

Metrics and Models

The previous section surveyed 72 scholarly studies of software process improvement (SPI) techniques, methods, and strategies, attempting to provide the reader with a good sense of the wide-ranging approaches to SPI. This section examines 14 well known studies and expositions of metrics and models for software management and engineering, and more importantly software process improvement (SPI), identifying 74 broad metrics classes and 487 individual software metrics (see Table 39).

Table 39

Survey of Metrics for Software Process Improvement (SPI)

Author	Metric Class	Metrics
Davidson	Productivity, Quality, Velocity, Customer Service, Precision, Enhancement, Extension, Redefinition	39
Garrison	Prevention Costs, Appraisal Costs, Internal Failure Costs, External Failure Costs	36
Kan	Software Quality, Reliability, Quality Management, Structural Design, Customer Satisfaction	35
Grady	Process and Product Descriptions, High-Level Process Measurements, Defect Failure Analysis	15
Daskalantonakis	Planning, Containment, Reliability, Defect Density, Customer Service, Non-Conformance, Productivity	18
Barnard	Cost, Cycle Time, Quality, Conformity, Efficiency, Effectiveness, Productivity	21
Florac	Things Received or Used, Activities and their Elements, Things Consumed, Things Produced	80
Herbsleb	Cost, Productivity, Calendar Time, Quality, Business Value	7
McGibbon	Reuse, Clean Room, Inspections, Walkthroughs, Traditional, Full	24
Hays	Personal Software Process (PSP)	35
Jones	Process Improvement, Application/System, Productivity, Cost, Quality	28
Burr	Size, Complexity, Conformity, Defectiveness, Time, Cost	32
Rosenberg	Personnel Resources, Software Analysis, Changes to Code	9
Rico	Defect Density	6
Rico	Relational Design Metrics, Object Oriented Design Metrics, Universal/Structural Design Metrics	63
Rico	Planning, Overall, Review Rate, Substage Duration, Substage Interval, Substage Efficiency	39

While Table 39 provides a quick overview of the kinds of metrics classes the 14 studies refer to, it was necessary to reexamine and reclassify the 487 individual software metrics, based on a more consistent set of criteria. The analysis identified 12 common classes of software metrics from the 74 broad classes identified by the 14 sources, based on a more consistent set of criteria (see Table 40).

Table 40

Reclassification of 487 Metrics for Software Process Improvement (SPI)

Author	Productivity	Design	Quality	Effort	Cycle Time	Size	Cost	Change	Customer	Performance	ROI	Reuse
Davidson	18		4	1	5		5		6			
Garrison				28	1		7					
Kan	5	9	14			2		4	1			
Grady	2	3	4	3	2	1						
Daskalantonakis	4		11	1	1		1					
Barnard	21											
Florac	15	7	14	9	4	9	7	12		3		
Herbsleb	2		1		1		2				1	
McGibbon				7	1	6	7				2	1
Hays	7		3	1	13	11						
Jones	14	2	6	3	1	2						
Burr	10	6	6		2	5			2	1		
Rosenberg	1	2		1		1		3				1
Rico	12	63	9	13	11							

It's not surprising that productivity, design, quality, and effort were the most frequently cited software metrics in the 14 studies, given that academic and industry use of these metrics, especially productivity and design, dates back nearly three decades. Size came in sixth place with only an 8% rate of occurrence, probably because function point proponents stigmatize the use of size metrics as incompetence, despite their continuing strategic importance.

The software metrics were reclassified using the following criteria, productivity—units per time, design—complexity, quality—defect density, effort—hours, cycle time—duration, size—lines of code or function points, cost—dollars, change—configuration management, customer—customer satisfaction, performance—computer utilization, ROI—return-on-investment, and reuse—percent of reused source code. The strength of this reclassification strategy is that the 487 individual software metrics fell succinctly into one of the 12 software metric classes without exception. The weakness of the reclassification is that it hides exactly what is being measured, such as productivity of a software life cycle versus software process.

