest:
1. Practitioners may be getting better at detecting defects... especially major defects.
2. Practitioners may be inserting more defects.

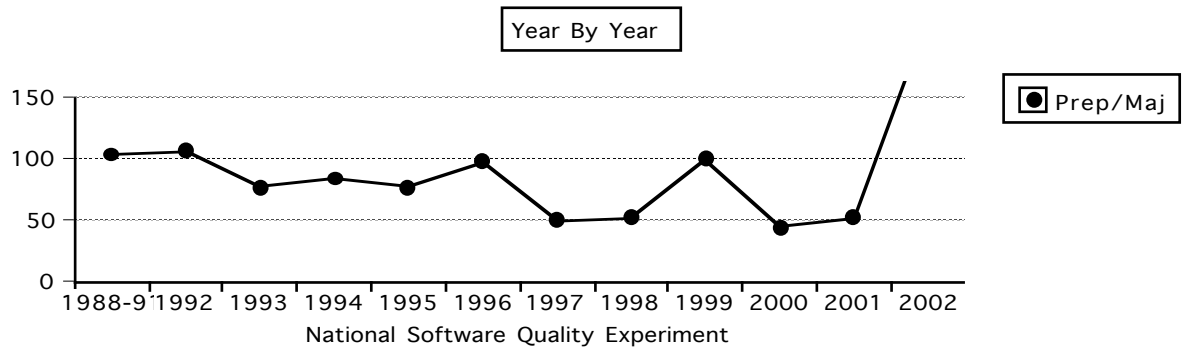


Figure 6b Preparation Per Major Defect by Year

Maj/Thousand- The average maj/thousand has operated in a moderately unstable manner throughout the time period.
1. This metric is sensitive to the distinction between new development code and legacy code.
2. This metric is sensitive to the defect insertion rate. Therefore, a high average maj/thousand may simply indicate the presence of a large number of defects.

3. This metric is sensitive to software product engineering mode which influences both detect insertion and detect detection rate.

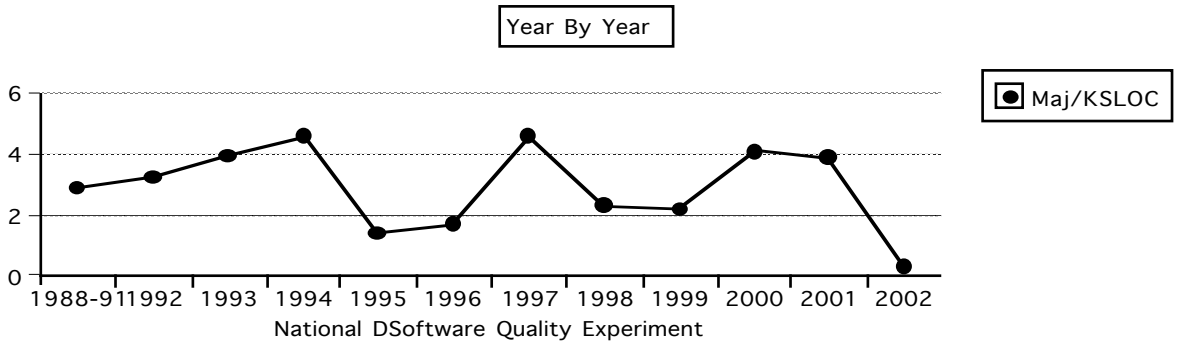


Figure 6c Major Defects Per Thousand Lines by Year

Min/Thousand- The average min/thousand has operated in a moderately unstable manner throughout the time period.
1. Practitioners seem to detect a steady volume of minor defects.
2. This metric is sensitive to the distinction between new development code and legacy code.
3. This metric is sensitive to the defect insertion rate. Therefore, a high average min/thousand may simply indicate the presence of a large number of defects.
4. This metric is sensitive to software product engineering mode which influences both detect insertion and detect detection rate.

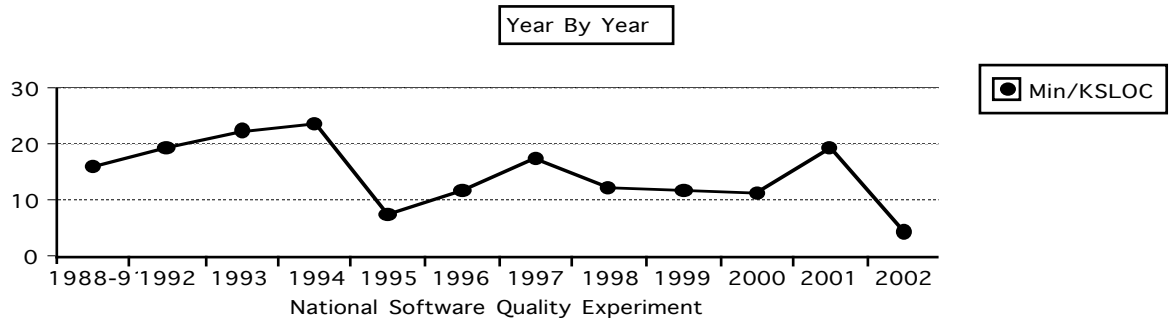


Figure 6d Minor Defects Per Thousand Lines by Year

Size/Conduct Hour- Average size/conduct hour has operated in a moderately stable manner throughout most of the time period.
1. This metric is sensitive to the distinctions between new development code and legacy code.
2. This metric is sensitive to the preparation effort prior to the session. Higher preparation effort may yield higher size/conduct hour.

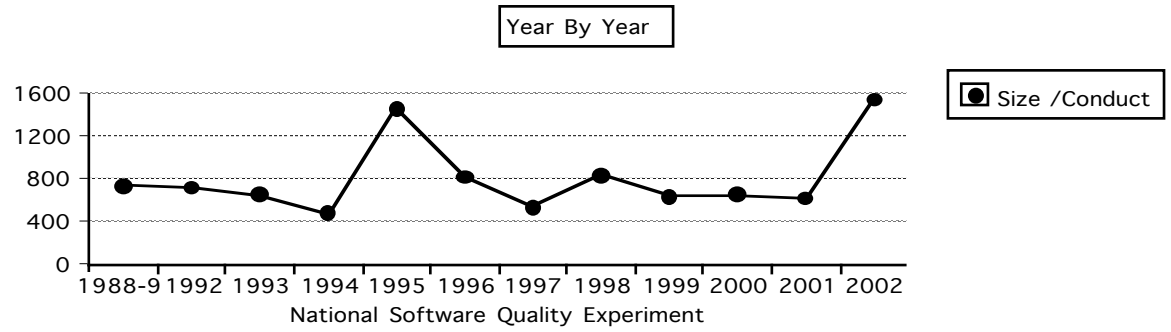


Figure 6e Size Per Conduct Hour by Year

Defects/Session- Average defects /session operated in a moderately stable manner throughout most of the time period.
1. Practitioners seem to detect a steady number of defects per session.
2. This metric is sensitive to the preparation effort prior to the session. Higher preparation effort may yield higher defects/session.

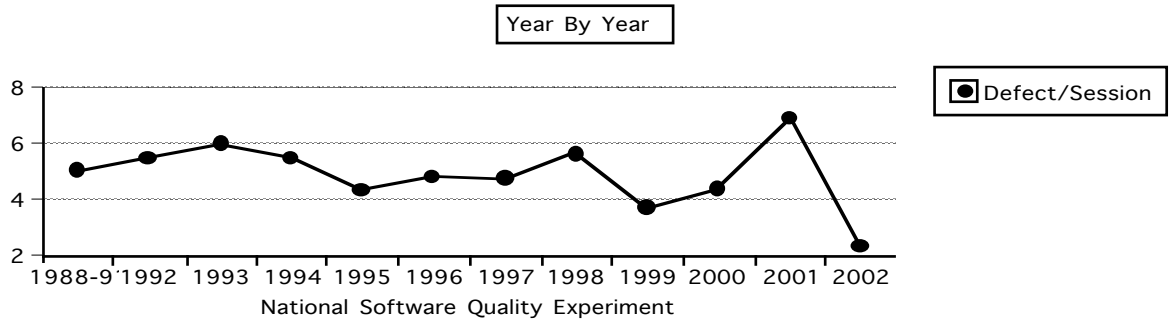


Figure 6f Defects Per Session by Year

Prep/Conduct- Average prep/conduct has consistently trended downward over the time period.
1. Practitioners increasingly are experiencing excessive overtime and even off the clock time and neglect preparation effort for software inspections.
2. Where average prep/conduct approaches equilibrium (1.0), defect leakage may be reduced.

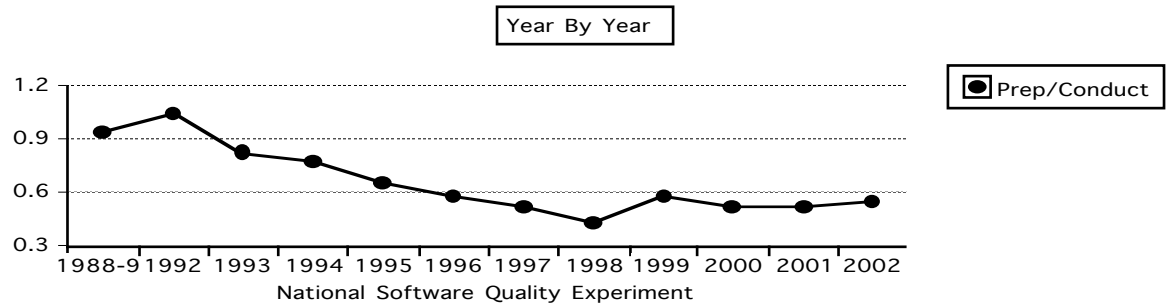


Figure 6g Preparation Per Conduct Effort by Year

Size/Session- Size/session has operated in a moderately stable manner throughout the time period.
1. One way to look harder for defects is to reduce size/session.
2. This metric is sensitive to the distinctions between new development code and legacy code.
3. This metric is sensitive to the preparation effort prior to the session.

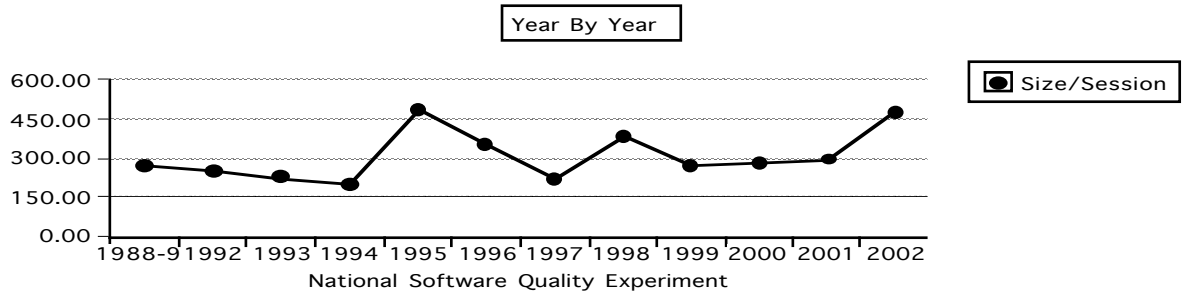


Figure 6h Size Per Session by Year

Savings/Cost (ROI)- Average savings/cost has operated in a marginally stable manner throughout the time period.
1. Return on investment fuels management commitment to the software process.
2. High defect detection results in high return on investment.

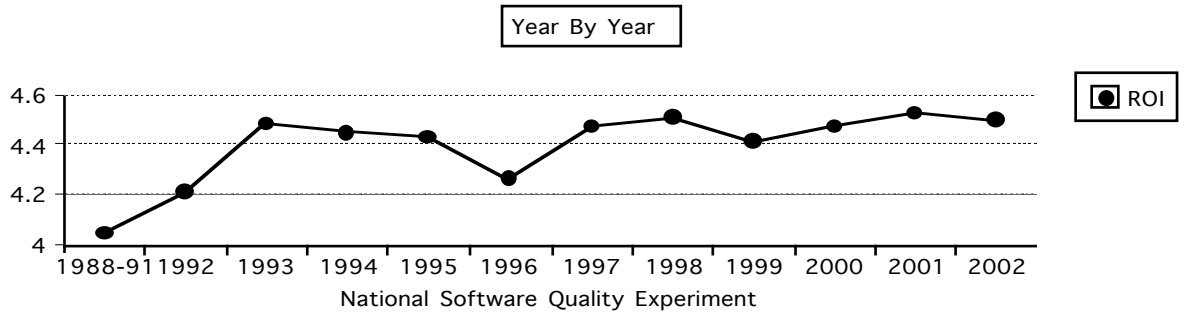
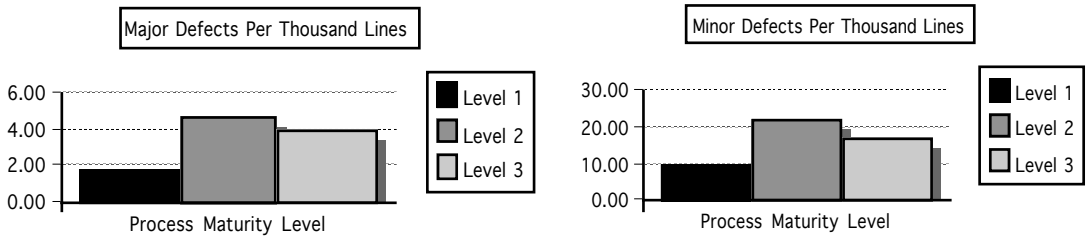


Figure 6i Return on Investment by Year

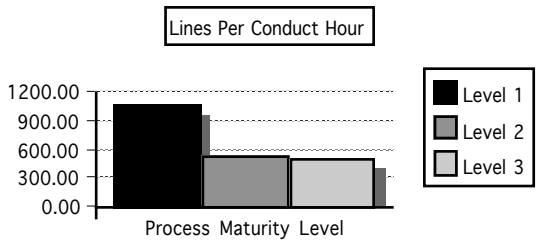
Reasoning About Findings: Software Process Maturity

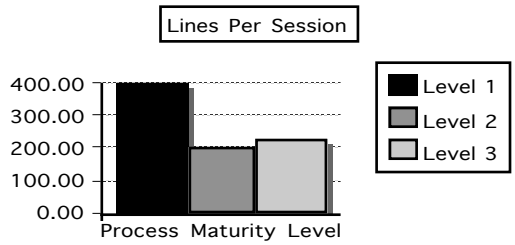
The level 1 major and minor defects per thousand lines are less than half of level 2 and level 3. Are level 1 organizations inserting less defects ... or simply finding less?



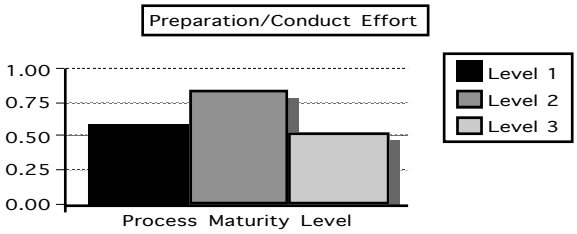
Level 1 lines per conduct hour and lines per session are double level 2 and level 3. One way to look harder for defects is to inspect smaller artifacts. Level 2 and level 3

organizations are looking harder... and finding more.

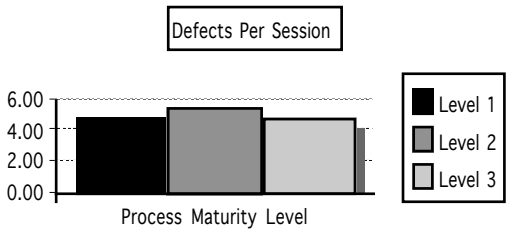


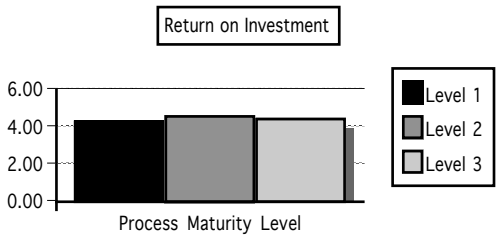


Level 2 preparation/conduct effort approaches 1.0 which is the desired equilibrium. Another way to look harder for defects is to increase preparation time. Level 1 and level 3 show a large shortfall in preparation... suggesting that more defects could be detected.



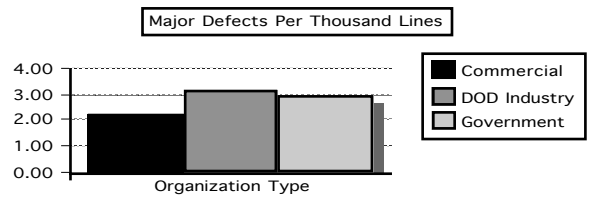
Level 1, 2, 3 defects per session and return on investment is consistently repeatable.

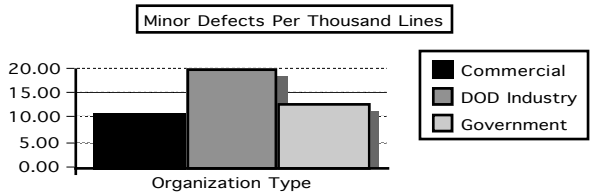




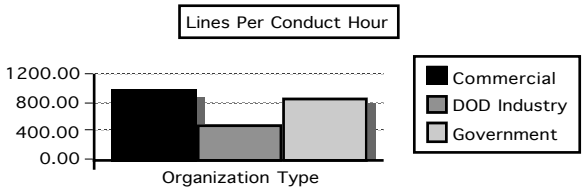
Reasoning About Findings: Organization Type

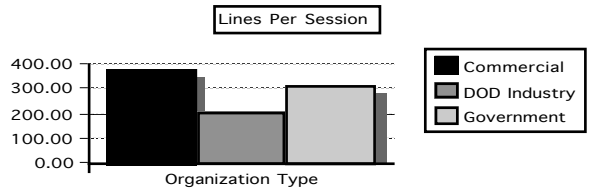
DOD Industry seems to find more defects. Interestingly DOD Industry detects substantially more minor defects. While minor defects do not effect execution, their detection and correction is beneficial for maintenance.



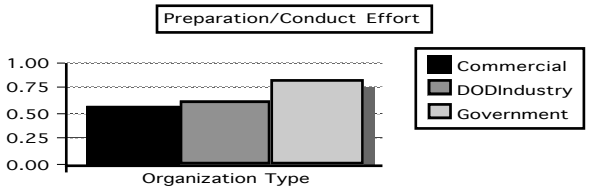


Again DOD Industry stands out because it is inspection smaller artifacts. One way to look harder for defects is to inspect smaller artifacts.

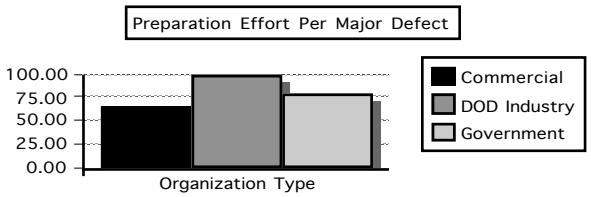


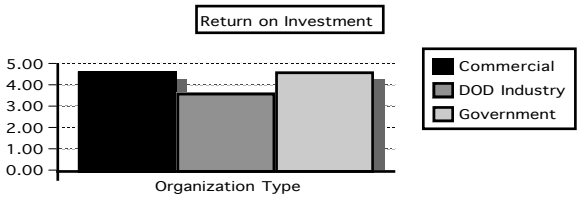


The Government preparation/conduct effort approaches 1.0 which is the desired equilibrium. Another way to look harder for defects is to increase preparation time.

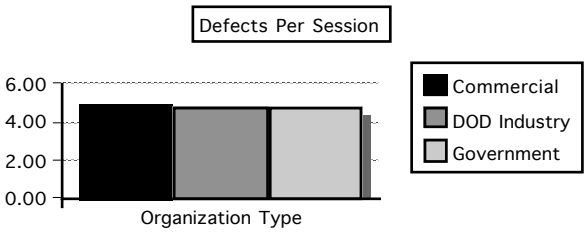


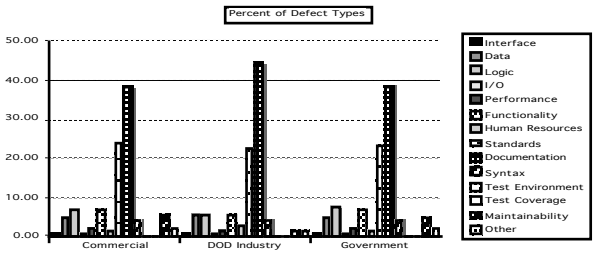
The DOD Industry return on investment lags others because its preparation per major defect is higher, and major defects strongly influence return on investment. Recall that DOD Industry looked harder by inspecting smaller artifacts.





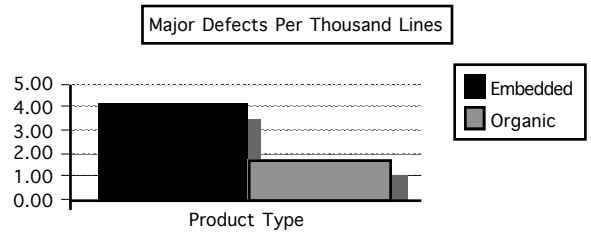
The defects per session and defect type distributions are consistently repeatable.