While some metrics are cited as high as 22% of the time in the case of productivity, versus only 1% for ROI, there is no prioritization and importance placed on the software metrics by the software metrics classes. As mentioned before, quality is a strategic and powerful metric, though only cited 15% of the time, and size is a principle input to most cost models, though only cited 8% of the time. The importance of using customer satisfaction measurement cannot be understated though it is only cited 2% of the time. And, reuse will begin to emerge as one of the most strategic metrics of the next millenium, though it is only cited 0% of the time on average (see Figure 11).

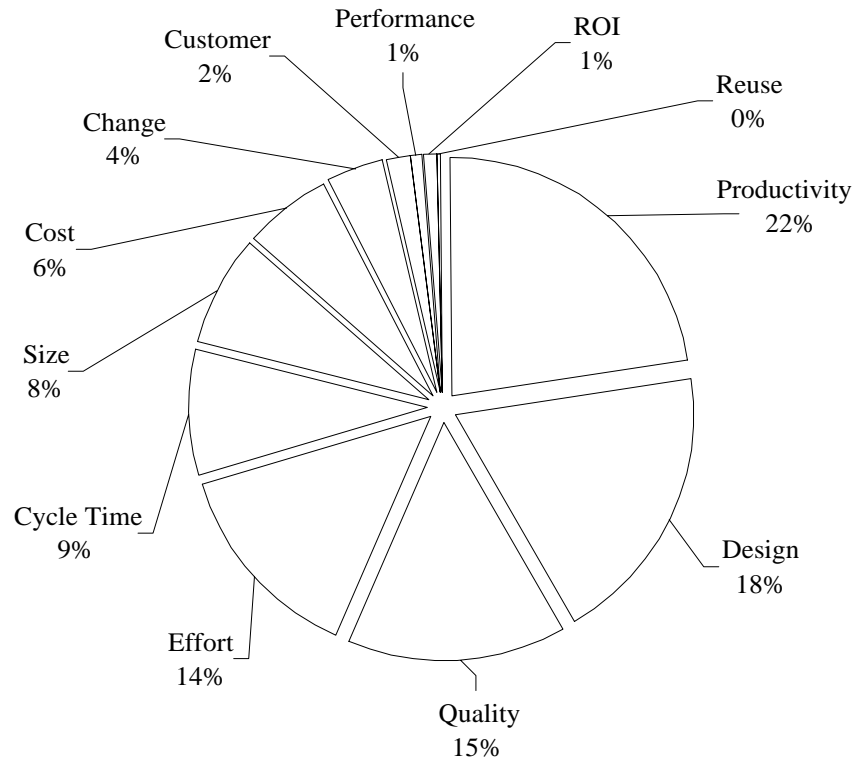


Figure 11. Citation frequency of metrics for software process improvement (SPI).

Davidson (1993) identified eight major metric classes and 39 individual metrics of what he calls “operating performance parameters and metrics,” for business transformation, an advanced form of process improvement, reengineering, and enterprise automation (see Table 41).

Table 41

Operating Parameters and Metrics for Business Transformation

Class	Metric
Productivity	Units Per Person
	Peak Output Level
	Cost Per Unit
	Cost Per Activity
	Revenue Per Employee
	Headcount
Quality	Defect Rates
	Yields
	Standards and Tolerances
	Variance
	Life Cycle Costs
Velocity	Inventory and Sales
	Throughput
	Cycle Times
	Time To Market
	Response Ratios
Customer Service	Retention
	Revenue Per Customer
	Repeat Purchase
	Brand Loyalty
	Customer Acquisition Cost
Business Precision	Referral Rate
	Cost of Variety
	Number of New Products
	Number of Product, Service, and Delivery Configurations
Enhancement	Customer Self-Design and Self-Pricing Flexibility
	Number of Features, Functions, and Services
	Information Flow to Customer
	Product and Service Revenue Ratio
	Customer Performance (Industrial)
Extension	Secondary Revenue Streams
	Customer Diversity
	Number of New Customers
	Channel Diversity
	New Revenue Sources
Business Redefinition	Broader Product and Market Scope
	Market Value
	New Lines of Business
	Percent of Revenue from New Units and Services

Garrison and Noreen (1997a) identify four major metrics classes and 36 individual metrics for what they call “typical quality costs,” for cost measurement, cost control, and cost minimization (see Table 42).