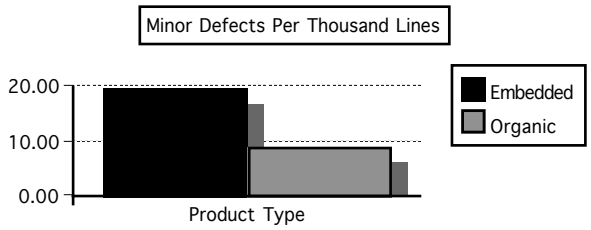




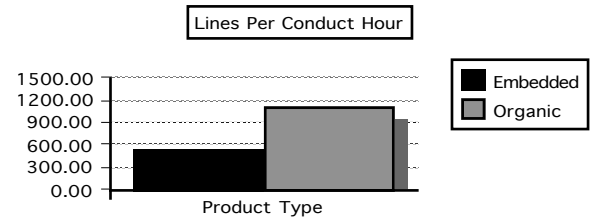
Reasoning About Findings: Product Type

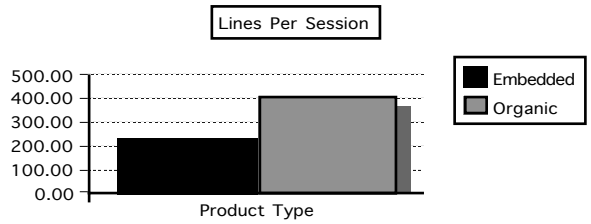
Embedded systems defect detection are more than twice of organic systems.



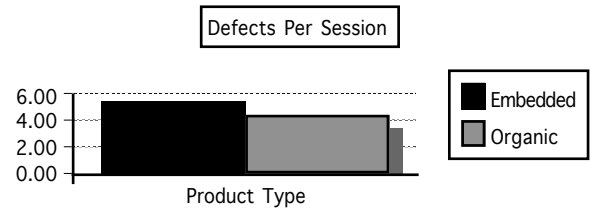


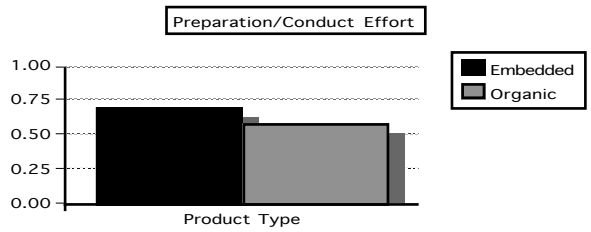
Embedded systems is inspecting smaller artifacts. One way to look harder is to inspect smaller artifacts.



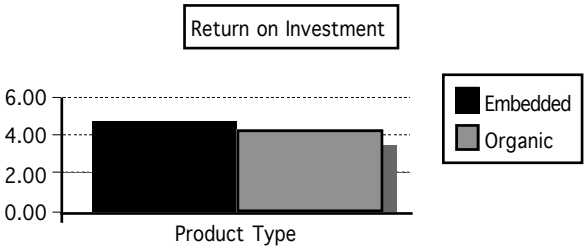


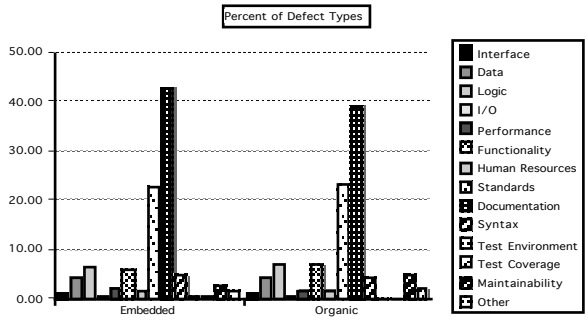
Embedded systems are slightly higher than organic systems in defects per session and preparation/conduct effort.



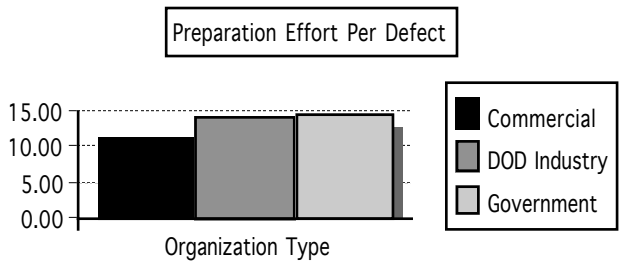


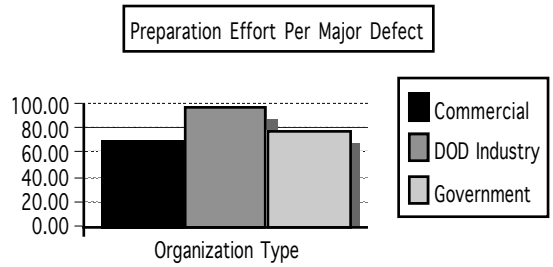
The return on investment and defect type distributions are consistently repeatable.

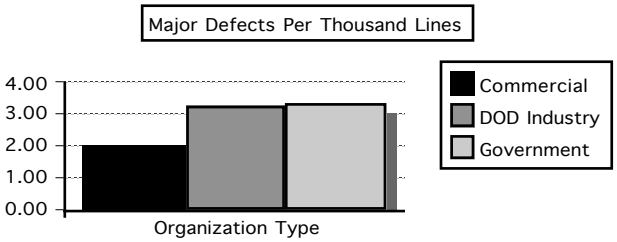


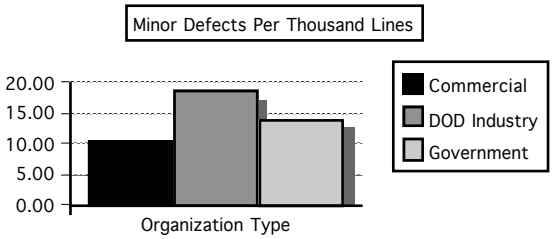


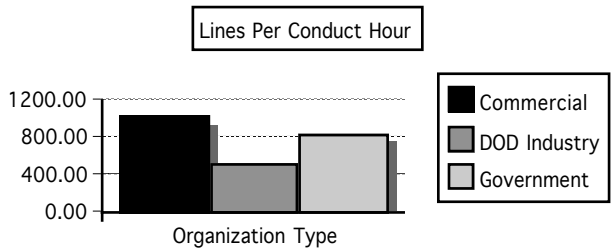
National Software Quality Experiment Metrics: Product Type

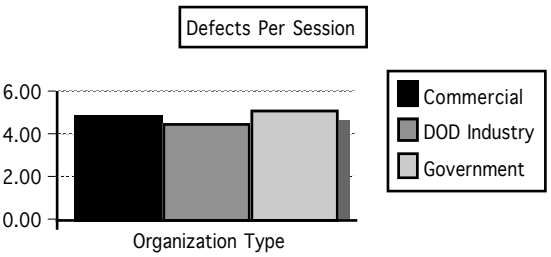


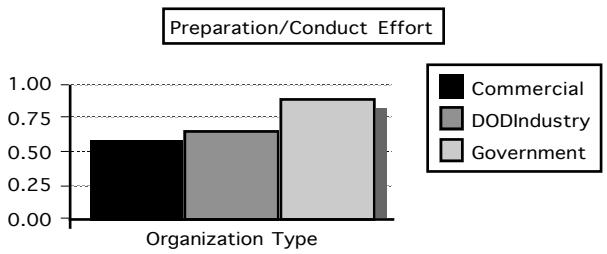


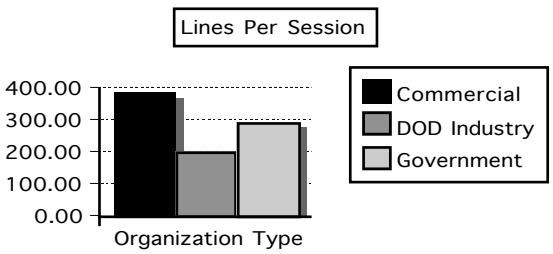


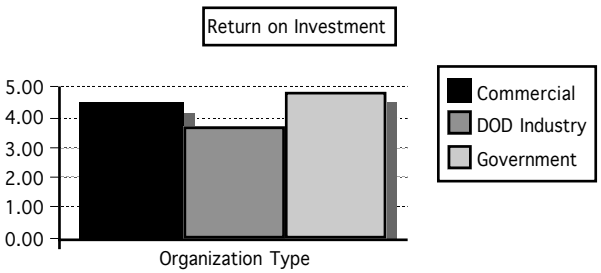


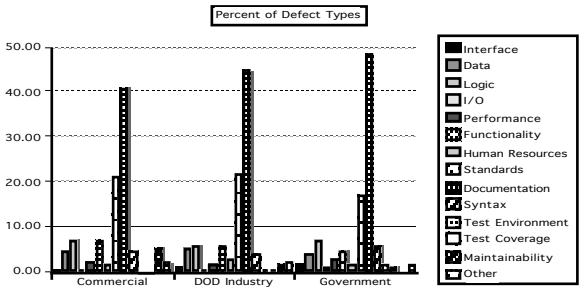




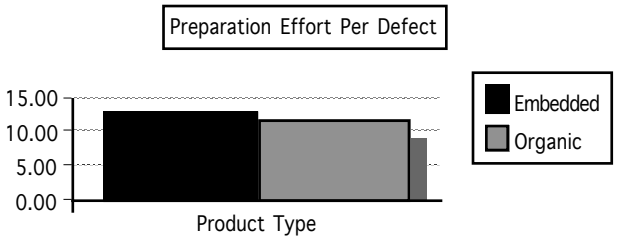


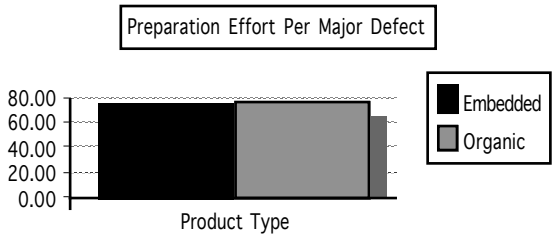


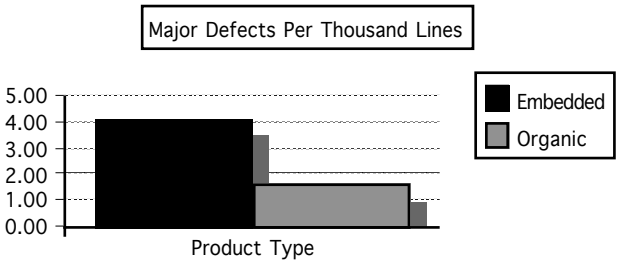


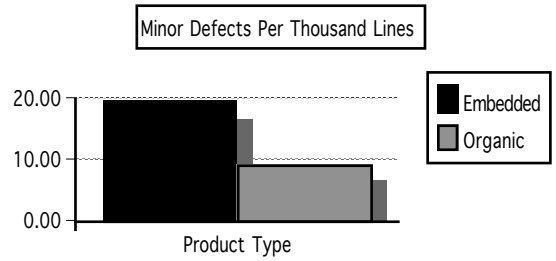


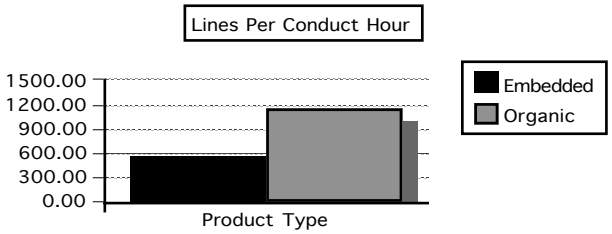
National Software Quality Experiment Metrics: Organization Type

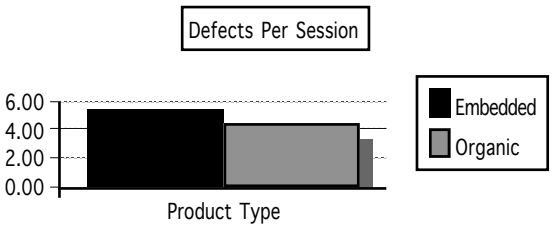


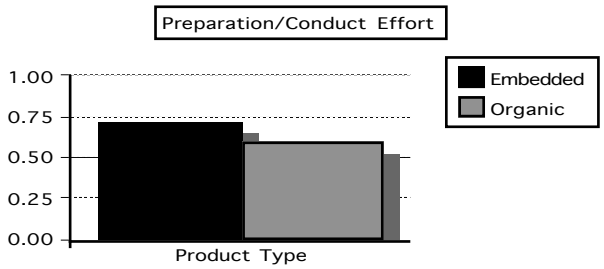


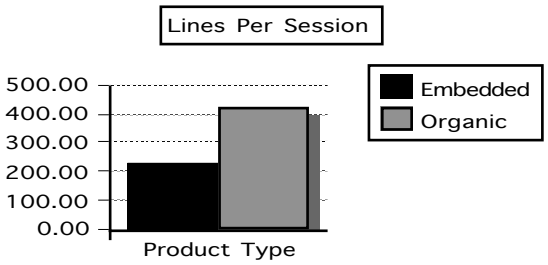


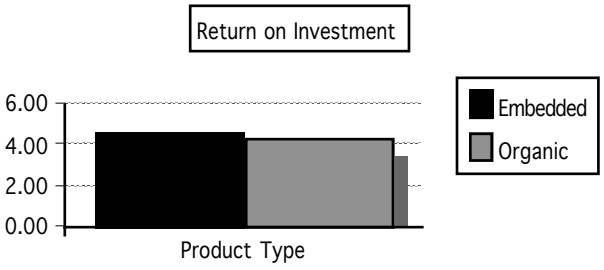


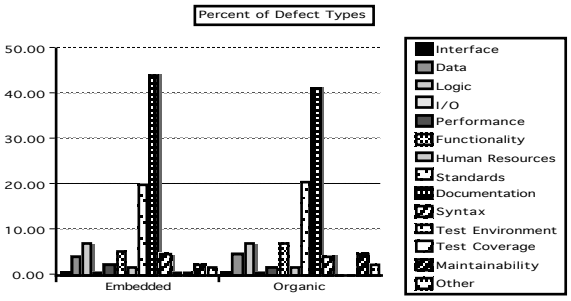




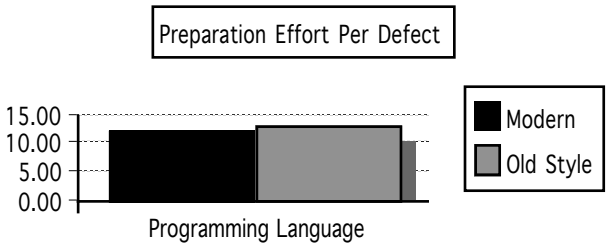


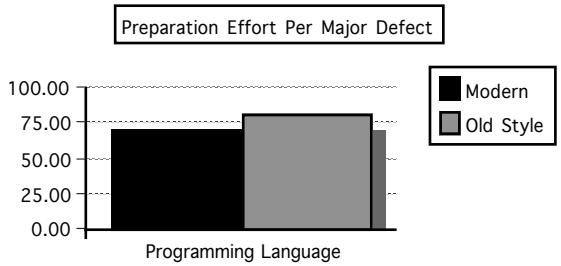


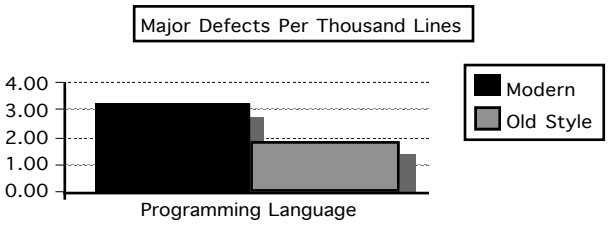


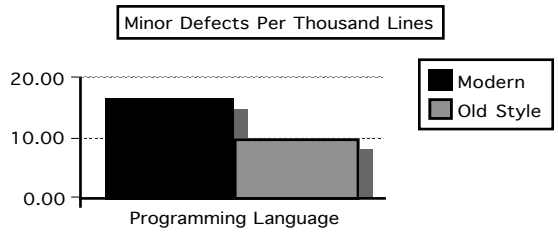


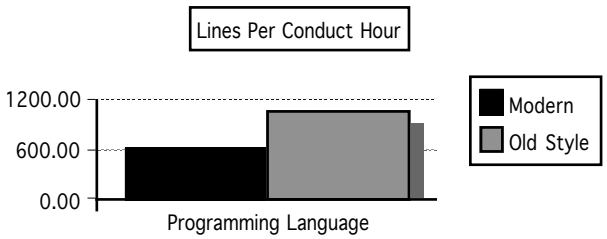
National Software Quality Experiment Metrics: Programming Language

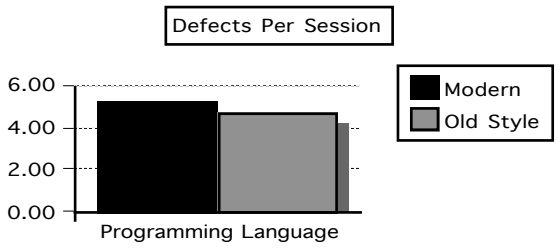


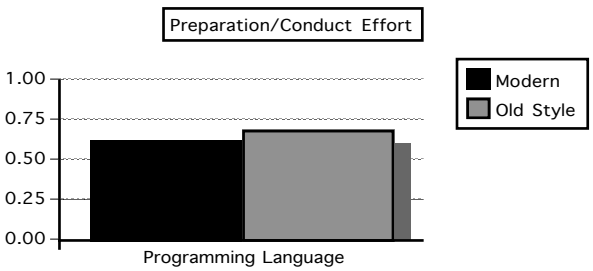


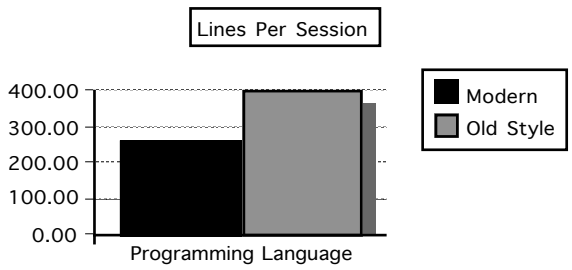


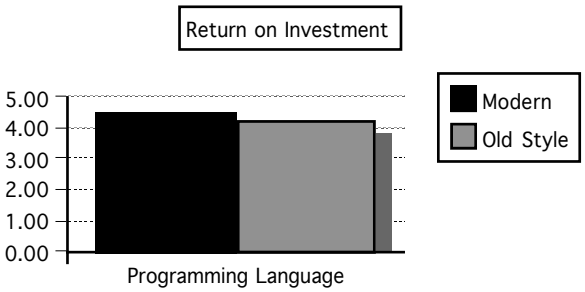


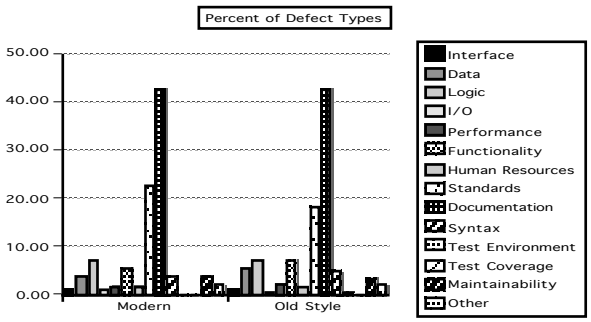




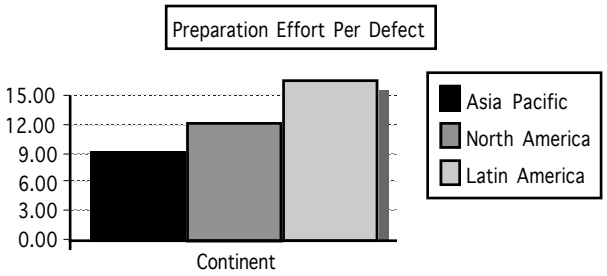


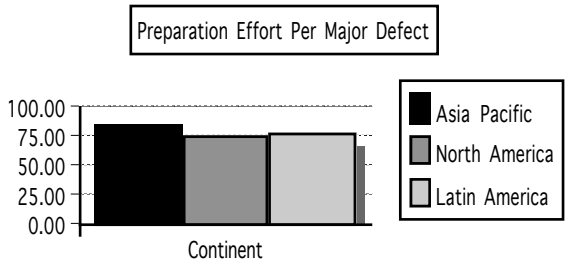


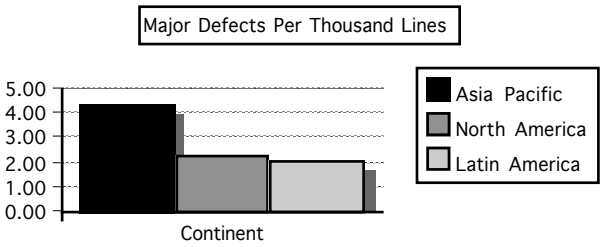


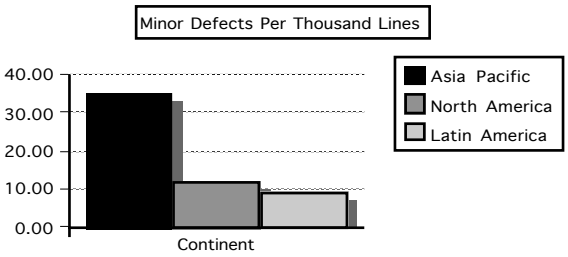


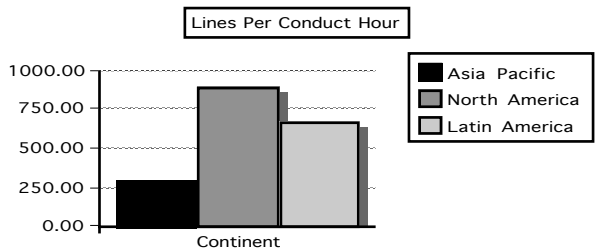
National Software Quality Experiment Metrics: Geographic Region

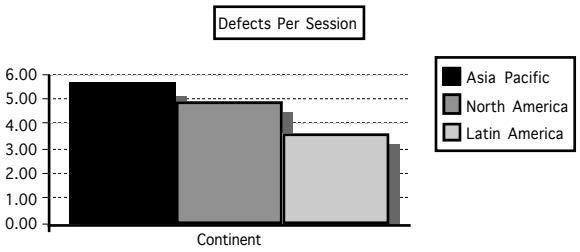


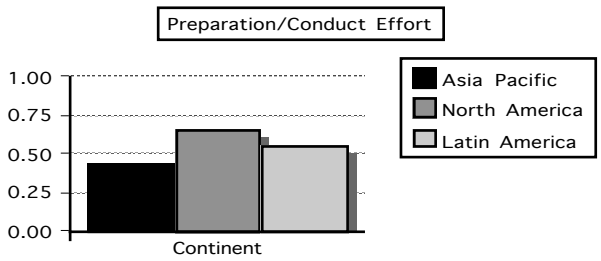


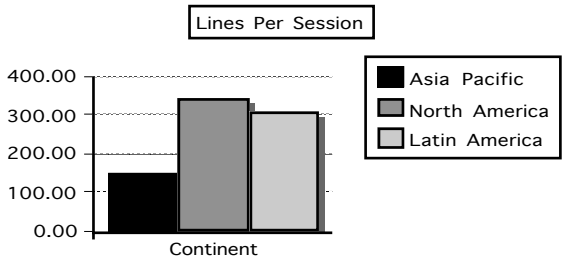


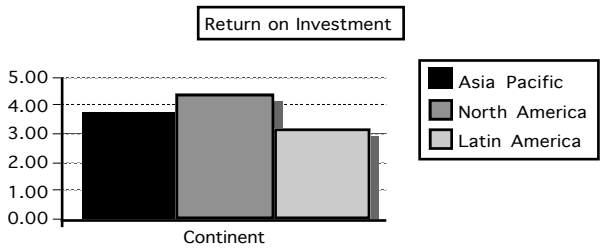


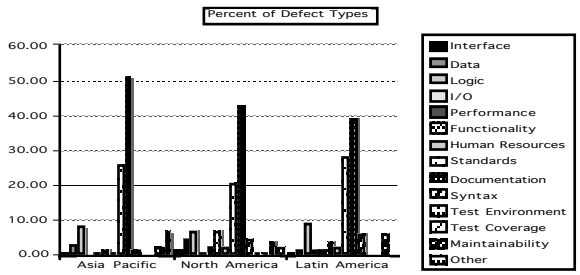




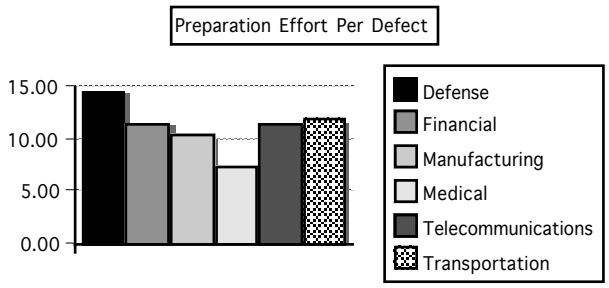


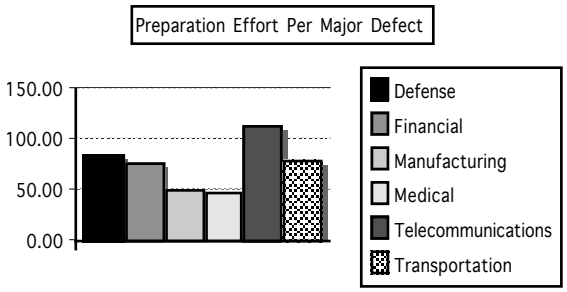


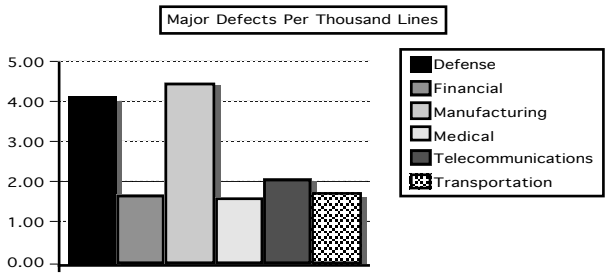


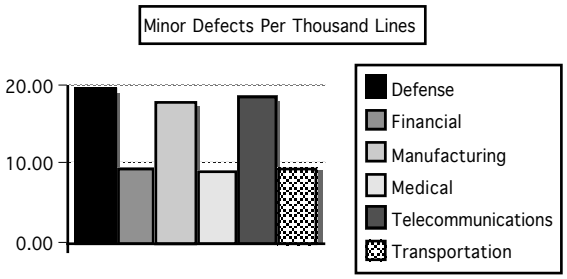


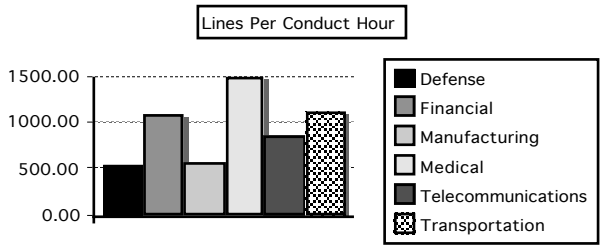
National Software Quality Experiment Metrics: Industry Type

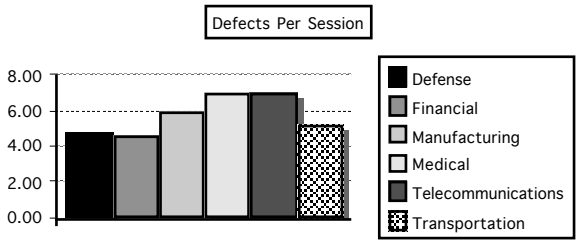


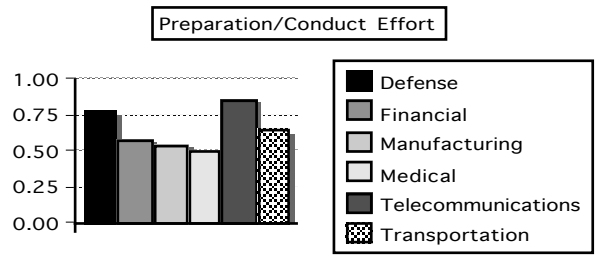


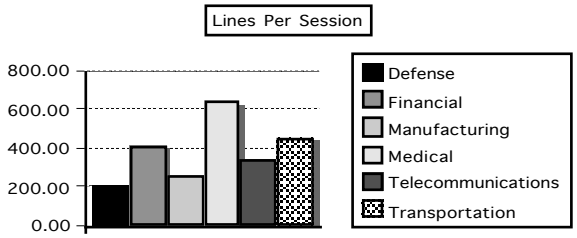


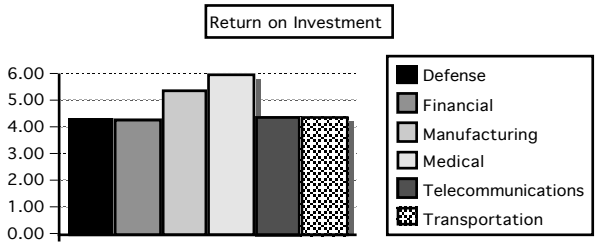


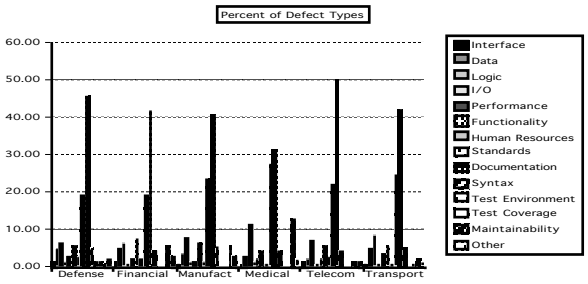






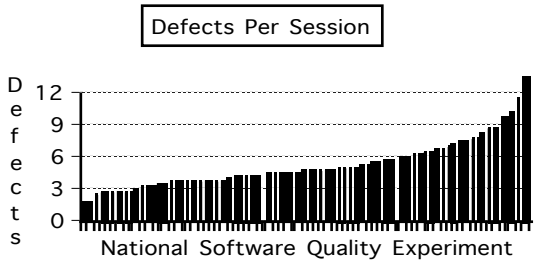


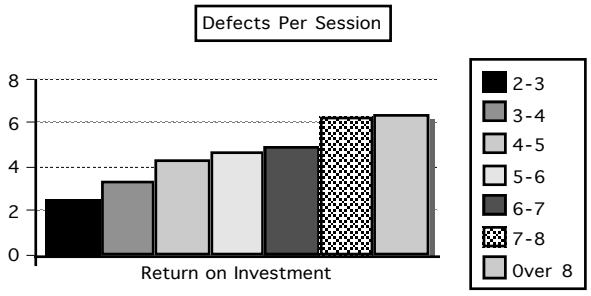


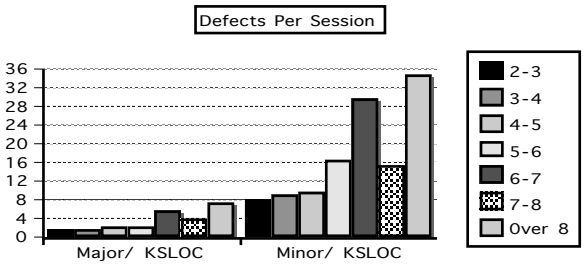


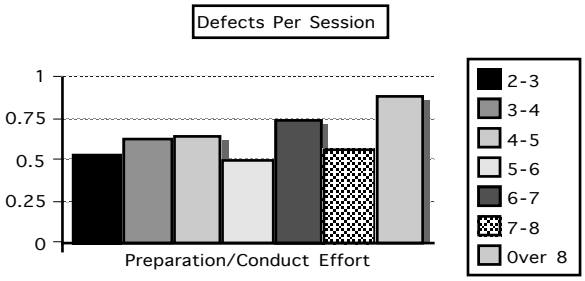
Defects Per Session Metric

The increase in defects per session is accompanied by a steadily increasing return on investment, an increasing defect density in both major and minor defects per thousand lines of source code, and an increasing preparation/conduct effort ratio.



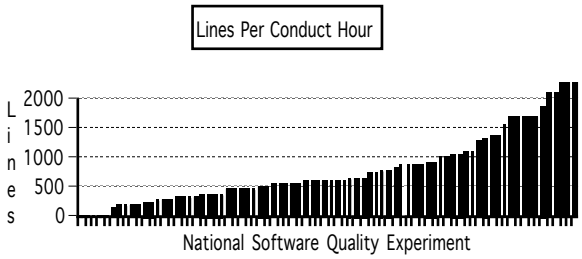


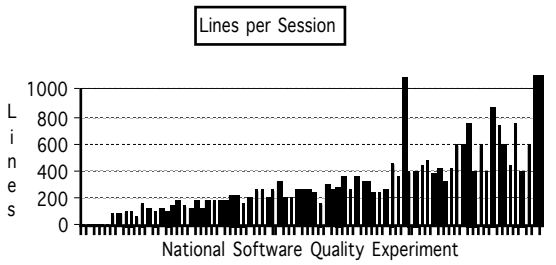


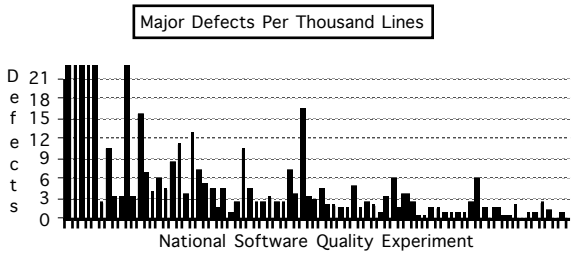


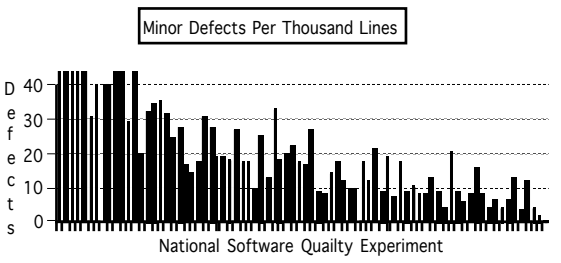
Lines Per Conduct Hour Metric

The increase in lines per hour is accompanied by a steadily increasing lines per session and a decreasing defect density in both major and minor defects per thousand lines of source code.





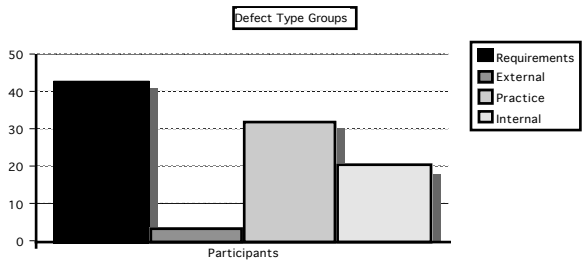


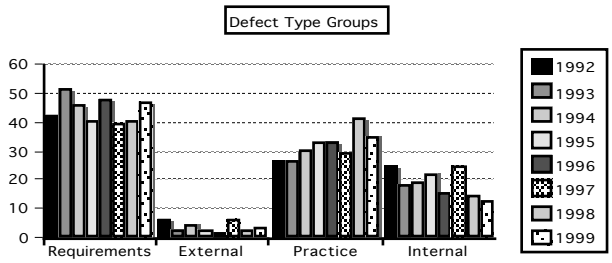


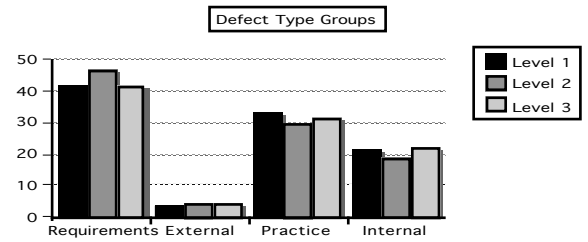
Defect Type Groups

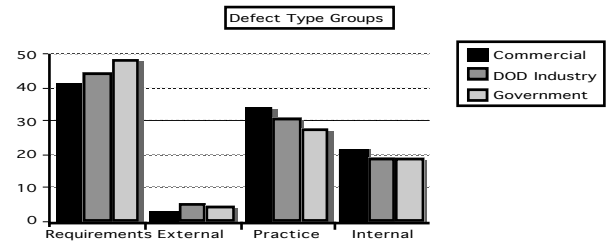
In addition to control panels, the NSQE Repository contains defect type distributions containing the frequency of occurrence of each defect type. These defect types can be organized for analysis as follows:

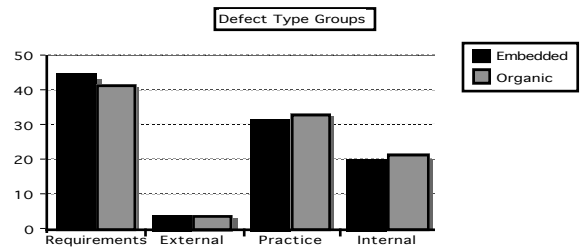
- Requirements
  - ◇ Documentation
- External
  - ◇ Interface, human factors, I/O
- Internal
  - ◇ Functionality, logic, data, performance
- Software practice
  - ◇ Syntax, standards, maintainability, other

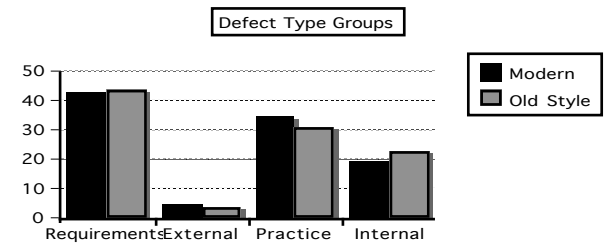


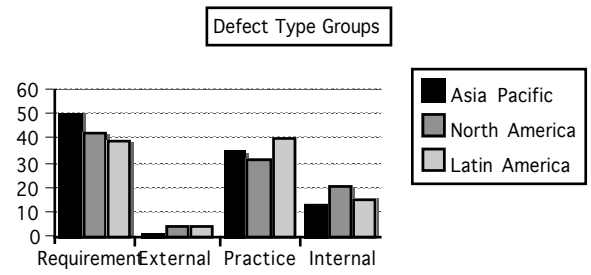


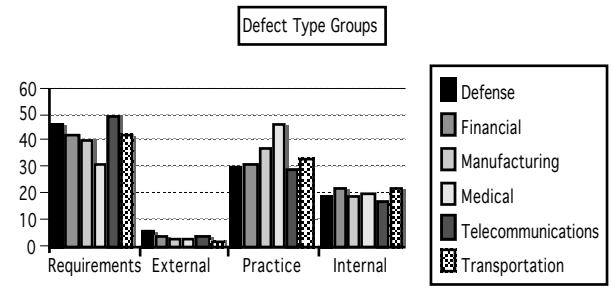










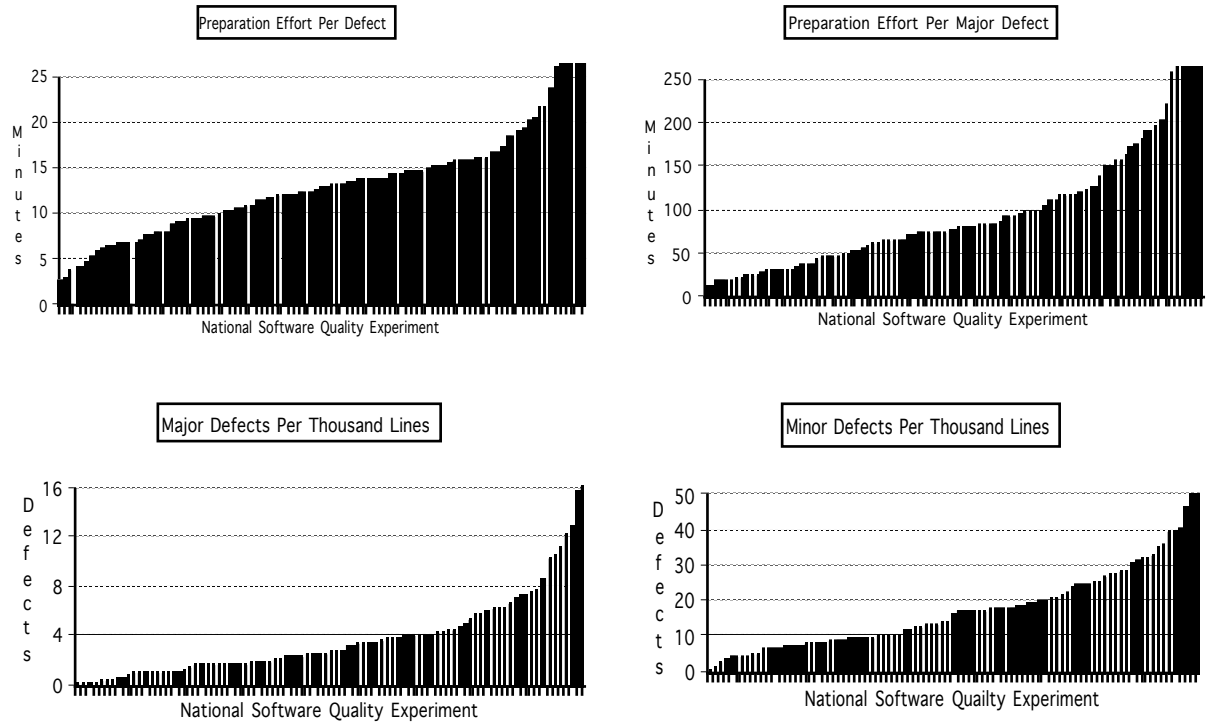


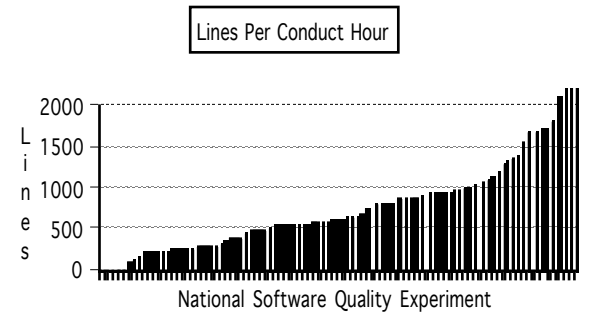
Derivation of Process Metrics: Control Panels

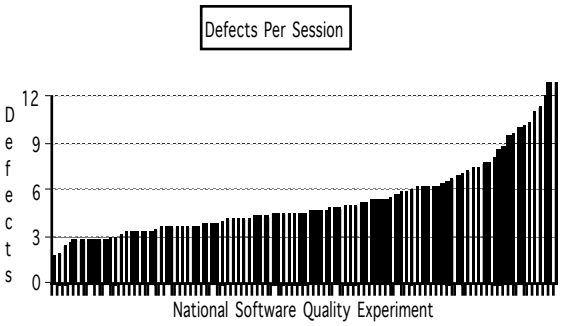
Metrics derived from these measurements are used in assessing the effectiveness and efficiency of the Software Inspections Process. It is useful to analyze the distribution of each derived metric. The following statistics are computed:

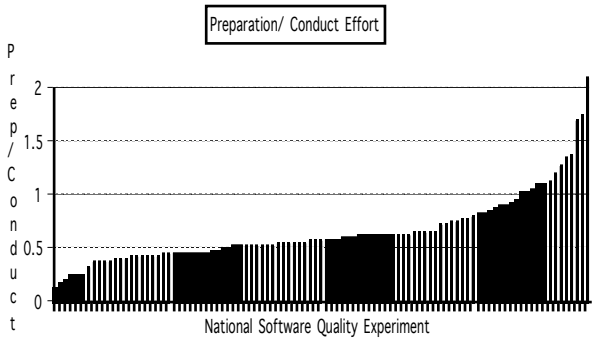
Bin	Prep/ Defect	Prep/ Major Defect	Major/ KSLOC	Minor/ KSLOC	Lines/ Conduct Hour	Defects Per Session		Prep/ Conduct	ROI
Average	12.18	72.24	2.46	12.15	858.7	4.90	0.64	4.50	
Max	63.64	700	52	168	3060	20	2.13	15.31	
Min	2.73	12.5	0.20	0.76	0	1.75	0.125	1.11	
STDEV	7.93	115.2	6.74	21.63	573.9	2.81	0.327	2.48	
LCL(20%)	7.96	39	1.08	8.07	292.7	3.35	0.433	2.85	
Median(50%)	12.82	79.83	2.87	16.79	624.5	4.471	0.571	3.98	
UCL(80%)	16.13	152	7.00	30.72	1046.8		6.745	0.821	6.67

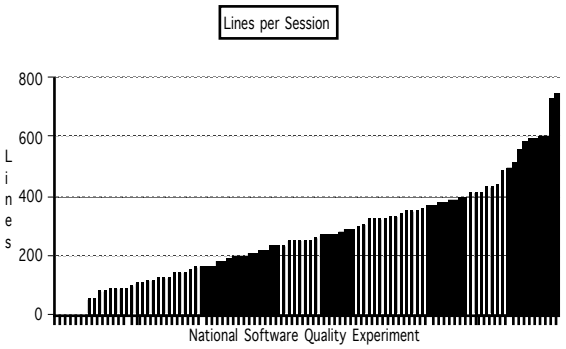
A graphic view of the sorted observations provides a quick look analysis of the distribution. These are shown below. It is clear that the observations on the extremes are less useful in predicting and setting expectation. Consequently control panels are derived by selecting the twentieth percentile, fiftieth percentile, and eightieth percentile. With these values the following control panels result.