Table 42

Typical Costs for Measuring Quality of Conformance

Class	Metric
Prevention Costs	Systems development
	Quality engineering
	Quality training
	Quality circles
	Statistical process control activities
	Supervision of prevention activities
	Quality data gathering, analysis, and reporting
	Quality improvement projects
	Technical support provided to suppliers
	Audits of the effectiveness of the quality system
Appraisal Costs	Test and inspection of incoming materials
	Test and inspection of in-process goods
	Final product testing and inspection
	Supplies used in testing and inspection
	Supervision of testing and inspection activities
	Depreciation of test equipment
	Maintenance of test equipment
	Plant utilities in the inspection area
Field testing and appraisal at customer site	
Internal Failure Costs	Net cost of scrap
	Net cost of spoilage
	Rework labor and overhead
	Reinspection of reworked products
	Retesting of reworked products
	Downtime caused by quality problems
	Disposal of defective products
	Analysis of the cause of defects in production
	Re-entering data because of keying errors
Debugging of software errors	
External Failure Costs	Cost of field servicing and handling complaints
	Warranty repairs and replacements
	Repairs and replacements beyond the warranty period
	Product recalls
	Liability arising from defective products
	Returns and allowances arising from quality problems
Lost sales arising from a reputation for poor quality	

Kan (1995) identified five major metrics classes and 35 individual metrics for what he called “metrics and models,” for software quality engineering, an established, yet advanced form of measurement-based management for software development (see Table 43).

Table 43

IBM Rochester Software Process Improvement (SPI) Metrics

Class	Subclass	Metrics and Models
Software Quality	Product Quality	Defect Density
		Customer Problems
		Customer Satisfaction
		Function Points
	In-Process Quality	Defect Density During Machine Testing
		Defect Arrival Pattern During Machine Testing
		Defect Removal Effectiveness
		Phase-Based Defect Removal Model Pattern
		Special Case Two-Phase Model
		Maintenance
Reliability	Exponential	Fix Backlog and Backlog Management Index
		Fix Response Time
		Percent Delinquent Fixes
	Reliability Growth	Fix Quality
		Cumulative Distribution Function (CDF)
		Probability Density Function (PDF)
		Rayleigh
		Jelinski-Moranda
		Littlewood
		Goel-Okumoto
Musa-Okumoto Logarithmic Poisson Execution		
Quality Management	Life Cycle	Delayed S and Inflection S
	Testing Phase	Rayleigh Life Cycle Reliability
		Problem Tracking Report
Structural Design	Complexity	Reliability Growth
		Source Lines of Code (SLOC)
		Halstead's Software Science
		Cyclomatic Complexity
		Syntactic Constructs
	Structure	Invocation Complexity
		System Partitioning
		Information Flow
		Fan-In and Fan-Out
Customer Satisfaction	Survey	In-Person, Phone, and Mail
	Sampling	Random, Systematic, and Stratified
	Analysis	CUPRIMDA

Grady (1997) identified three major metrics classes and 15 individual metrics for what he called “baseline measurements for all software process improvement programs,” as part of his plan, do, check, and act (PDCA)-based methodology—specifically the check phase—evaluate results, ensure success, and celebrate (see Table 44).

Table 44

Hewlett Packard Software Process Improvement (SPI) Metrics

Class	Metric
Process and Product Descriptions	Development Type
	Computer Programming Language
	Type of Product
High-Level Process Measurements	Product Size
	Effort
	Productivity before Changes
	Productivity after Changes
	Activity Breakdown before Changes
	Activity Breakdown after Changes
	Defects before Changes
	Defects after Changes
	Project Calendar Time before Changes
Project Calendar Time after Changes	
Defect Failure Analysis	Defect Failure Analysis before Changes
	Defect Failure Analysis after Changes

Grady and Caswell (1986) report that Hewlett Packard uses 12 other strategic software metrics. Hewlett Packard’s first six software metrics include, average fixed defects per working day, average engineering hours per fixed defect, average reported defects per working day, bang, branches covered per total branches, and defects per thousands of non-commented source statements. Hewlett Packard’s last six software metrics include, defects per line of documentation, defects per testing time, design weight, non-commented source statements per engineering month, percent overtime per 40 hours per week, and (phase) engineering months per total engineering months.