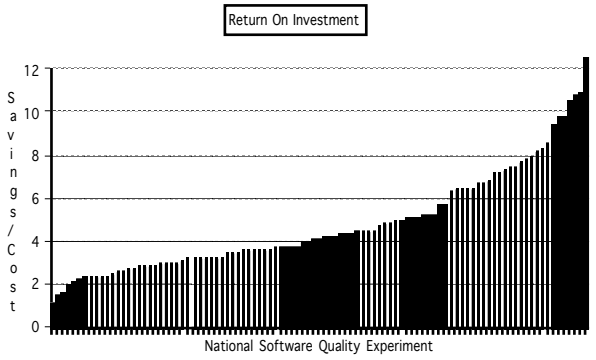


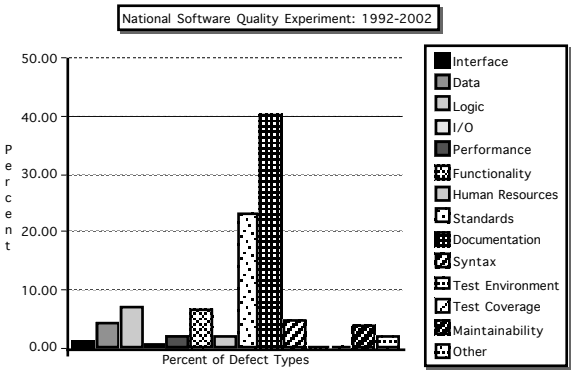

Lines Per Conduct Hour
Lines
2000
1500
1000
500
0
National Software Quality Experiment


Defects Per Session
Defects
12
9
6
3
0
National Software Quality Experiment


Preparation/ Conduct Effort
Prep/Conduct
2
1.5
1
0.5
0
National Software Quality Experiment

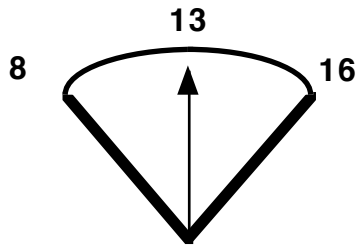

Lines per Session
Lines
800
600
400
200
0
National Software Quality Experiment


Return On Investment
Savings/Cost
12
10
8
6
4
2
0
National Software Quality Experiment


National Software Quality Experiment: 1992-2002
Percent
50.00
40.00
30.00
20.00
10.00
0.00
Percent of Defect Types
Interface
Data
Logic
I/O
Performance
Functionality
Human Resources
Standards
Documentation
Syntax
Test Environment
Test Coverage
Maintainability
Other

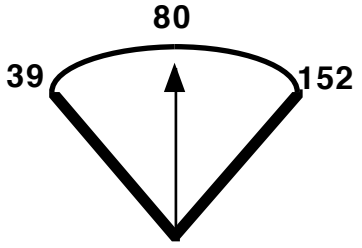
DERIVATION OF PROCESS METRICS: CONTROL PANELS

Defect Detection Rate Gauge



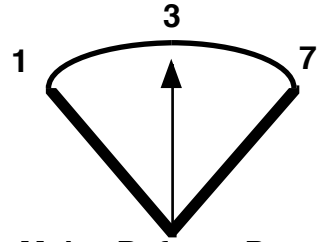
Minutes of Preparation Effort Per Defect

Defect Detection Rate Gauge



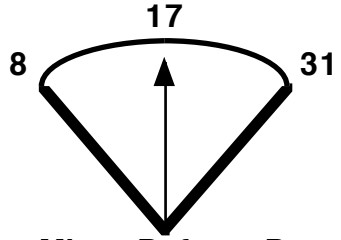
Minutes of Preparation Effort Per Major Defect

Defect Density Gauge



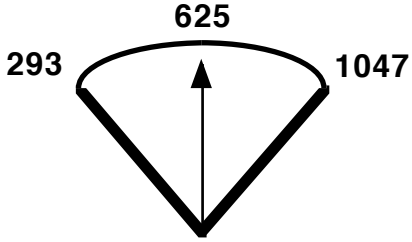
Major Defects Per Thousand Lines

Defect Density Gauge



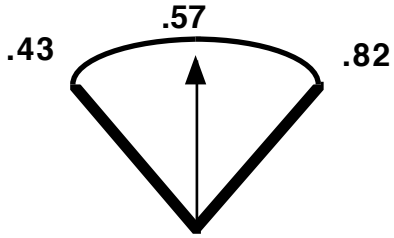
Minor Defects Per Thousand Lines

Inspection Conduct Rate Gauge



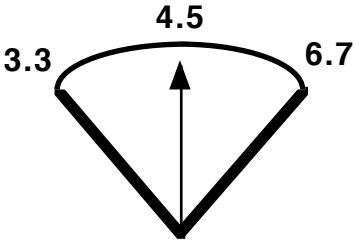
Lines Per Conduct Hour

Inspection Conduct Rate Gauge



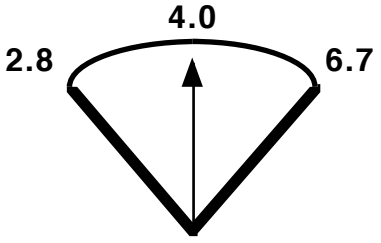
Preparation/Conduct

Defect Detection Rate Gauge



Defects Per Session

Return on Investment Gauge



Net Savings/ Detection Cost

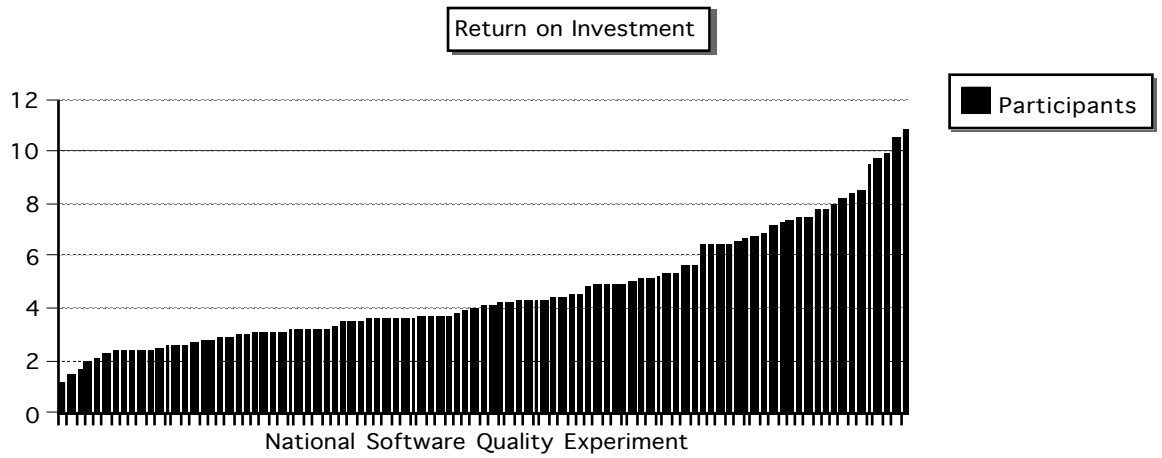
Quality and Product Metrics

Software Inspections practice yields useful measurements easy to obtain. Thesae measurements can then be used to derive quality and product metrics.

Measurements taken include preparation effort, conduct time, number of participants, size of artifact, major defects, and minor defects. Useful quality metrics derived from these measurements include size per session and preparation effort/conduct effort ratio where conduct effort is conduct time times number of participants. Useful product metrics derived from these measurements include major defects per size and minor defects per size.

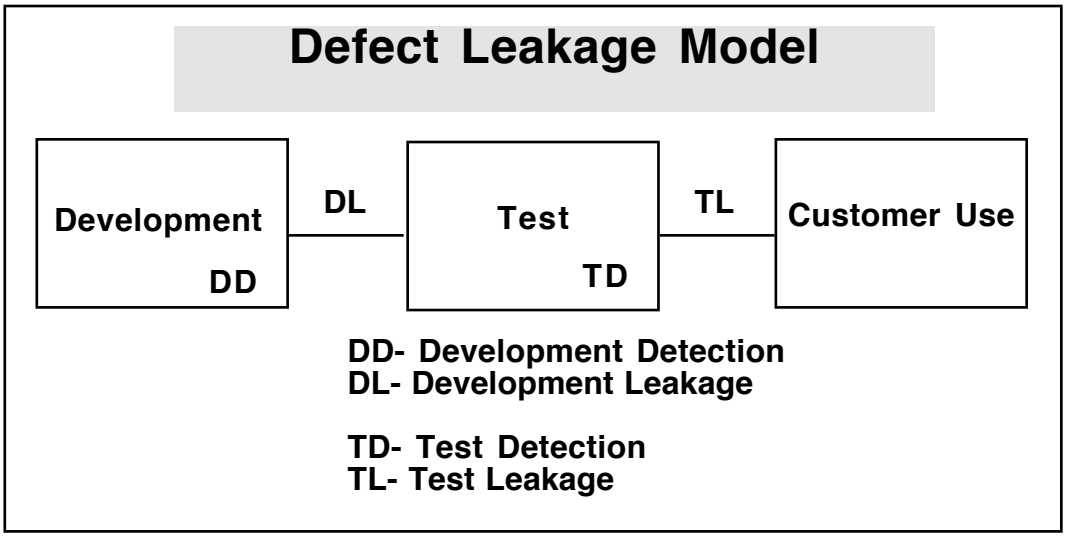
Return on Investment

Managers are interested in knowing the return on investment to be derived from software process improvement actions. The National Software Quality Experiment gathers the data needed to determine this for the Software Inspections Process.



The return on investment for software inspections is defined as net savings divided by detection cost, where net savings is cost avoidance less cost to repair now and detection cost is the cost of preparation effort and the cost of conduct effort. Savings result from early detection and correction avoiding the increased cost multiplier associated with detection and correction of defects later in the life cycle.

An undetected major defect that escapes detection and leaks to the next phase of the life cycle may cost two to ten times to detect and correct. A minor defect may cost two to four times to detect and correct. The net savings then may be up to nine times for major defects and up to three times for minor defects. The defined measurements collected in the Software Inspections Lab may be combined in complex ways to form this derived metric. A full discussion of these complexities can be found in "Return on Investment Using Software Inspections" at http://members.aol.com/ONeillDon/roi-essay.html. For those with software inspections results from their organization, the online calculator found at http://members.aol.com/ONeillDon/nsqe-roi.html can be used to compute return on investment.



Computing ROI
ROI: Net Savings/Detection Cost

Reasoning About Net Savings
Net Savings: Cost Avoidance-Cost to Repair Now

Net Savings:
- Major Defects * {(M1 * DD)+(M1 * DD) * (M2 * TL)*C1-C1}+Minor Defects * (M3-C2)

Where:
- M1: (2-10) Additional Cost to Repair Multiplier for Development to Test Major Defect Leakage
- M2: (2-10) Additional Cost to Repair Multiplier for Test Customer Use Major Defect Leakage
- M3: (2-4) Additional Cost to Repair for Minor Defect Leakage
- DD: (.5-.95) Defect Detection Rate for Development to Test
- TL: (.05-.5) Test Leakage Rate for Test to Customer Use
- C1: Average Cost to Repair Major Defect
- C2: Average Cost to Repair Minor Defect

Reasoning About Detection Cost
Detection Cost:
- Preparation Effort + Conduct Effort

Detection Cost:
- {Minutes of Preparation Effort + (Minutes of Conduct Time * P)}/60

Where:
- P: (4-6) Number of participants
- 60 minutes per hour

ANALYSIS OF ANNUAL AVERAGES
The 1992 hypothesis being investigated in the National Software Quality Experiment is whether software problems are being reduced by a factor of ten by the year 2000. An analysis of annual averages and ranges suggests that a moderately stable process is in

operation, and that there is little pressure to reestablish the control limits at an improved level of practice.

Analysis of Annual Averages

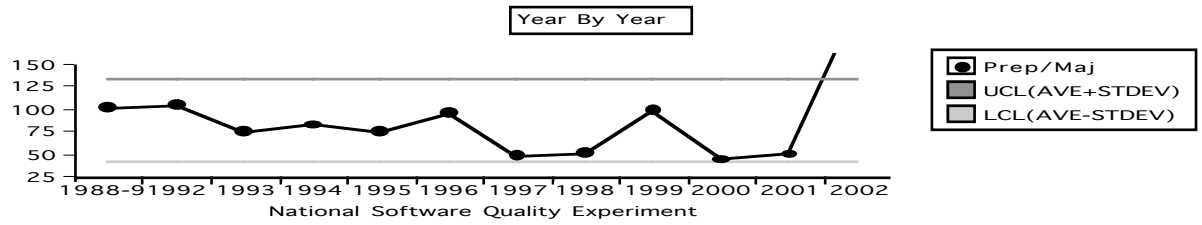
Metric	In	Out	
Prep/Defect	7	5	
Prep/Major	11	1	
Major/KSLOC	8	4	
Minor/KSLOC	8	4	
Size/Conduct	10	2	
Defect/Session		9	3
Prep/Conduct	9	3	
Size/Session	9	2	
ROI	10	2	
Total	81	27	

Averages

An analysis of the averages by year throughout the time period for the metrics under study suggest that these metrics are moderately stability [SPC 92]. With 12 years and 9 metrics there are 108 data points. Eighty-one (81) fall within one standard deviation of the average and 27 outside these limits. A discussion of each average metric follows.

Prep/Defect- The average prep/defect has operated in a stable manner in the early years with less predictability in the later years. A lower prep/defect may suggest:
1. There may be more inserted defects to find.
2. Practitioners may be getting better at detecting defects.

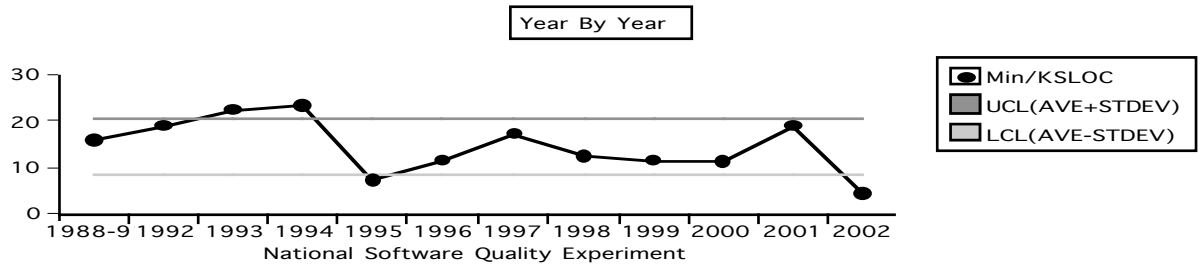
Prep/Maj- The average prep/maj has operated in a stable manner almost throughout the period. A lower prep/defect may suggest:
1. Practitioners may be getting better at detecting defects... especially major defects.
2. Practitioners may be inserting more defects.



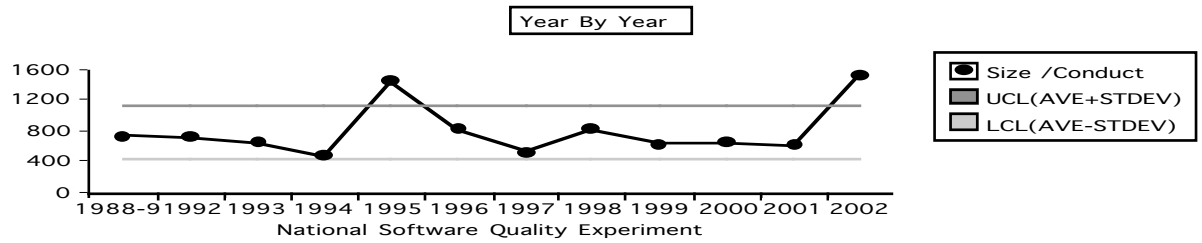
Maj/Thousand- The average maj/thousand operated in a moderately unstable manner throughout the time period.
1. This metric is sensitive to the distinction between new development code and legacy code.
2. This metric is sensitive to the defect insertion rate. Therefore, a high average maj/thousand may simply indicate the presence of a large number of defects.
3. This metric is sensitive to software product engineering mode which influences both

detect insertion and detect detection rate. Therefore, a high average maj/thousand may indicate a high defect detection rate and the presence of a low number of defects.

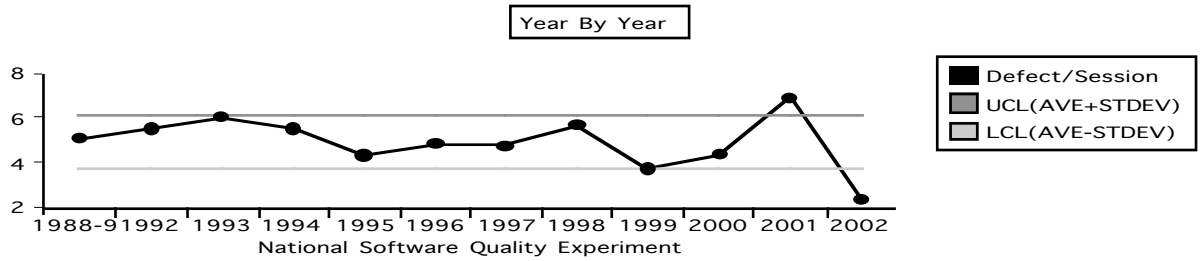
Min/Thousand- Average min/thousand has operated in a moderately stable manner for four consecutive data points preceded by a special cause data point.
1. Practitioners seem to detect a steady volume of minor defects.
2. This metric is sensitive to the distinction between new development code and legacy code.
3. This metric is sensitive to the defect insertion rate. Therefore, a high average maj/thousand may simply indicate the presence of a large number of defects.
4. This metric is sensitive to software product engineering mode which influences both detect insertion and detect detection rate. Therefore, a high average maj/thousand may indicate a high defect detection rate and the presence of a low number of defects.



Size/Conduct Hour- Average size/conduct hour operated in a moderately stable manner.
1. This metric is sensitive to the distinctions between new development code and legacy code.
2. This metric is sensitive to the preparation effort prior to the session. Higher preparation effort may yield higher size/conduct hour.

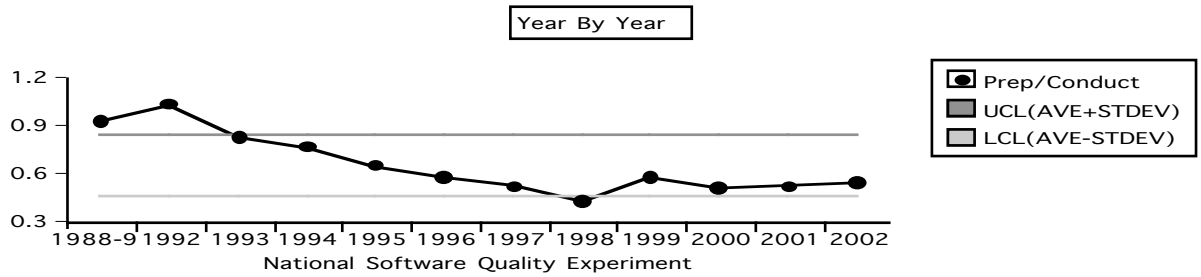


Defects/Session- Average defects /session operated in a moderately stable manner throughout the time period.
1. Practitioners seem to detect a steady number of defects per session.
2. This metric is sensitive to the preparation effort prior to the session. Higher preparation effort may yield higher defects/session.



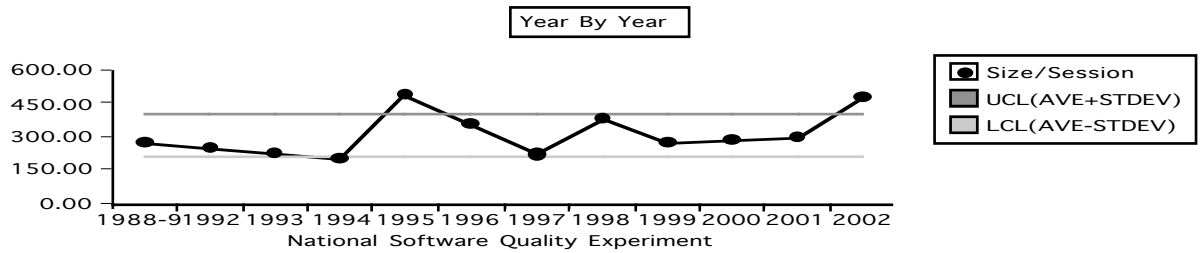
Prep/Conduct- Average prep/conduct has consistently trended downward over the time period.

1. Practitioners increasingly are experiencing excessive overtime and even off the clock time and neglect preparation effort for software inspections.
2. Where average prep/conduct approaches equilibrium (1.0), defect leakage may be reduced.



Size/Session- Size/session operated in a moderately stable manner throughout the time period.

1. One way to look harder for defects is to reduce size/session.
2. This metric is sensitive to the distinctions between new development code and legacy code.
3. This metric is sensitive to the preparation effort prior to the session. Higher preparation effort may yield higher size/conduct hour.



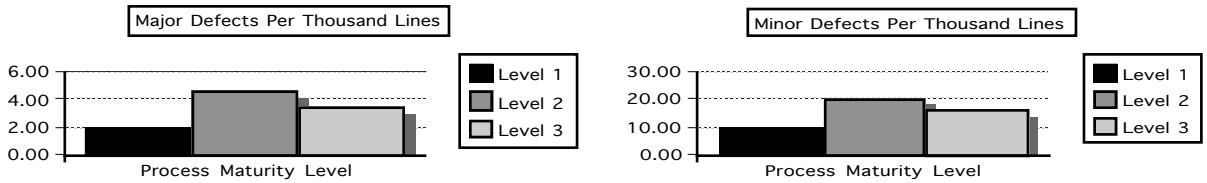
Savings/Cost (ROI)- Average savings/cost has operated in a marginally stable manner throughout the time period.

1. Return on investment fuels management commitment to the software process.
2. High defect detection results in high return on investment.

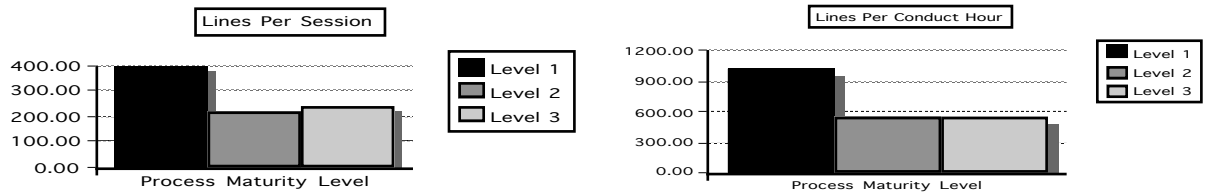

Year By Year
4.6
4.5
4.4
4.3
4.2
4.1
1988-9
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
National Software Quality Experiment
ROI
UCL(AVE+STDEV)
LCL(AVE-STDEV)

Reasoning About Findings: Software Process Maturity

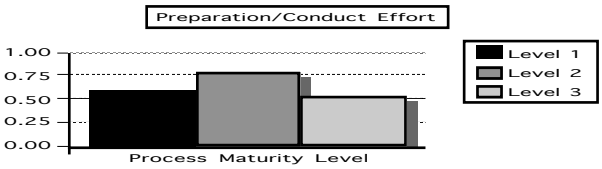
The level 1 major and minor defects per thousand lines are less than half of level 2 and level 3. Are level 1 organizations inserting less defects ... or simply finding less?



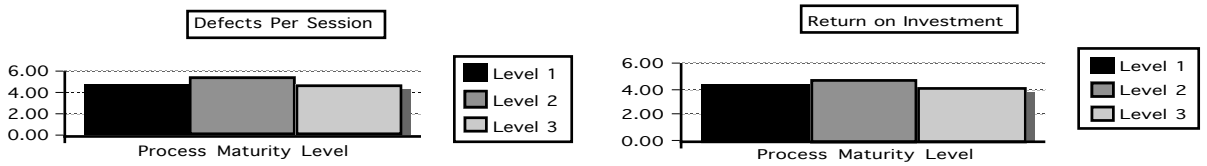
Level 1 lines per conduct hour and lines per session are double level 2 and level 3. One way to look harder for defects is to inspect smaller artifacts. Level 2 and level 3 organizations are looking harder... and finding more.



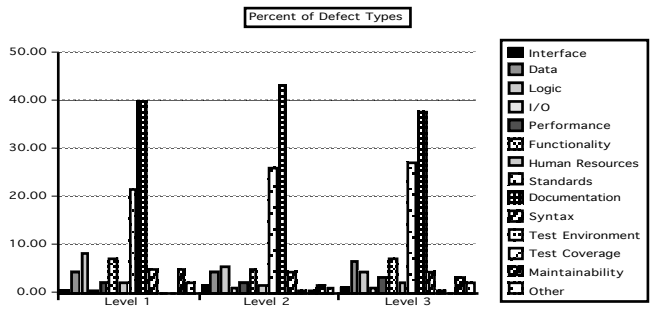
Level 2 preparation/conduct effort approaches 1.0 which is the desired equilibrium. Another way to look harder for defects is to increase preparation time. Level 1 and level 3 show a large shortfall in preparation... suggesting that more defects could be detected.



Level 1, 2, 3 defects per session and return on investment are consistently repeatable.

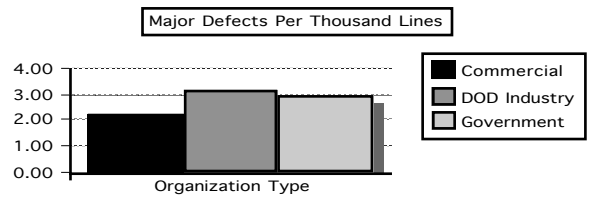


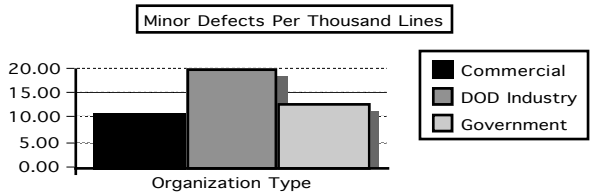
Level 1, 2, 3 defect type distributions are consistently repeatable.



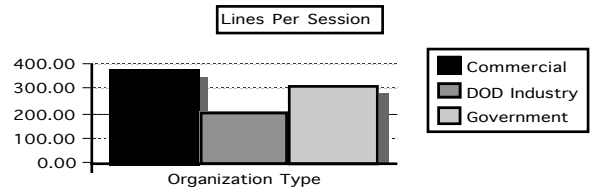
Reasoning About Findings: Organization Type

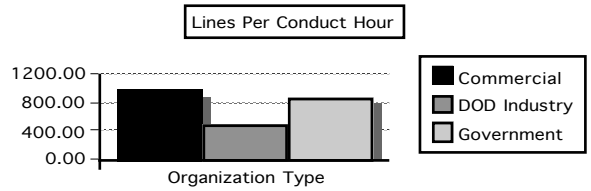
DOD Industry seems to find more defects. Interestingly DOD Industry detects substantially more minor defects. While minor defects do not effect execution, their detection and correction is beneficial for maintenance.



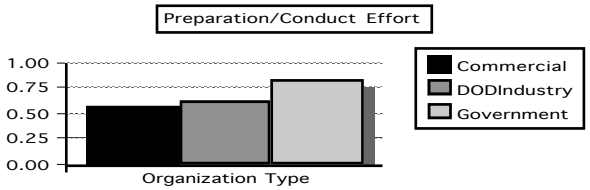


Again DOD Industry stands out because it is inspecting smaller artifacts. One way to look harder for defects is to inspect smaller artifacts.

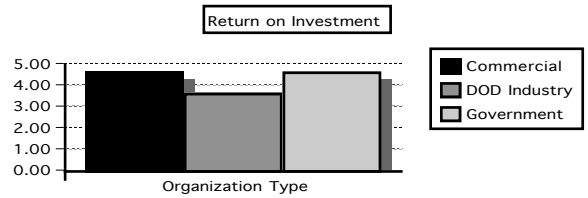


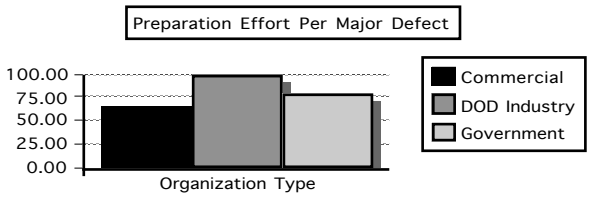


The Government preparation/conduct effort approaches 1.0 which is the desired equilibrium. Another way to look harder for defects is to increase preparation time.

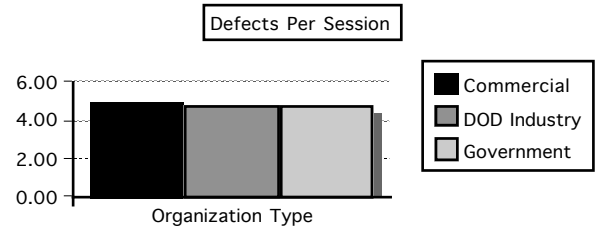


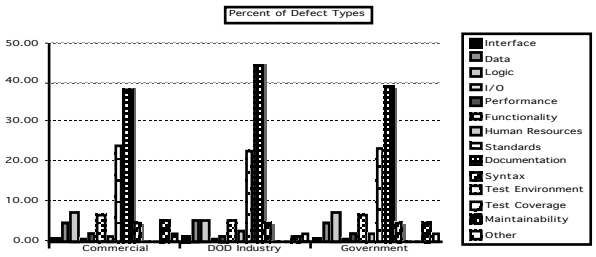
The DOD Industry return on investment lags others because its preparation per major defect is higher, and major defects strongly influence return on investment. Recall that DOD Industry looked harder by inspecting smaller artifacts.





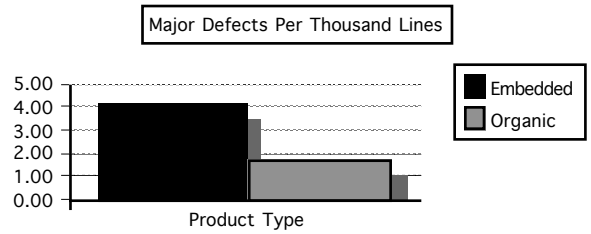
The defects per session and defect type distributions are consistently repeatable.

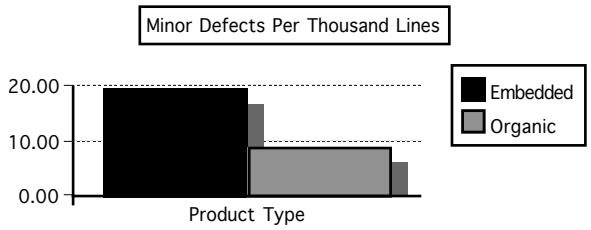




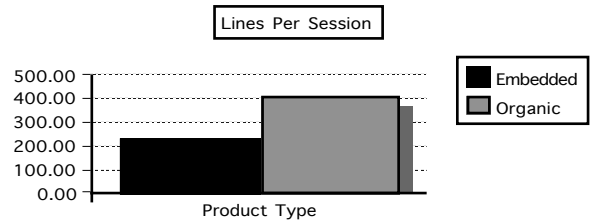
Reasoning About Findings: Product Type

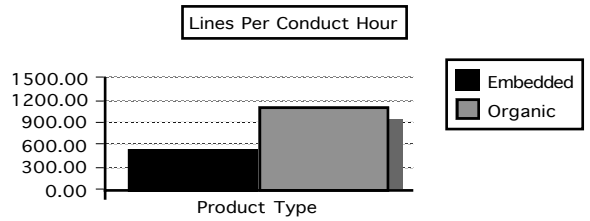
Embedded systems defect detection aismore than twice that of organic systems.



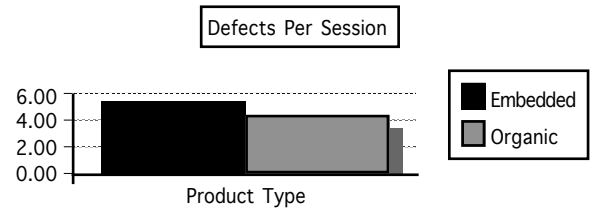


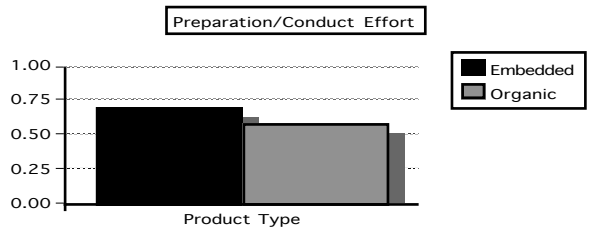
Embedded systems are inspecting smaller artifacts. One way to look harder is to inspect smaller artifacts.



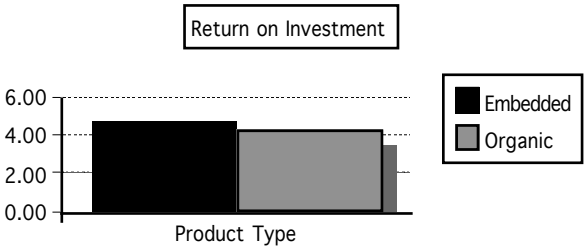


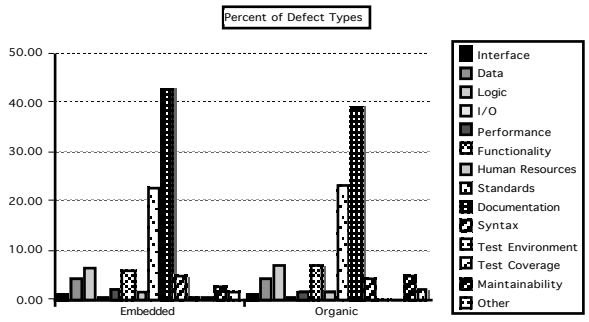
Embedded systems are slightly higher than organic systems in defects per session and preparation/conduct effort.





The return on investment and defect type distributions are consistently repeatable.





Special Study on Personal Software Process

The National Software Quality Experiment (NSQE) collects core samples of software product quality in the Software Inspection lab. Disciplined software engineering is the most formal practice of software product engineering and might be patterned after Clean Room software engineering, Personal and Team Software Process, and Extreme Programming techniques. As the National Software Quality Experiment (NSQE) operation expands to capture software inspection measurements from organizations of all kinds, it is becoming a resource to answer important software engineering questions.

While PSP is expected to enable level 4 and 5 capability, it is also expected to deliver measurably superior performance and products. But how does the practice of Personal Software Process compare to organizational performance at SEI CMM process maturity levels 1, 2, and 3?

National Software Quality Experiment

The National Software Quality Experiment (NSQE) collects core samples of software product quality in the Software Inspection lab. These core samples are assigned to analysis bins including year, software process maturity level, product type, organization type, programming language, global region, and industry type. Dozens of organizations have supplied thousands of core samples to this national database. The NSQE supplies the initial measurements and metrics needed by an organization to set its expectations for minutes of preparation effort per defect, defects per thousand lines of code, lines inspected per conduct hour and session, defects per session, preparation/conduct effort, and return on investment.

Disciplined Software Engineering

Disciplined software engineering is the most formal practice of software product engineering and might be patterned after Clean Room software engineering, Personal and Team Software Process, and Extreme Programming techniques. The result is expected to be well specified, professionally engineered, expertly architected with source code components organized and made understandable through templates for repeating patterns whose completeness, correctness, style, and rules of construction can be reasoned about with confidence. For many, this is the expectation for an SEI CMM level 4 and 5 organization. It is expected to experience low defect insertion and high defect detection rates. Without question, the focus of disciplined software engineering practitioners is on eliminating every possible defect even if defect detection costs exceed net savings and the return on investment falls below the break even point. For this group, every practitioner is riveted on achieving perfection.

Special Study Basis

As the National Software Quality Experiment (NSQE) operation expands to capture software inspection measurements from organizations of all kinds, it is becoming a resource to answer important software engineering questions. For example, NSQE is beginning to experience data collection from the practice of Personal Software Process (PSP). While the NSQE analysis bin of PSP results is thin containing only 86 inspection sessions, the comparison of results by Software Engineering Institute (SEI) Capability Maturity Model (CMM) level may be of interest.

Special Study Results

In conducting the special study, the NSQE metrics collected from PSP practitioners are

gathered and matched with the NSQE metrics results collected from level 1, level 2, and level 3 participants. The PSP/ level ratios for NSQE metrics are derived for each level 1-3. An analysis of the variances for the ratios of each metric and each level indicates a total variance for level 1 of 40.53, for level 2 of 25.20, and for level 3 of 26.05. A review of each individual metric shows that the least variance occurred once for level 1, three times for level 2, and five times for level 3. Therefore, the small sample of PSP results collected so far indicates that PSP performance most closely matches that of level 3 practitioners.

While there is a minimum variance between PSP and level 3 results and many of the NSQE metrics are closely comparable, three specific metrics standout in contrasting PSP to level 3 results:

1. Preparation effort per major defect is 195% [126.13 PSP versus 64.81 level 3]
2. Major defects per thousand lines of code is 62% [2.33 PSP versus 3.78 level 3]
3. Minor defects per thousand lines of code is 136% [22.22 PSP versus 16.34 level 3]

PSP and Level 3 Comparison

NSQE Metric	PSP	Level 3	%
minutes of preparation effort per defect	11.95	12.18	98
minutes of preparation effort per major defect	126.13	64.81	195
major defects per thousand lines of code	2.33	3.78	62
minor defects per thousand lines of code	22.22	16.34	136
lines inspected per conduct hour	460	520	88
defects per session	4.66	4.72	99
preparation/conduct effort	.56	.53	105
lines per session	189	234	81
return on investment	3.18	4.28	74

TSP Quality Guidelines

In addition, it is interesting to note that the lines of code per conduct hour suggested in the Team Software Process (TSP) Quality Guidelines is less than 200 lines compared to the PSP usage in NSQE measured at 460 lines. It should be noted that the lines of code per session for NSQE PSP is 189.

Conclusion

While PSP is expected to enable level 4 and 5 capability, it is also expected to deliver measurably superior performance and products. While this study is based on a thin sample of NSQE PSP results, these results do not indicate a convincingly superior performance associated with PSP practice.

Bibliography

1. O'Neill, Don, "National Software Quality Experiment: A Lesson in Measurement 1992-2000", http://members.aol.com/ONeillDon/nsqe-results.html
2. "Personal Software Process", CMU/SEI-2000-TR-022
3. "Team Software Process", CMU/SEI-2000-TR-023
4. Paulk, Mark C., "The Capability Maturity Model: Guidelines for Improving the Software Process", Addison-Wesley Publishing Company, 1995

Metric	PSP	Level 1	Percent	Level 1 Net
Prep Effort /Defect	11.95	11.543855816	103.51827145	3.51827145
Prep Effort/Major	126.13	70.503002001	178.90018357	78.9001835
Major/KSLOC	2.33	1.9664096822	118.49005937	18.4900593
Minor/KSLOC	22.22	10.043250518	221.24311207	121.243112
Lines/Conduct Hour	460	1023.8210145	44.929728293	55.0702717
Defects/Session	4.66	4.7312661499	98.493719279	1.50628072
Prep/Conduct Effort	0.56	0.59141782692	94.687710533	5.31228946
ROI	3.18	4.4617061678	71.273182957	28.7268170
Lines/Session	189	393.95503876	47.975017808	52.0249821
Average				40.5324741

Metric	PSP	Level 2	Percent	Level 2 Net
Prep Effort /Defect	11.95	14.220472441	84.033776301	15.9662236
Prep Effort/Major	126.13	80.094420601	157.47663702	57.4766370
Major/KSLOC	2.33	4.4273851825	52.627	47.373
Minor/KSLOC	22.22	20.509117627	108.34205744	8.34205744
Lines/Conduct Hour	460	549.05581638	83.780188876	16.2198111
Defects/Session	4.66	5.2076719577	89.483362967	10.5166370
Prep/Conduct Effort	0.56	0.81125021735	69.029257314	30.9707426
ROI	3.18	4.5739931839	69.523496694	30.4765033
Lines/Session	189	208.83730159	90.501073593	9.49892640
Average				25.2045043

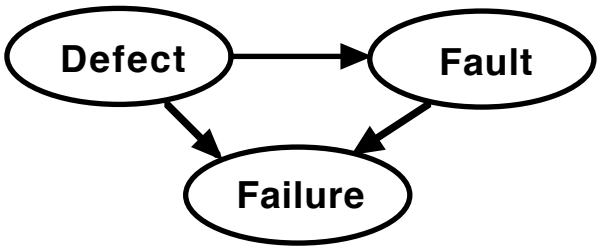
Metric	PSP	Level 3	Percent	Level 3 Net
Prep Effort /Defect	11.95	12.183465459	98.083751627	1.91624837
Prep Effort/Maior	126.13	64.807228916	194.62335007	

CRITICAL DEFECT PREDICTION

Critical Defect, Fault, and Failure Prediction

If critical software defects, faults, and failures can be predicted, perhaps they can be detected, controlled, and prevented. If this could become standard software practice, the software industry could replace chaos and unpredictability with trustworthy software systems that earn public confidence. The goal is to be the recognized leader in trusted software systems practice.