Daskalantonakis (1992) identified seven major metrics classes and 18 individual metrics for what he called a “practical and multi-dimensional view of software measurement,” in support of Motorola’s company-wide metrics program (see Table 45).

Table 45

Motorola Software Process Improvement (SPI) Metrics

Class	Metric
Project Planning	Schedule Estimation Accuracy Effort Estimation Accuracy
Defect Containment	Total Defect Containment Effectiveness Phase Containment Effectiveness
Software Reliability	Failure Rate
Software Defect Density	In-Process Faults In-Process Defects Total Release Defects Total Release Defects Delta Customer-Found Defects Customer-Found Defects Delta
Customer Service	New Open Problems Total Open Problems Mean Age of Open Problems Mean Age of Closed Problems
Non-Conformance Cost	Cost of Fixing Problems
Software Productivity	Software Productivity Software Productivity Delta

Diaz and Sligo (1997) report that Motorola uses three strategic metrics for measuring the effects of software process improvement (SPI), quality, cycle time, and productivity. Quality is defined as defects per million earned assembly-equivalent lines of code (a form of defect density measurement). Cycle time is defined as the amount of calendar time for the baseline project to develop a product divided by the cycle time for the new project. And, productivity is defined as the amount of work produced divided by the time to produce that work.

Barnard and Price (1994) identified seven major metrics classes and 21 individual metrics for what they called “managing code inspection information,” in support of AT&T’s efforts to ensure a more consistently effective Software Inspection Process (see Table 46).

Table 46

AT&T Software Inspection Process (SPI) Metrics

Class	Metric
Cost	Average Effort per Thousand Lines of Code
	Percentage of Re-Inspections
Cycle Time	Average Effort per Thousand Lines of Code
	Total Thousand Lines of Code Inspected
Quality	Average Faults Detected per Thousand Lines of Code
	Average Inspection Rate
	Average Preparation Rate
Conformity	Average Inspection Rate
	Average Preparation Rate
	Average Lines of Code Inspected
	Percentage of Re-Inspections
Efficiency	Total Thousand Lines of Code Inspected
Effectiveness	Defect Removal Efficiency
	Average Faults Detected per Thousand Lines of Code
	Average Inspection Rate
	Average Preparation Rate
	Average Lines of Code Inspected
Productivity	Average Effort per Fault Detected
	Average Inspection Rate
	Average Preparation Rate
	Average Lines of Code Inspected

The Software Inspection Process metric classes answer these questions: How much do inspections cost? How much calendar time do inspections take? What is the quality of the inspected software? To what degree did the staff conform to the procedures? What is the status of inspections? How effective are inspections? What is the productivity of inspections?

Florac and Carleton (1999) identified four major metrics classes, 19 metrics subclasses, and 80 individual metrics for what they call “measurable attributes of software process entities,” in support of statistical process control (SPC) for software process improvement (see Table 47).

Table 47

SEI Software Process Improvement (SPI) Metrics

Things Received or Used	Activities & their Elements	Things Consumed	Things Produced
Changes	Flow Paths	Effort	Status of Work Units
Type	Processing Time	# Development Hours	# Designed
Date	Throughput Rates	# of Rework Hours	# Coded
Size	Diversions	# of Support Hours	# Tested
# Received	Delays	# of Preparation Hours	Size of Work Units
Requirements Changes	Backlogs	# of Meeting Hours	# of Requirements
Requirements Stability	Length, Size	Time	# of Function Points
# Identified	Queues	Start Time or Date	# of Lines of Code
% Traced to Design	Buffers	Ending Time or Date	# of Modules
% Traced to Code	Stacks	Duration of Process	# of Objects
Problem Reports		Wait Time	# of Bytes in Database
Type		Money	Output Quantity
Date		Cost to Date	# of Action Items
Size		Cost Variance	# of Approvals
Origin		Cost of Rework	# of Defects Found
Severity			Test Results
# Received			# of Passed Test Cases
Funds			% Test Coverage
Money			Program Architecture
Budget			Fan-In
Status			Fan-Out
People			Changes
Years of Experience			Type
Type of Education			Date
% Trained in XYZ			Size
Employment Codes			Effort Expended
Facilities & Environment			Problems & Defects
Space per Employee			# of Reports
Noise Level			Defect Density
Lighting			Type
# of Staff in Cubicles			Origin
# of Staff Sharing Office			Distribution by Type
Investment in Tools			Distribution by Origin
Computer Usage Hours			# Open
% Capacity Utilization			# Closed
			Resource Utilization
			% Memory Utilized
			% CPU Capacity Used
			% I/O Capacity Used