There is insufficient defect, fault, and failure data available from the nation's factory floor. There is insufficient process, method, and tooling to combine defect data obtained through software inspections practice, software fault data obtained through software product test and use, and software failure data obtained through software system operation into predictions of trustworthy software system operation.



Trustworthy software systems are dependable in operation from a user perspective. Trustworthy software systems are convincingly reliable from an engineering perspective. Engineers assess and predict the reliability of a system in terms of fault analysis, mean time to failure, availability, and mean time to repair. Managers report on the emerging quality of the software system in terms of completeness, correctness, conformance to requirements, compliance with standards, adherence to rules of construction for the application domain, and various viewpoints.

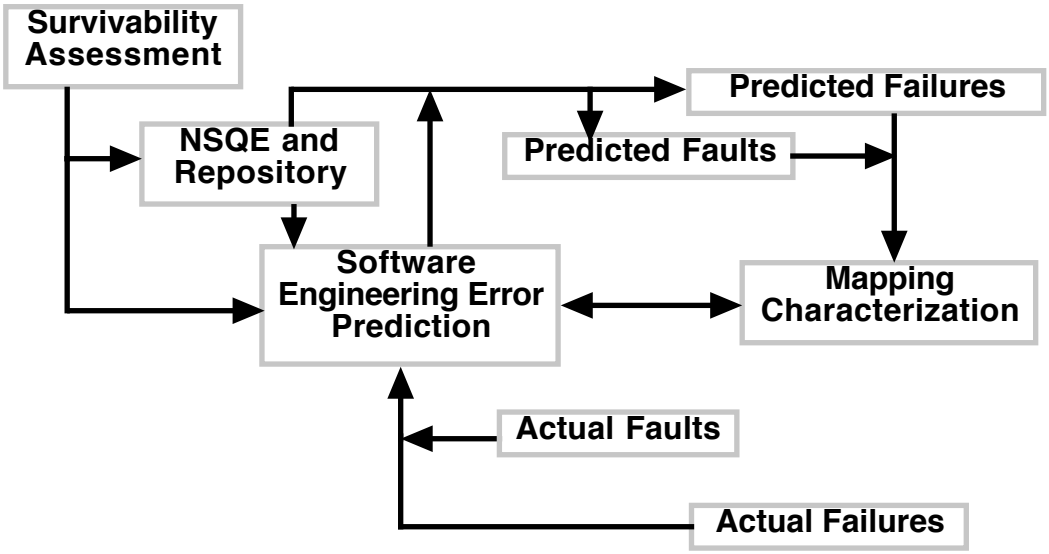
The trustworthiness of software systems threatens the harmonious operation of critical industries and is impacting public confidence in the orderliness of society and its institutions both public and private. Bridging the gap of prediction practice among defects, faults, and failures remains an unsolved problem.

- Defects are detected early using software inspections as exit criteria for activities in the software life cycle. A defect is an instance where the software artifact does not meet the standard of excellence set as the exit criteria for the activity.
- Faults are detected later through exercise during integration and system test. A fault is an instance where the exercise of a software component yields an incorrect result.
- Failures are detected in operational test and operational deployment. A failure is a user visible instance where the operation of a software system does not meet expectation.

Critical Defect Prediction Model

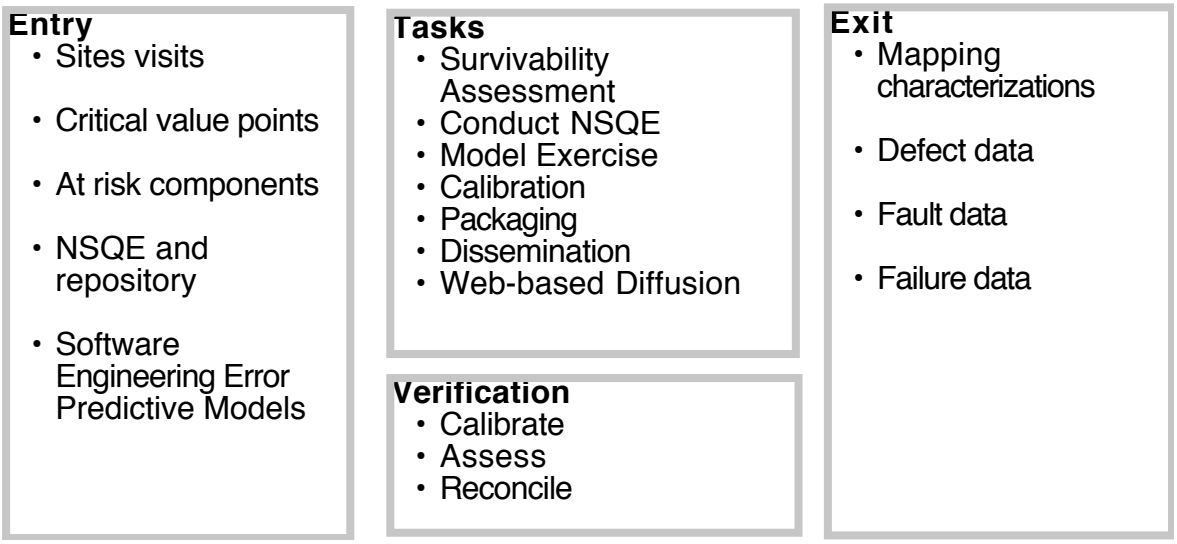
Critical Defect, Fault, and Failure Prediction project is based on the integration of three models. Critical components are pinpointed through a *Survivability Assessment* of the concept of operations, software architecture, and the rules construction for its components. *The National Software Quality Experiment (NSQE)* with its Software Inspection Lab and its Repository of core samples uses defect detection to derive metrics capable of calibrating defect leakage prediction and defect leakage type distribution.The question then is, "To what extent are *Software Engineering Error Prediction Models* capable of utilizing defect

leakage prediction and defect leakage type distribution to predict faults and failures?" The answer lies in the integration of models.



Disseminating the knowledge, skills, and behaviors for predicting defects, faults,and failures among the industry practitioners on the factory floor is accomplished by push and pull. Several initiatives generate pull including SEI CMM, 6-sigma,and ISO 9001, and the growing number of lawsuits stemming from software failure.Push is generated through distribution of an error prediction kit composed of data base structures, data repository, spreadsheet templates, and user handbook; licensing the use of certain data; reporting to user groups and industry conferences. Once a community of interest has been assembled, the diffusion strategy can best be operated from an open web site on the internet following the new rules of the new economy.

Project Plan



Prediction Goal, Question, Metric

The goal, question, metric (GQM) template introduced by Dr. Vic Basili can be used to focus the approach:

- The goal is to utilize defect data from the NSQE and the Software Inspection Lab to predict critical faults and failures and to calibrate Software Engineering Error Prediction Models.
- Several questions are asked. What are the critical software components? What is the defect type distribution of faults and failures? What is the defect leakage from design and code into test operations and from test to field operations?
- The metrics generated by the NSQE include both Software Inspection Lab Operations control panels and defect type distributions.

The NSQE Repository contains thousands of core samples from which control panels of upper and lower limits are derived. The control panel metrics are based on personnel effort, software component size, and defects detected and include:

- Minutes of preparation effort per major defect
- Minutes of preparation effort per defect
- Minutes of preparation effort per major defect
- Major defects per thousand line of code
- Minor defects per thousand line of code
- New development lines per conduct hour
- Legacy lines per conduct hour
- Defects per session
- Preparation effort / conduct effort
- Lines per session
- Return on investment

Spreadsheet Road Map

Spreadsheet road map and expressions:

A	Title	analysis bin name
B	Prep Effort	minutes of effort reported
C	Conduct Time	minutes of wall clock time
D	Major Defects	defects affecting execution
E	Minor Defects	defects not affecting execution
F	Lines of Code	non-blank lines in artifact being inspected
G	Pages of Doc	pages in artifact being inspected
H	Sessions	number of inspections

Spread Sheet Expressions

Prep/Defect	"=B/(D+E)"
Prep/Major	"=B/D"
Major/Size	"=D/(F/1000)"
Minor/Size	"=E/(F/1000)"
Size/Conduct Hour	"=F/(C/60)"
Defects/Session	"=(D+E)/H"
Prep/Conduct	“=B/(C*4)”
Lines /Session	"=F/H"
Return on Investment	“=((D*9)+E)/((B+(C*4))/60)”

Software inspections practice detects 60-90% of defects inserted. Occasionally this drops below 50%; infrequently it exceeds 95%. A defect detection range of 60-90% is defect leakage range of 40-10%. If the nominal defect detection is 75%; the nominal defect leakage would be 25%. These control panel metrics reveal the effectiveness of the software inspections practice and the expectation for detection and leakage.

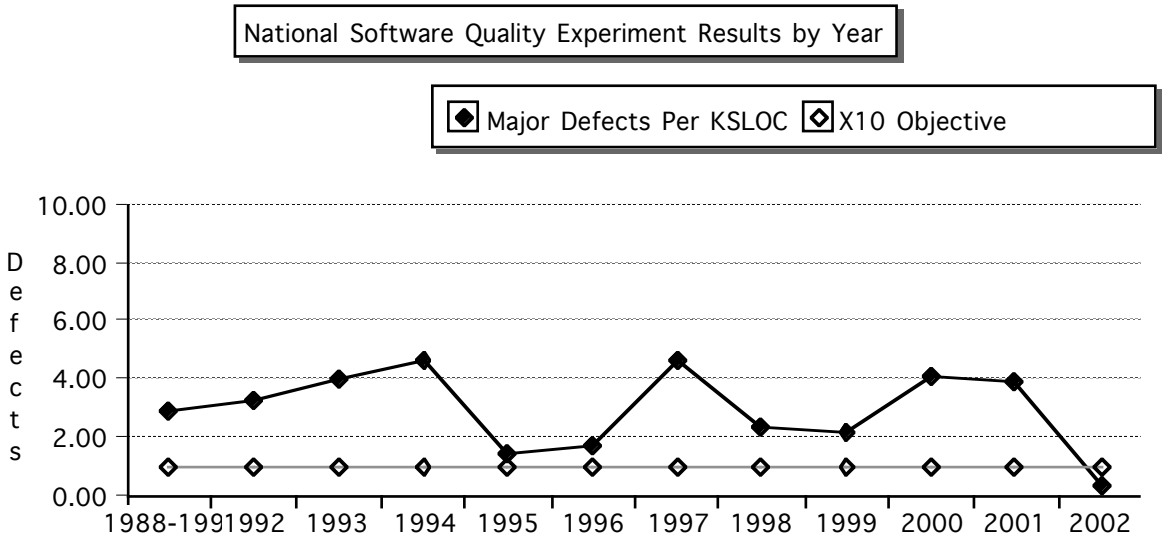
Fault Detection	Detection Scale
Prep/Maj	"=((Prep/(Major)-81))/(152-47)"
Prep/Defect	"=((Prep/(Maj+Min))-13)/(16-9)"
Maj/KSLOC	"=(Maj/(Size/1000)-3)/(6-1)"
Min/KSLOC	"=(Min/(Size/1000)-18)/(31-9)"
Size/Conduct	"=-((Size/(Conduct/60)-602)/(1115-292))"
Defects /Session	"=((Maj+Min)/Sessions-4.7)/(6.6-3.5)"
Prep/Conduct	"=(Prep/(Conduct*4)-.58)/(.84-.44)"
Lines/Session	"=((F/H")-255)/(420-125)
ROI	"=(((Maj*9)+Min)/((Prep+(Conduct*4))/60)-3.9)/(6.4-2.8)"
Average	“= SUM(Detection Scale Values)”
Prediction	
Predicted Major Leakage	"=(Maj/(Size/1000)*(1-(0.6+0.3*(Average))))"
Predicted Minor Leakage	"=(Min/(Size/1000)*(1-(0.6+0.3*(Average))))"

CONCLUSION

CLOSING OBSERVATIONS

The hypothesis of the National Software Quality Experiment set in 1992 is that software problems will be reduced by a factor of ten by the year 2001. The 1992-2002 data collected through the mechanism of the National Software Quality Experiment and its Software Inspection Lab strongly suggest that the hypothesis is not proved.

- Has the stated objective to reduce software problems been achieved? The short answer to the question is that the objective has not been met. The most compelling evidence is found in the 'Year' data under NSQE Results by Analysis Bin. The metrics data for each year meander in a limited control range, showing no signs of breaking out.
- Both the process metrics and the defect type percents are somewhat stable. To achieve a factor of ten reduction over the time period, there would need to be a breakout, a significant breakout. In the absence of a breakout, it is asserted that the improvement goal was not met.
- When software inspections data for an organization is collected, it generally falls into the patterns revealed by the NSQE once there are a few dozen software inspections. For example, if an organization is characterized as a level 2, defense industry sector, and embedded software type, the NSQE data with just those characteristics are selected as the benchmark.



In closing it needs to be restated that the data does not suggest progress towards the Year 2000 goal to reduce software problems by a factor of ten. Hunting for defects in software is a target rich opportunity. The harder the project looks for errors, the more it finds. The way to look harder is to reduce the volume of product inspected in each session and to increase preparation effort. In doing this, there continues to be a favorable return on investment for software inspections; savings exceed costs by 4 to 1.

The data suggests that increased software process maturity results in increased defect detection, with the expected result of lower defect leakage in the field. At level 1 the project lacks a shared vision for a standard of excellence for software engineering products. At level 2 attention is paid to establishing a standard of expectation, a standard of excellence, and so more defects are identified. At level 3 the standard is set and the well defined, fined grained processes for software product engineering are in place and in

practice with software inspections operating as the exit criteria for each activity of the life cycle. Many commercial enterprises possess a large volume of legacy software and find themselves anchored at level 1.

The data also suggests that defect density decreases with increasing program size... up to a point. All programs contain a beginning, an end, and a context for operation within the larger system. Starting, finishing, and fitting in are all more error prone than the body of the program which gives it size.

In addition the data suggests that the organization's neglect of its software process exceeds the poor workmanship of individual programmers as the source of errors. Documentation and standards defect types account for nearly two-thirds of all defects, and these are the responsibility of the organization and its process.

Software products are not well connected to the requirements or business case that inspired their creation. Much of the documentation type defect detection results from the lack of traceability from the code to the design to the specification to the requirements.

Perhaps the Government should consider commercial practices. In addition, DOD Industry may be limited by the Government acquisition management practices it must adhere to. Consumers of embedded software products hold producers to higher standards than producers of organic software products hold themselves.

SPONSORING THE NATIONAL EXPERIMENT

In order to meet the objective to reduce software problem rates by a factor of ten by the year 2000, there must be steady improvement year by year. The 1992-2002 defect profiles collected in the National Software Quality Experiment provide the benchmark measurements to track progress towards the objective. In order to sustain the experiment, organizations must participate and sponsor this activity. Three steps are needed to sponsor the National Software Quality Experiment in an organization:

1. Senior management publicly communicates a buy-in for the experiment and its benefits and advocates its use in the organization and beyond.

2. Program managers install the Software Inspections Process, authorize training for project personnel, and initiate the practice of fact-based software management.

3. Software Inspection Lab results are disseminated among practitioners, program managers, and senior management.

The results of the Inspection Lab provide the energy to install the process and to improve software products. Software inspections are viewed as competency enhancing by practitioners, and there is little resistance once senior management makes a public commitment to sponsor the change. Despite this, only 22% of the organizations conducting the SEI's Software Process Assessment (SPA) are credited with practicing peer reviews [Kitson 92].

An effective technology transition strategy for installing the software inspections process is to first prototype the Inspection Lab and collect results. The participant feedback from the prototype sessions usually rates the willingness and ability to conduct inspections on the project at 4.0 to 4.4 on a 5.0 scale. The actual inspection results from the lab are fed back

to practitioners, project management, and senior management. These results and discussions about them with senior management provide the proper environment to obtain senior management commitment, sponsorship, and funding for training conduct and student labor. The planning of production training may involve a make-buy decision on the training mechanism depending on the number of students and the organization approach to in-house training.

The software inspections training needed by the organization includes training for practitioners, orientation for project managers, and briefing for senior management. For practitioners the training must span the behavior, skills, and knowledge needed to carry out the defined roles in the Software Inspections Process. For project managers, the orientation should overview what the practitioners are taught along with specific responsibilities associated with using inspection results to reduce defect leakage. The senior management presentation should highlight the oversight opportunities associated with defect prevention.

Once the organization is trained, the monitoring of software inspection results at the monthly program review can begin. This should focus on the effectiveness of defect detection and correction activities and the efficiency of the process. In addition the oversight of organization inspection results at the senior management quarterly review should focus on patterns of neglect, defect prevention, and the selection of improvement opportunities. The results being communicated to management should also be fed back to practitioners on a regularly scheduled periodic basis. In addition, reporting Inspection Lab results to the National Software Quality Experiment results in feedback from the national database useful in obtaining an industrial calibration of the organization's software product quality.

FIELD MEASUREMENT LESSONS

In conducting the National Software Quality Experiment, valuable lessons in field measurement are being learned. These lessons are forming the prescription for obtaining lasting value in measurement:

1. Measurement must be aligned with business and performance needs. These activities must be built into the normal operation of the organization.
   - To do this, the goals to be met and questions to be answered in management, engineering, and operations must precede the collection of data.
2. Metrics must be carefully pinpointed and rigorously defined. Extraordinary steps must be applied to obtain consistency and uniformity.
   - Without a well defined process for data collection and analysis, the variance in the measurement process itself impacts the accuracy of results.
3. Attention must be paid to the confidentiality of results. The opportunity for improvement is increased when the measured results are made more widely available.
   - However, individuals and groups naturally resist having their shortcomings made public. If ignored, this resistance will defeat the measurement program.
   - The organization must strike a balance between public and private data.

NEXT STEPS

The National Software Quality Experiment is a demonstrated mechanism for collecting uniform and consistent measurements of software product quality. It provides the vantage point for software product quality and the field experience in measurement needed to jump start the practice of fact-based software management.

As the centerpiece of the experiment, the Software Inspection Labs have been installed

in software factories around the country. The National Software Quality Experiment collects, organizes, and packages core samples of software product quality. These measurements are increasing the understanding of the state of the practice and how to measure it. Based on these results and the identification of common problems, organizations are challenged to:

1. Establish a tradition of baseline management with fine-grained traceability among requirements, specification, design, code, and test artifacts
2. Establish a tradition of modern software engineering design and coding practices
3. Establish a tradition of uniform recording style and its enforcement
4. Establish a tradition of visible evolution of modern domain architectures and product lines

The organization that wishes to calibrate its software inspection results with the National Software Quality Experiment is invited to conduct an NSQE assessment at: http://members.aol.com/ONeillDon/nsqe-assessment.html

The usefulness and success of the National Software Quality Experiment depends on sustaining a continuous stream of core samples. Organizations from industry, government and military, and commercial enterprise are invited to participate and enrich this national database resource.

The prediction of critical faults and failures using the core samples of defect data from the National Software Quality Experiment remains a future challenge now underway.

Fast Forward

In closing, it must be said that the future may be different than we expect. The DOD Technology Strategy focused on inward looking goals set in an earlier time:

1. Improving productivity by a factor of two
2. Reducing software problems by a factor ten

As software development is becoming better understood, there is a growing recognition and acceptance that fielding software systems involves a process of experimentation with hypotheses, alternatives, measurements, analysis, and iteration. These systems are less deterministic in requirements, capabilities, and features and more non-deterministic in execution paths, platforms, and scalability.

Commercial organizations today focus on the outward looking goals needed to achieve global competitiveness. These organizations are focused on controlling personnel resources, managing customer relationships, innovating new products and features, and managing the risk of event threats.

Future Directions

In reasoning about future trends of Peer Reviews, the topics considered include increasing rate of software problems, improving the practice of defect prevention and prediction, extending the practice of Peer Reviews to systems engineering, understanding the process of experimentation in software development, exploiting technology in automating the Peer Reviews, and adapting to changes in business environment.

Software problem rates are not decreasing. The results of the National Software Quality Experiment 2000 show no systematic improvement towards fulfilling the national goal of a

ten times reduction in software problems set in 1992. The defect rates continue to range from 1 to 10 defects per thousand lines of source code.

The factors that may be contributing to defect rates include:

- The emphasis on quicker, better, and cheaper
- The trend towards code and upload practice as the life cycle model
- The preoccupation on improving software process maturity and mastering the management track practices of the Software Engineering Institute's Capability Maturity Model for Software level 2, an obstacle to many
- The downsizing of middle management and senior technical staff known to hold the line on product quality.

While software inspections have been in use for over twenty-five years, defect prevention remains an immature practice. Defect prevention is a CMM level 5 key process area, and some organizations have achieved level 5. As more organizations seek to adopt the practice of defect prevention, its benefits and methods may become more well understood stimulating others to adopt the practice.

Similarly, defect prediction remains an underdeveloped practice. If software defects, faults, and failures can be predicted, perhaps they can be detected, controlled, and prevented. Model based techniques calibrated with defect detection early in the life cycle to predict defect rates in later life cycle activities have been demonstrated [Gaffney 97]. More modest efforts utilizing software inspections data to estimate the number of defects remaining to be found in testing are being applied on the project [Harding 98]. However, there is insufficient defect, fault, and failure data available from the nation's factory floor [O'Neill 99]. In addition there is insufficient process, method, and tooling to combine defect data obtained through software inspections practice, software fault data obtained through software product test and use, and software failure data obtained through software system operation into predictions of trustworthy software system operation [Wallace 97].

While the benefits and usage of software inspections on code artifacts is well known, there is increasing interest in extending software inspections to all phases of the life cycle including requirements, specifications, design, code, and test artifacts. The CMMI project with the inclusion of Peer Reviews in the Product Verification process area extends Peer Reviews to both systems engineering and software artifacts.

To achieve the best possible practice of software inspections, both managers and technical practitioners are encouraged to decriminalize defects. People make mistakes sometimes, yet software must be bit perfect. When managers and technical participants view with alarm the defects detected in software inspections, it produces a negative impact. On the other hand, when managers genuinely decriminalize defects and use their detection as a means to prevent their recurrence, it produces a positive result. During a software inspection session, the litmus test for decriminalization lies in the reaction of participants when a major defect is detected. Does the group say 'good catch' or 'bummer'?

With the growing recognition that fielding software involves a process of experimentation and with the increasing pressures of competition and demand for innovation, software walkthroughs may experience increasing usage. Software walkthroughs encourage and support the learning essential to experimentation. In favoring the group interaction needed to achieve consensus, software walkthroughs may contribute to increased innovation in software products.

There is interest in automating software inspections. The value of programming languages with strong typing, robust compilers, static analyzers and traceability tools, and complexity metrics [McCabe 94] is recognized. However, software inspections is a reasoning activity and will remain essentially a human activity. The use of information technology innovations to support the logistics of preparation, scheduling, conduct, and results repository operations are sources for improved industry practice.

Software inspections are being conducted effectively using groupware tools. However, where global software development teams conducting geographically dispersed inspection sessions are using 'follow the sun' software development tactics, software inspection participants may be separated by both geography and time zones complicating the logistics of their application [Carmel 99].

Software inspections usage is increasing in e-commerce applications where code and upload is the typical life cycle practice. In an environment of rapid change and frequent releases, there is an absence of robust testing and sometimes even regression testing.

BIBLIOGRAPHY

[AHDEL 76] "The American Heritage Dictionary of the English Language", Houghton Company, 1976

[Basili 96] Basili, V. R., Green, S., Laitenberger, O., Lanubile, F., Shull, F., Sorumgard, S., Zelkowitz, M.V., "The Empirical Investigation of Perspective-Based Reading"; Empirical Software Engineering: An International Journal, Vol. 1, No. 2, 1996

[Basili/Boehm 01] Basili, Vic, Barry Boehm, "Top Ten Defect Reduction List", IEEE Software, January 2001

[Basili 02] Basili, V.R., Caldiera, G., and Rombach, H.D., "The Experience Factory" Encyclopedia of Software Engineering. John Wiley & Sons, Inc., Volume 2, pp. 929-945, 2002

[Business Week 99] "Software Hell: Is There a Way Out?", Business Week, pp.104-118, 6 December 1999

[DOD STS 91] Department of Defense Software Technology Strategy, draft prepared for the Director of Defense Research and Engineering [DDR&E], December 1991

[Ebenau 94] Ebenau, Robert G. and Susan H. Strauss, "Software Inspection Process", McGraw-Hill, Inc., 1994, pp.236-240

[Fagan 76] Fagan, M., "Design and Code Inspections to Reduce Errors in Program Development", IBM Systems Journal, 15, 3, 1976, pp.182-211

[Fagan 87] Fagan, M., "Advances in Software Inspections", IEEE Transactions on Software Engineering, 12, 7, 1987

[Florac 92] Florac, William B., "Software Quality Measurement: A Framework for Counting Problems and Defects", CMU/SEI-92-TR-22, September 1992

[Freedman 90] Freedman, D.P., G.M. Weinberg, "Handbook of Walkthroughs, Inspections, and Technical Reviews", Dorset House Publishing Co., Inc., 1990, pp. 89-161

[Gaffney 97] Gaffney, John E., "Software Defect Estimation, Prediction, and the CMM", Metrics '97 Conference, 1997

[GCN 99] Olsen, Florence, "Is It Possible to Build Software Free Defects", Government Computing News, 8 March 1999

[Gilb 93] Gilb, Tom and Dorothy Graham, "Software Inspection", Addison Wesley Longman Limited, 1993, pp. 40-136

[Humphrey 89] Humphrey, Watts S., "Managing the Software Process", Addison-Wesley Publishing Company, Inc., 1989, pp, 171-190, 463-486

[IBM 99] "Orthogonal Defect Classification", IBM Research, Center for Software Engineering, http://www.research.ibm.com/softeng/ODC, 1999

[Joyce 89] Joyce, Edward J., "Is Error-Free Software Achievable?", Datamation, 15 February 1989

[Juristo 98] Juristo, Natalia, "An Adaptation of Experimental Design to Empirical Validation of Software Engineering Theories", Twenty -Third Annual Software Engineering Workshop, Greenbelt, Maryland, 2-3 December 1998

[Kitson 92] Kitson, David H. and Stephen M. Masters, "An Analysis of SEI Software Process Assessment Results: 1987-1991", 15th ICSE

[Linger 79] Linger, R.C., H.D. Mills, B.I. Witt, "Structured Programming: Theory and Practice", Addison-Wesley Publishing Company, Inc., 1979, pp. 147-212

[Linger 98] Linger, Richard C., “Survivability Assessment”, SEI Symposium, Pittsburgh, 1998

[Lindner 91] Lindner, Richard J., "Software Development at a Baldrige Winner, IBM Rochester", TQM for Software Conference, sponsored by The National Institute for Software Quality and Productivity, November 1991

[NIST 99] Wallace, Dolores, “Error, Fault, and Failure Data Collection and Analysis”, NIST Information Technology Laboratory (ITL), hissant.ncsl.nist.gov/eff, 1999

[Nolan 96] Nolan, Thomas W. and Lloyd P. Provost, "Understanding Variation", IEEE Engineering Management Review, pp. 65-74, Spring 1996

[O'Neill 88] O'Neill, Don and Albert L. Ingram, "Software Inspections Tutorial", Software Engineering Institute Technical Review 1988, pp. 92-120

[O'Neill 89] O'Neill, Don, "Software Inspections Course and Lab", Training Offering for Practitioners, Software Engineering Institute, 1989

[O'Neill 92] O'Neill, Don, "Software Inspections: More Than a Hunt for Errors", Cross Talk, Issue 30, January 1992

[O'Neill 94] O'Neill, Don, "National Software Quality Experiment", International Conference on Software Quality, Washington DC, 1994

[O'Neill 95a] O'Neill, Don, "Software Inspections Management Workshop", Training Offering for Managers, 1995

[O'Neill 95b] O'Neill, Don, "Software Inspections Process: Executive Presentation", Training Offering for Executives, 1995

[O'Neill 95,96] O'Neill, Don, "National Software Quality Experiment: Results 1992-1995", Software Technology Conference, Salt lake City, 1995 and 1996

[O'Neill 96] O’Neill, Don, “Peer Reviews Key Process Area Handbook”, Don O'Neill Consulting, Gaithersburg, Maryland, 1996

[O'Neill 97a] O'Neill, Don, "Issues in Software Inspection", IEEE Software,

Vol .14 No 1., January 1997, pp. 18-19

[O'Neill 97b] O'Neill, Don, “Setting Up a Software Inspection Program”, CrossTalk, The Journal of Defense Software Engineering, Vol. 10 No. 2, February 1997

[O'Neill 97c] O'Neill, Don, "National Software Quality Experiment: A Lesson in Measurement 1992-1996", Quality Week Conference, San Francisco, May 1997 and Quality Week Europe Conference, Brussels, November 1997, pp. 1-25

[O'Neill 98a] O'Neill, Don, “Software Inspections and the Year 2000 Problem”, CrossTalk, The Journal of Defense Software Engineering, Vol. 11 No. 1, January 1998

[O'Neill 98b] O'Neill, Don, “National Software Quality Experiment: A Lesson in Measurement 1992-1997”, CrossTalk, The Journal of Defense Software Engineering, Vol. 11 No. 12, Web Addition, December 1998

[O'Neill 98c] O'Neill, Don, “National Software Quality Experiment: A Lesson in Measurement 1992-1997”,Twenty-Third Annual Software Engineering Workshop, NASA Goddard Space Flight Center, Greenbelt, Maryland, 2-3 December 1998

[O'Neill 99a] O'Neill, Don, “National Software Quality Experiment: A Lesson in Measurement 1992-1997”,First Annual International Software Assurance Certification Conference, Chantilly, Virginia, 1 March 1999

[O'Neill 99b] O'Neill, Don, "Software is Research... A Process of Experimentation", The Competitor, Vol3 No 1, 1999

[O'Neill 00] O'Neill, Don, “National Software Quality Experiment: A Lesson in Measurement 1992-1999”, Software Technology Conference 2000, Salt Lake City, May 2000

[O'Neill 01] “Return on Investment Using Software Inspections”, 11th International Conference on Software Quality, 22-24 October 2001, Pittsburgh, Pennsylvania http://members.aol.com/ONeillDon/roi-essay.html

[O'Neill 02] “Peer Reviews”, Encyclopedia of Software Engineering Second Edition, John Wiley &Sons, Inc., January 2002 http://members.aol.com/ONeillDon2/peer-reviews.html

[Paulk 95] Paulk, Mark C., “The Capability Maturity Model: Guidelines for Improving the Software Process”, Addison-Wesley Publishing Company, 1995, pp. 270-276

[Prowell 99] Prowell, Stacy J., Carmen J.Trammell, Richard C. Linger, Jesse H. Poore, “Cleanroom Software Engineering: Technology and Process”, Addison-Wesley Longman, Inc., 1999, page 17

[Radice 02] Radice, Ronald A., “High Quality Low Cost Software Inspections”, Paradoxicon Publishing, Andover, Massachusetts, 2002

[SEI 97] "Practical Software Measurement: Measuring for Process Management and Improvement", Software Engineering Institute, CMU/SEI-97-H8-003,

1997

[SEI 00.1] “Personal Software Process”, CMU/SEI-2000-TR-022

[SEI 00.2] “Team Software Process”, CMU/SEI-2000-TR-023

[SPC 92] "Statistical Process Control (SPC), Reference Manual", Chrysler Corporation, Ford Motor Company, and General Motors Corporation, 1992

[Tichy 98] Tichy, Walter F., “Should Computer Scientists Experiment More?”, Computer, 31(5), 32-40, May 1998

[Van Verth 92] Van Verth, Patricia B., "A Concept Study for a National Software Engineering Database", CMU/SEI-92-TR-23, July 1992

[Voas 98] Voas, Jeffrey M. and Gary McGraw, “Software Fault Injection”, John Wiley & Sons, Inc., New York, 1998

[Wallace 96] Wallace, Dolores R., Marvin V. Zelkowitz, "Experimental Models for Software Diagnostics", NIST Computer System Laboratory, NISTIR 5889, September 1996

[Wallace 97] Wallace, Dolores R., Laura M. Ippolito, and Herbert Hecht, "Error, Fault, and Failure Data Collection and Analysis", http://hissa.ncsl.nist.gov, Quality Week, San Francisco, May 1997

[Wang 82] Wang, Alan and Nick Kirschner, "Software Inspections at Gaithersburg", IBM FSD Software Engineering Exchange, Vol. 4, No. 2, October 1982

[Weinberg 84] Weinberg, G. M. and Weiss, D. P., "Reviews, Walkthroughs, and Inspections", IEEE Transactions on Software Engineering, No. 1, Vol. SE-10, January 1984, 68-72

[Weller 93] Weller, Edward F., "Lessons From Three Years of Inspection Data", IEEE Software Magazine, September 1993

[Zelkowitz 98] Zelkowitz, Marvin V. and Dolores. Wallace, “Experimental Models for Validating Technology”, Computer, 31(5), 23-31, May 1998

Author: Don O'Neill

Don O'Neill is a seasoned software engineering manager and technologist currently serving as an independent consultant. Following his twenty-seven year career with IBM's Federal Systems Division, Mr. O'Neill completed a three year residency at Carnegie Mellon University's Software Engineering Institute (SEI) under IBM's Technical Academic Career Program. There he developed a blueprint for charting software engineering evolution in the organization including the training architecture and change management strategy needed to transition skills into practice.

As an independent consultant, Mr. O'Neill conducts defined programs for managing strategic software improvement. These include implementing an organizational Software Inspections Process, implementing Software Risk Management, and conducting Global Software Competitiveness Assessments. Each of these programs includes the necessary practitioner and management training.

In his IBM career, Mr. O'Neill completed assignments in management, technical performance, and marketing in a broad range of applications including space systems, submarine systems, military command and control systems, communications systems, and management decision support systems. He was awarded IBM's Outstanding Contribution Award three times:

1. Software Development Manager for the Global Positioning Ground Segment (500,000 source lines of code) and a team of 70 software engineers within a $150M fixed price program.

2. Manager of the FSD Software Engineering Department responsible for the origination of division software engineering strategies, the preparation of software management and engineering practices, and the coordination of these practices throughout the division's software practitioners and managers.

3. Manager of Data Processing for the Trident Submarine Command and Control System Engineering and Integration Project responsible for architecture selections and software development planning (1.2M source lines of code).

Mr. O'Neill served on the Executive Board of the IEEE Software Engineering Technical Committee and as a Distinguished Visitor of the IEEE. He is a founding member of the National Software Council and the Washington DC Software Process Improvement Network (SPIN). He is an active speaker on software engineering topics and has served as the Program Chairman and Program Committee member for several conferences. He has numerous publications to his credit. Mr. O'Neill has a Bachelor of Science degree in mathematics from Dickinson College in Carlisle, Pennsylvania.

Contact Information

Don O'Neill
Independent Consultant
9305 Kobe Way
Montgomery Village, Maryland 20886

Phone: (301) 990-0377

email: ONeillDon@aol.com
http://members.aol.com/ONeillDon/index.html

Don O'Neill Photo oneill.gif
http://members.aol.com/ONeillDon/oneill.gif

word count: 18,350

National Software Quality Experiment
Prediction Statistics (1992-1998)
Inspection Lab Operations

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
Total	12.164243012	74.872700149	2.3492688762	12.110825679	901.04840266	4.967094703	0.66037876528	4.5115737255
Average	13.947873927	116.11004934	3.7682694314	16.637492552	914.9310785	5.5364536887	0.71302383661	4.8310625288
Max	63.636363636	816.11004934	35.142857143	50.704225352	9122.2304833	20	2.1341463415	15.311850312
Min	2.7272727273	14.75	0	0	0	1.7777777778	0.125	1.1089494163
Range	60.909090909	801.36004934	35.142857143	50.704225352	9122.2304833	18.222222222	2.0091463415	14.202900896
STDEV	8.6390979648	120.98438956	5.0650642888	12.188842693	1184.5835154	2.9295997363	0.36226981654	2.5193105155
VAR	74.634013646	14637.222518	25.65487625	148.56788619	1403238.105	8.582554615	0.13123941997	6.3469254733
Detection	-0.00377253803	-0.06895247993	-0.00819496634	-0.1963330474	-0.34017122235	0.1154345067	0.10951905497	0.09109737268
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Process Maturity								
Level 1	11.175509687	73.851882845	1.7655236809	9.9017068699	1101.0215535	4.8064516129	0.59816320998	4.4421166023
Average	14.010635561	123.29234127	3.1726196506	16.29014321	1127.8640928	5.6317340991	0.69808431539	4.8024798577
Max	63.636363636	823.29234127	16.337644656	50.704225352	9122.2304833	20	2.1341463415	15.311850312
Min	2.7272727273	14.75	0	0	0	1.7777777778	0.125	1.1089494163
Range	60.909090909	808.54234127	16.337644656	50.704225352	9122.2304833	18.222222222	2.0091463415	14.202900896
STDEV	10.00434655	133.93899317	3.2794464231	11.641784235	1435.6811149	3.3380190385	0.36390672576	2.6211868097
VAR	100.0869499	17939.653891	10.754768842	135.53114018	2061180.2637	11.142371102	0.13242810505	6.8706202912
Detection	-0.01486114659	-0.19162410828	-0.16386035176	-0.30756762992	-0.58369345382	0.06873593398	-0.02573215222	0.07142113379
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Level 2	14.290993704	79.824159021	4.7046298161	21.573677093	540.86764801	5.4279346211	0.84632968031	4.6808734821
Average	14.275587186	99.054409432	4.7585788045	16.741365981	553.31445228	5.5004789013	0.79593682544	4.7682801733
Max	32.440433213	382.05440943	35.142857143	50.571428571	2494.75	14.235294118	1.7519305019	10.849176172
Min	5.8653846154	20.842105263	0	0	0	2.696969697		

0.37841191067 2.378768021
Range 26.575048598 361.21230417 35.142857143 50.571428571 2494.75 11.538324421
1.3735185912 8.470408151
STDEV 6.3433579768 59.648923719 7.9057172863 14.052700107 556.06549694 2.4381894269
0.40015120485 2.216085508
VAR 40.238190421 3557.9941009 62.500365811 197.47838031 309208.83689 5.9447676814
0.16012098674 4.9110349789
Detection 0.05001258985 0.1949123702 0.6199012843 0.28014486871 0.09844776598
0.24939959916 0.51376017458 0.13905764365
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556
0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75
0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33333333 4.6662274923 26.64824319 1115.0943396 7
0.9056122449 6.4295874822

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
Level 3	12.211352657	62.413580247	4.0352712599	16.589448513	513.04792332	4.7045454545	0.53839190628	4.4110622685
Average	12.643946477	122.77273917	4.4276199385	18.375186043	700.94235529	5.0811678207	0.56363249285	5.1778534751
Max	23.892307692	640.43940584	10.297215071	40.48677744	1359.2465753	7.8571428571	0.90081206497	10.909090909
Min	5.1923076923	22.5	0.44326241135	5.291005291	193.49433962	3.8235294118	0.42965246637	1.6295392127
Range	18.7	617.93940584	9.8539526597	35.195772149	1165.7522357	4.0336134453	0.4711595986	9.2795516963
STDEV	6.0313665283	176.6645739	3.2653433921	11.213191511	387.28660089	1.5565258493	0.15850958839	3.0592851704
VAR	36.377382198	31210.37167	10.662467468	125.73566386	149990.91123	2.4227727196	0.02512528961	9.3592257538
Detection	-0.13910949113	-0.06310761079	0.44140566931	0.02917666228	0.13232592116	0.03911205074	-0.15566976895	0.0626238721
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Programming Language								
Old Style	12.877863731	81.129334583	1.8394997664	9.7491763627	1116.3017609	4.6326671261	0.69414151458	4.3331613599
Average	14.450789198	104.90619265	3.8321678418	13.924896333	1082.6246124	4.989121986	0.7655066756	4.8463881108
Max	32.440433213	622.57285932	35.142857143	50.571428571	9122.2304833	14.235294118	1.7519305019	10.849176172
Min	6.8779527559	20.842105263	0	0	0	2.4838709677	0.39451476793	1.4591238358
Range	25.562480457	601.73075406	35.142857143	50.571428571	9122.2304833	11.75142315	1.357415734	9.3900523362
STDEV	5.6353888182	94.314214085	6.3467045348	11.790318061	1612.6498262	2.3791420272	0.35567302585	2.2315378781
VAR	31.757607132	8895.1709785	40.280658452	139.01159997	2600639.4618	5.6603167854	0.12650330132	4.9797613012
Detection	0.06419003457	0.01958607085	-0.14413339563	-0.31524791729	-0.60230130291	0.01821718783	0.18291633604	0.04055562605
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Modern	11.31612447	67.800847458	3.4207608294	17.074814648	641.17565056	5.4332372719	0.61961741016	4.7368798565
Average	13.473696672	126.67368564	3.7080223587	19.195083273	756.82003224	6.0525092942	0.66354001699	4.8166126943
Max	63.636363636	826.67368564	16.337644656	50.704225352	2494.75	20	2.1341463415	15.311850312
Min	2.7272727273	14.75	0	0	0	1.7777777778	0.125	1.1089494163
Range	60.909090909	811.92368564	16.337644656	50.704225352	2494.75	18.222222222	2.0091463415	14.202900896