Herbsleb, Carleton, Rozum, Siegel, and Zubrow (1994) identified five major metrics classes and seven individual metrics for measuring the benefits of SEI Capability Maturity Model® for Software (SW-CMM®)-based software process improvement (see Table 48).

Table 48

SEI CMM-Based Software Process Improvement (SPI) Metrics

Class	Metric
Cost	Thousands of Dollars per Year Spent on SPI Dollars per Software Engineer per Year Spent on SPI
Productivity	Gain per Year in Productivity Gain per Year in Early Detection of Defects
Calendar Time	Reduction per Year in Calendar Time to Develop Software System
Quality	Reduction per Year in Post-Release Defect Reports
Business Value	Business Value Ratio of SPI Efforts

Herbsleb et al. also recommend that organizations use four more additional classes of metrics to measure software process improvement (SPI), balanced scorecard, CMM/SEI core measures, business value, and quality metric classes. Balanced scorecard consists of financial, customer satisfaction, internal processes, and innovation and improvement activity metrics. CMM/SEI core measures consist of resources expended on software process improvements, resources expended to execute the software processes, amount of time (calendar time) it takes to execute the process, size of the products that result from the software process, and quality of the products produced metrics. Business value consists of increased productivity, early error detection and correction, overall reduction of errors, improved trends in maintenance and warranty work, and eliminating processes or process steps metrics. Quality consists of mean time between failures, mean time to repair, availability, and customer satisfaction metrics.

McGibbon (1996) identified three major metrics classes and 24 individual metrics for what he called “a business case for software process improvement” comparing Software Reuse, the Software Inspection Process, and the Clean Room Methodology (see Table 49).

Table 49

DACS Software Process Improvement (SPI) Metrics

Class	Metric
Without, 30% Reuse, 60% Reuse, & 90% Reuse	Estimated Source Lines of Code % Reuse Equivalent Ratio on Reuse Equivalent Code COCOMO Effort Estimate Equivalent Cost Estimated Rework (New Code) Estimated Rework (Reused Code) Estimated Rework (Total Rework) Estimated Maintenance Costs Development Effort + Maintenance Savings of Reuse Over No Reuse % Reduction
Clean Room, Inspections, & Walkthroughs	Estimated Source Lines of Code Equivalent Ratio Equivalent Code Effort Estimate Equivalent Cost
Traditional, Inspections, Reuse, Clean Room, & Full	Development Costs Rework Costs Maintenance Costs Development + Maintenance Savings Software Process Improvement (SPI) Costs Return-on-Investment (ROI)

McGibbon identified another six major metric classes and 23 individual metrics for performing a detailed analysis and comparison of the Software Inspection Process and what he called "Informal Inspections," otherwise known as Walkthroughs.

Hays and Over (1997) identified 34 individual strategic software metrics in support of the Personalsm Software Process (PSPsm) pioneered by Watts S. Humphrey (see Table 50).