STDEV 10.803076168 142.25085564 3.5547559659 12.167837105 514.31063649 3.3197013408 0.36655534873 2.7964083389
VAR 116.70645468 20235.30593 12.636289977 148.05625981 264515.43081 11.020416992 0.13436282368 7.8198995979
Detection -0.08059040346 -0.17417810551 0.27753622117 0.05361604473 -0.02370477557 0.2509410674 0.02090741339 0.15492347211
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556 0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75 0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33333333 4.6662274923 26.64824319 1115.0943396 7 0.9056122449 6.4295874822

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
Product Type								
Embedded	13.142696449	73.681981982	4.3583421024	20.075858711	542.36166815	5.3097269625	0.7257054126	4.6593345571
Average	14.621852651	118.76829566	4.5704015179	18.639566375	628.31277051	5.8184943814	0.75202201633	4.9965057194
Max	63.636363636	818.76829566	35.142857143	50.704225352	2494.75	20	2.1341463415	15.311850312
Min	2.8736263736	14.75	0	0	0	1.7777777778	0.24212962963	1.1089494163
Range	60.762737262	804.01829566	35.142857143	50.704225352	2494.75	18.222222222	1.8920167119	14.202900896
STDEV	9.9359064055	128.67565638	6.0281638248	13.594837168	529.75048826	3.3551214522	0.40153531935	2.8342820125
VAR	98.722236099	16557.424546	36.338759099	184.81959762	280635.57981	11.256839959	0.16123061269	8.0331545262
Detection	-0.01670669148	0.05244372812	0.52755789397	0.20472601768	0.09662838615	0.21503690769	0.25153350565	0.13295596517
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Organic	11.174979691	76.339622642	1.4983528097	8.7373425773	1251.6346354	4.6628787879	0.59652744051	4.3554642939
Average	12.859139066	111.81595912	2.4725175993	13.4033733	1377.9298837	5.0808494929	0.65002677706	4.5638081439
Max	30.961538462	629.48262579	12.778235779	34.784015531	9122.2304833	10.142857143	1.2790697674	9.7571743929
Min	2.7272727273	20	0.31786395423	0.75798327547	292.93838863	2.6666666667	0.125	1.6295392127
Range	28.234265735	609.48262579	12.460371825	34.026032256	8829.2920947	7.4761904763	1.1540697674	8.1276351802
STDEV	6.009891557	109.7384708	2.516432598	8.8003376734	1714.4914901	2.047073952	0.28393841683	1.9274648915
VAR	36.118796527	12042.531974	6.3324330204	77.445943166	2939481.0695	4.190511765	0.08062102455	3.7151209081
Detection	0.01216187966	-0.1916898646	-0.23510591741	-0.36619624485	-0.76710624523	0.02699964764	-0.02928817282	0.04687373764
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Organization Type								
Commercial	10.850646805	68.458367683	1.9715242337	10.467119592	1048.2323878	5.0183606557	0.58948803135	4.6511067045
Average	11.671345026	86.915546053	3.5619040162	15.429429927	844.64441889	5.6317269609	0.56792893688	5.0594786598
Max	32.625	369.91554605	16.337644656	35.568010347	2494.75	11.375	1.1908284024	15.311850312
Min	2.7272727273	14.75	0	0	0	1.7777777778	0.125	2.0725388601
Range	29.897727273	355.16554605	16.337644656	35.568010347	2494.75	9.5972222222	1.0658284024	13.239311452

STDEV 6.4659939308 60.44355121 3.7650975561 9.5672023126 517.94645323 2.2914049475 0.20537799276 2.7343175175
VAR 41.809077514 3653.4228829 14.175959607 91.531360091 268268.52841 5.2505366334 0.04218011991 7.4764922867
Detection -0.0734481025 -0.23192967678 -0.10892687102 -0.27909770434 -0.51940814663 0.13033739992 -0.0445912362 0.13062512876
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556 0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75 0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33333333 4.6662274923 26.64824319 1115.0943396 7 0.9056122449 6.4295874822

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
DOD Ind.	14.338060384	97.641744548	3.2312291757	18.773340849	508.84241079	4.5352697095	0.66892180297	3.6476169772
Average	14.863279892	137.33311323	4.141917977	23.598323034	560.40319419	5.5548114633	0.74676373552	3.7532410689
Max	26.176470588	621.66644656	10.452961672	50.704225352	1549.4230769	20	1.3894230769	6.4295874822
Min	5.8653846154	48.636363636	0.54406964091	8.5106382979	153.41389728	2.4838709677	0.33484162896	1.4591238358
Range	20.311085973	573.03008293	9.9088920311	42.193587054	1396.0091796	17.516129032	1.0545814479	4.9704636464
STDEV	4.5834869399	113.04391679	2.9822799517	12.291898719	366.31246886	4.1180673871	0.34112482584	1.2296618753
VAR	21.008352528	12778.927123	8.8939937106	151.09077412	134184.82484	16.958479005	0.11636614681	1.5120683275
Detection	0.24355577393	0.20075190872	0.22699444685	0.13914102968	0.13744728766	-0.01009601467	0.12809087602	-0.15364958153
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Gov	14.252067743	75.86163522	3.3732179226	14.581918986	842.55014896	5.2350515464	0.89836146971	4.986529961
Average	16.962980423	146.29068191	3.7815151162	12.444012857	1350.5177701	5.3595680636	0.92720586345	5.4107282459
Max	63.636363636	846.29068191	35.142857143	50.571428571	9122.2304833	14.235294118	2.1341463415	10.849176172
Min	6.7692307692	20.842105263	0	0	0	1.8333333333	0.39451476793	1.1089494163
Range	56.867132867	825.44857665	35.142857143	50.571428571	9122.2304833	12.401960785	1.7396315736	9.7402267557
STDEV	12.943143303	184.20054018	7.9350786916	13.933569839	2077.0421635	2.7892470561	0.474866108	2.803539109
VAR	167.52495857	33929.839004	62.965473841	194.14436847	4314104.1491	7.7798991398	0.22549782052	7.8598315355
Detection	0.00696975038	0.19008284655	0.2648581127	-0.07190740251	-0.26893353259	0.1933289379	0.62687276024	0.22564588131
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33333333	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Annual								
1992	15.485466914	107.00854701	3.2592344977	19.262911583	713.44153693	5.575862069	1.0367671414	4.2555700114
Average	16.302940833	102.0739189	3.2586981498	18.797537927	972.58352067	5.9188683646	1.0636600607	4.955097305
Max	32.440433213	285.461674	4.6662274923	27.94246816	2127.8688525	9.6	1.7006056018	7.1519147857
Min	6.8779527559	64.703703704	1.3867488444	11.658962506	462.53068134	3.6956521739	0.73523294509	2.8128941836
Range	25.562480457	220.7579703	3.2794786479	16.283505654	1665.3381712	5.9043478261	0.96537265671	4.3390206021

STDEV 10.516182291 43.67251233 1.1775536233 6.1769387261 636.95845679 2.0996720415 0.38608602389 1.7325400556
VAR 110.59008999 1907.2883332 1.3866325358 38.154572026 405716.07568 4.408622682 0.14906241784 3.0016950442
Detection 0.39875247126 0.34311003896 0.23446253273 0.16379212401 -0.11170833924 0.29240176423 0.9277546553 0.01857507405
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556 0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75 0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33 4.6662274923 26.64824319 1115.0943396 7 0.9056122449 6.4295874822

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
1993	11.561200924	76.427480916	3.9976807348	22.429735421	651.6871064	6.0985915493	0.82963208485	5.2010869565
Average	13.059047899	114.00316138	17202.501047	100006.62781	791.69340384	6.5777800631	0.91852712853	4.9774994999
Max	19.051724138	397.00316138	54000	290000	2494.75	14.235294118	1.7519305019	6.4873902745
Min	7.9680232558	50.759259259	0.20042088386	4.2589437819	0.05410279531	2.696969697	0.53262518968	2.378768021
Range	11.083700882	346.24390212	53999.799579	289995.74106	2494.6958972	11.538324421	1.2193053122	4.1086222535
STDEV	4.2419585972	97.783905834	24801.057948	139815.38939	1035.3030469	4.5151263653	0.51846225427	1.7031350109
VAR	17.99421274	9561.6922402	615092475.35	19548343110	1071852.399	20.386366095	0.2688031091	2.9006688655
Detection	0.01514652429	-0.14377159754	0.43138152929	0.32324951766	-0.03650535991	0.44435800852	0.47746105403	0.28642689986
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822

1994	13.877348066	84.288590604	4.6376992032	23.530876494	471.31540342	5.5521472393	0.76766503667	4.3507728483
Average	14.66112834	86.345830546	3.5346219018	16.85664016	428.78161826	5.1337189127	0.72069149965	4.1603221553
Max	18.368421053	202.67916388	8.6225479629	35.568010347	784.12844037	7.3214285714	0.91984304933	5.0367764631
Min	12.029268293	61.65	0	0	0.2	3.7307692308	0.57798165138	2.9021372328
Range	6.33915276	141.02916388	8.6225479629	35.568010347	783.92844037	3.5906593406	0.34186139795	2.1346392303
STDEV	2.4291778739	20.793087336	3.2676301775	14.180684433	296.1144664	1.5490489739	0.13973156534	0.92098399837
VAR	5.900905143	432.35248097	10.677406977	201.09181099	87683.777213	2.3995527236	0.01952491035	0.84821152526
Detection	0.11375552689	0.1435915715	0.60205312085	0.37869468751	0.1831467255	0.28550791839	0.34275007973	0.04554471623
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822

1995	12.242280946	76.942307692	1.4121992287	7.4634235462	1451.5033148	4.400244798	0.65715522905	4.4174333006
Average	12.625028943	108.54697955	5.933867828	16.754776474	1263.4566276	4.972037282	0.74108932044	5.3330746865
Max	16.698795181	455.04697955	35.142857143	50.571428571	3060	8.8333333333	1.3894230769	10.849176172
Min	8.7894736842	24.62601626	0.6197981229	1.4064697609	206.28683694	2.9	0.39451476793	1.9736842105
Range	7.9093214968	430.42096329	34.52305902	49.16495881	2853.7131631	5.9333333333	0.99490830897	8.8754919615
STDEV	2.3114754864	97.690619527	11.156209021	15.509910309	895.70837681	1.8643587491		

0.31246005055 3.0232112522
VAR 5.3429189243 9543.4571436 124.46099972 240.5573178 802293.49629 3.4758335454
0.09763128319 9.1398062751
Detection 0.02160446177 -0.05927035412 -0.25808020567 -0.43034121117 -1.0105012541 -
0.04934744243 0.1025113675 0.06442869705
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556
0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75
0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33 4.6662274923 26.64824319 1115.0943396 7
0.9056122449 6.4295874822

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
1996	12.37195122	97.548076923	1.7365604415	11.955550732	826.90369348	4.8520710059	0.58364975262	3.6008282777
Average	14.261302962	175.22309208	1730.3434269	3018.2186221	1327.9005294	5.9131373796	0.62794205686	3.9346090366
Max	23.892307692	692.88975875	19000	33000	9122.2304833	20	1.0261299435	9.5031055901
Min	7.6153846154	20.842105263	0.31786395423	0.75798327547	0.26905829596	2.5757575758	0.43996840442	1.4591238358
Range	16.276923077	672.04765349	18999.682136	32999.242017	9121.961425	17.424242424	0.58616153908	8.0439817543
STDEV	5.0409265726	174.19725371	5727.698417	9943.8400282	2613.2273401	4.795074442	0.20340483774	2.2071292155
VAR	25.41094071	30344.683199	32806529.156	98879954.506	6828957.1311	22.992738905	0.04137352801	4.871419374
Detection	0.2800812459	-0.04318223083	-0.17158388227	-0.20415152408	-0.24987967592	0.08199738544	-0.05728314648	-0.16690417063
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
1997	10.463196421	49.959223301	4.6196212808	17.437949068	540.99482368	4.8215686275	0.52024021352	5.2502493849
Average	12.795723549	70.260583685	5.6103188587	19.221919809	544.95956847	5.3495069615	0.57681783583	5.4741849518
Max	30.961538462	271.51058369	16.337644656	45.950155763	853.69863014	10.142857143	1.1908284024	15.311850312
Min	6.6168224299	14.75	1.6638935108	8.5106382979	251.76470588	2.4838709677	0.33484162896	2.3497636732
Range	24.344716032	256.76058369	14.673751145	37.439517465	601.93392426	7.6589861753	0.85598677344	12.962086639
STDEV	6.5119374997	47.440043374	4.4837986888	10.009063486	202.72778814	2.7361687319	0.22143288543	3.5561315988
VAR	42.405329999	2250.5577153	20.104450682	100.18135188	41098.556083	7.4866193295	0.04903252275	12.646071948
Detection	-0.31686874936	-0.28000044401	0.59723234154	0.07190075869	0.09829289467	0.07313041496	-0.1951299706	0.30035393339
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
1998	8.2382851446	52.630573248	2.3241010762	12.523500067	843.53381894	5.7314285714	0.42991675338	4.931776007
Average	10.51824683	76.825418673	1771.3845149	12243.932528	798.80036112	5.5723584585	0.44822210568	5.0318213057
Max	32.625	250.82541867	23000	159000	1690.7908163	11.375	0.82075471698	10.909090909
Min	2.7272727273	20	0.92165898618	4.399323181	0.11111111111	1.7777777778	0.125	2.0725388601
Range	29.897727273	230.82541867	22999.078341	158995.60068	1690.6797052	9.5972222222	0.69575471698	8.8365520489
STDEV	7.9587717223	54.008419099	6378.405282	44094.711311	435.28817925	2.4186942474		

0.17166483978 2.5261406373
VAR 63.342047328 2916.9093336 40684053.942 1944343565.6 189475.799 5.8500818623
0.02946881722 6.3813865196
Detection -0.28335959297 -0.55604402673 -0.01490637968 -0.17555387379 -0.27013142095
0.33762458472 -0.39148531874 0.21013484617
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556
0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75
0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33 4.6662274923 26.64824319 1115.0943396 7
0.9056122449 6.4295874822

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
Industry Type								
Telecomm	11.578378378	112.73684211	2.1131068231	18.461880665	865.95505618	6.9811320755	0.8595505618	4.3634009495
Average	10.65570474	157.21536797	3.5939095786	20.482505846	826.30952386	8.4133333333	0.77086385958	5.4851563857
Max	14.861111111	641.5487013	7.3333333333	31.090723751	1549.4230769	20	1.3894230769	10.909090909
Min	5.1923076923	22.5	0.54406964091	16.010922179	360	3	0.40744274809	2.593238062
Range	9.6688034187	619.0487013	6.7892636924	15.079801572	1189.4230769	17	0.98198032881	8.315852847
STDEV	4.5112354557	187.48554501	3.023153227	6.2404415432	472.84460753	6.6669312447	0.42156118665	3.1892371195
VAR	20.351245336	35150.829589	9.139455434	38.943110654	223582.02287	44.447972222	0.17771383409	10.171233404
Detection	0.4706076531	-0.1416403997	-0.07117151384	0.12345824094	-0.29743543503	0.70091048706	0.5425012213	0.04912208201
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Transport	11.972861842	74.280612245	1.7482673422	9.0981259644	1213.3225108	5.3100436681	0.65652056277	4.547162107
Average	12.71763751	102.05920599	2.2340136	11.623458586	2035.8014082	4.7943308839	0.58452126405	4.3094078759
Max	18.368421053	385.05920599	4.945498587	27.94246816	9122.2304833	7.8571428571	0.82469318663	7.1985885121
Min	6.9681818182	31.285714286	0.20042088386	0.75798327547	462.53068134	2.696969697	0.42965246637	2.378768021
Range	11.400239235	353.7734917	4.7450777031	27.184484885	8659.699802	5.1601731601	0.39504072026	4.8198204911
STDEV	3.437234436	80.53461829	1.7660027747	8.6176290078	2939.3130588	1.5972073629	0.12513857612	1.570909967
VAR	11.814580568	6485.8247431	3.1187658003	74.263529716	8639561.2579	2.5510713602	0.01565966323	2.4677581245
Detection	-0.0117835895	-0.09269704192	-0.16846204208	-0.34802991116	-0.72045071157	0.21512897329	0.1011316582	0.10117906713
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Financial	11.060845516	74.158940397	1.5980154976	9.116096355	1184.6076055	4.6525735294	0.58493522775	4.3907455013
Average	13.058775918	92.522099685	3.2869038313	15.561778515	788.98464487	5.1541532878	0.59553137545	4.2625168714
Max	30.961538462	293.77209969	12.778235779	34.784015531	1710.1273941	10.142857143	1.1908284024	7.435530086
Min	5.8653846154	34.451612903	0.70564761418	6.3973534903	292.93838863	2.7222222222	0.37841191067	2.3497636732
Range	25.096153847	259.32048678	12.072588165	28.386662041	1417.1890055	7.4206349208	0.81241649173	5.0857664128

STDEV 6.4008766514 53.136201464 3.2563577691 9.2677008216 386.35691863 2.1423701033 0.2172771633 1.5621842727
VAR 40.971221906 2823.455906 10.60386592 85.890278519 149271.66858 4.5897496594 0.04720936569 2.4404197018
Detection -0.01330982944 -0.20585043231 -0.20852920065 -0.34712505765 -0.68548242814 0.02400393297 -0.05448863533 0.05686841396
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556 0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75 0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33 4.6662274923 26.64824319 1115.0943396 7 0.9056122449 6.4295874822

Bin	Prep/Defect	Prep/Maj	Maj/KSLOC	Min/KSLOC	Lines/Conduct	Defects/Sess	Prep/Conduct	ROI
Manufact	10.160243408	47.933014354	6.6271363795	24.637727114	425.12244439	6.3612903226	0.56268254325	5.7321544102
Average	11.915031714	60.127599638	2880.6655117	19891.967213	528.26904873	6.3209120464	0.52514971482	6.1897267265
Max	32.625	234.12759964	23000	159000	1115.0943396	11.375	0.84255202629	15.311850312
Min	2.7272727273	14.75	1.0152284264	4.399323181	0.11111111111	1.7777777778	0.125	2.0725388601
Range	29.897727273	219.37759964	22998.984772	158995.60068	1114.9832285	9.5972222222	0.71755202629	13.239311452
STDEV	9.5777428641	51.449742676	8129.4404936	56208.134312	349.45235377	3.1318320904	0.25533339189	4.054557149
VAR	91.73315837	2647.0760215	66087802.74	3159354362.8	122116.94755	9.8083722426	0.06519514101	16.439433674
Detection	-0.34228531919	-0.31758766654	1.1325697012	0.43442734714	0.23939933949	0.52072393098	-0.10286403642	0.43687093773
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Medical	7.2421524664	48.939393939	1.6158648549	9.3034643163	1499.8164015	6.96875	0.49418604651	5.9840262134
Average	7.4834646392	89.956573501	1.5007043696	13.0617762	1358.736501	7.085	0.57088730252	5.4621177456
Max	8.7894736842	228.62324017	2.3792529146	20.461575065	1690.7908163	9.6	0.95918367347	7.7561311025
Min	6.2602040816	26.673913043	0.6197981229	6.789480831	784.3373494	5	0.3912627551	3.5269121813
Range	2.5292696026	201.94932713	1.7594547917	13.672094234	906.4534669	4.6	0.56792091837	4.2292189212
STDEV	1.0819106165	49.850855065	0.86045849502	6.8076531392	425.93872546	2.0531845184	0.26431651603	2.1712239056
VAR	1.1705305821	2485.1077507	0.74038882165	46.344141264	181423.79785	4.2155666667	0.06986322064	4.7142132484
Detection	-0.32966139062	-0.67963368904	-0.20376937202	-0.33769061851	-1.0693357057	0.69731104651	-0.25176946411	0.50822272334
LCL(14)	7.9680232558	46.606557377	0.92165898618	6.789480831	293.91763464	3.5555555556	0.45052631579	2.9021372328
Median	12.719101124	75.222222222	2.3792529146	16.010922179	621.71449461	4.75	0.60769230769	4.1889570552
UCL(54)	16.026041667	126.33	4.6662274923	26.64824319	1115.0943396	7	0.9056122449	6.4295874822
Defense	14.538855282	83.651941098	4.1252712904	19.610225371	537.96494355	4.7596899225	0.77351950881	4.3025852923
Average	16.499238064	140.05872838	3810.6332551	19114.878166	731.2642072	5.0096903636	0.84916154985	4.6482167668
Max	63.636363636	840.05872838	54000	290000	3060	14.235294118	2.1341463415	10.849176172
Min	6.7692307692	20.842105263	0.43706293706	1.4064697609	0.05410279531	1.8333333333	0.33484162896	1.1089494163
Range	56.867132867	819.21662312	53999.562937	289998.59353	3059.9458972	12.401960785	1.7993047125	9.7402267557
STDEV	10.427930258	150.03939416	11575.625361	63624.754419	736.9570696	2.4285137672		

0.42953861556 2.4874972821
VAR 108.74172947 22511.8198 133995102.49 4048109374.9 543105.72243 5.8976791175
0.18450342226 6.1876427287
Detection 0.10576945682 0.22566442699 0.46540567745 0.18128023017 0.10198260585
0.05514241933 0.35547719307 0.03189385051
LCL(14) 7.9680232558 46.606557377 0.92165898618 6.789480831 293.91763464 3.5555555556
0.45052631579 2.9021372328
Median 12.719101124 75.222222222 2.3792529146 16.010922179 621.71449461 4.75
0.60769230769 4.1889570552
UCL(54) 16.026041667 126.33 4.6662274923 26.64824319 1115.0943396 7
0.9056122449 6.4295874822