Table 50

Personal Software Process (PSP) Metrics

Metric	Definition
Interruption Time	Elapsed time for small interruptions from project work such as a phone call
Delta Time	Elapsed time in minutes from start to stop less interruptions
Planned Time in Phase	Estimated time to be spent in a phase for a project
Actual Time in Phase	Sum of delta times for a phase of a project
Total Time	Sum of planned or actual time for all phases of a project
Time in Phase to Date	Sum of time in actual time in phase for all completed projects
Total Time to Date	Sum of time in phase to date for all phases of all projects
Time in Phase to Date %	100 * time in phase to date for a phase divided by total time in phase to date
Compile Time	Time from the start of the first compile until the first clean compile
Test Time	Time from the start of the initial test until test completion
Defect	Any element of a program design or implementation that must be changed
Defect Type	Project defect type standard
Fix Time	Time to find and fix a defect
Lines of Code	Logical line of code as defined in the engineer's counting & coding standard
Base Lines of Code	Lines of code from a previous version
Deleted Lines of Code	Deletions from the base lines of code
Modified Lines of Code	Modifications to the base lines of code
Added Lines of Code	New objects, functions, procedures, or any other added lines of code
Reused Lines of Code	Lines of code from a previous program that is used without modification
New Lines of Code	Sum of added lines of code
Changed Lines of Code	Sum of modified lines of code
Total Lines of Code	Total program lines of code
Total New Reused	New or added lines of code that were written to be reusable
Lines of Code Type	Base, deleted, modified, added, reused, new, changed, total, total new reused
Lines of Code/Hour	Total new & changed lines of code developed divided by development hours
Estimating Accuracy	Degree to which the estimate matches the result
Test Defects/KLOC	Defects removed in the test phase per new and changed KLOC
Compile Defects/KLOC	Defects removed in compile per new and changed KLOC
Total Defects/KLOC	Total defects removed per new and change KLOC
Yield	Percent defects injected before the first compile removed before first compile
Appraisal Time	Time spent in design and code reviews
Failure Time	Time spent in compile and test
Cost of Quality	Appraisal time plus failure time
Appraisal/Failure Ratio	Appraisal time divided by failure time
Review Rate	Lines of code reviewed per hour

Various forms of defect density metrics and appraisal to failure ratio are the key metrics to focus on. Appraisal to failure ratio must reach a modest 67% in order achieve zero defects.

Jones (1997a) identified five major metrics classes and 24 individual metrics for what he called “the six stages of software excellence” for quantifying the impact of software process improvements (see Table 51).

Table 51

SPR Software Process Improvement (SPI) Metrics

Class	Metric
Process Improvement	Process Improvement Expenses per Capita Process Improvement Stages in Calendar Months Improvements in Delivered Defects Improvement in Development Productivity Improvements in Development Schedule Organization Size in Number of People Capability Maturity Model for Software Level
Application/System	Application Class Programming Language Size in Function Points Size in Lines of Code
Productivity	Work Hours per Month (Function Points) Average Monthly Salary Function Points per Month Lines of Code per Month Cost per Function Point Cost per Line of Code
Cost	Work Hours per Function Point per Activity Staff (Number of People) per Activity Effort (Months) per Activity Schedule (Months) per Activity Costs per Activity Percent of Costs per Activity
Quality	Potential Defects (Estimated) Defect Removal Efficiency Delivered Defects Defects per Function Point Defects per Thousand Lines of Code

Burr and Owen (1996) identified seven major metrics classes and 32 individual metrics for what he called “commonly available metrics” with which to perform statistical process control (SPC) for software process improvement (SPI) (see Table 52).

Table 52

Software Process Improvement (SPI) Metrics for SPC

Class	Subclass	Subclass
Product	Size	Lines of Code per Module
		Modules per Function
		Functions per System
		Data Types per Areas
		Variables
	Complexity	McCabe's
		Path Count
		Call Count
		Data Types
	Conformity	Completeness
		Functional Differences
		Supplier Confidence
		Customer Confidence
Defectiveness	Defect Count	
	Function Failures	
	Data Faults	
	Machine or System Failures	
	Reliability	
Process	Time	Lines of Code per Day
		Lines of Code per Hour
		Modules per Month
		Review Time
		Stage Time
		Preventative per Total Time
		Corrective per Total Time
	Cost	Systems Utilization
		Cost per Line of Code
		Cost per Line of Module
		Cost of Correction
	Defectiveness	Cost of Failure
		Error Count
		Error Rate per Module

Rosenberg, Sheppard, and Butler (1994) identified three broad metrics classes, three metrics subclasses, and nine individual metrics for what they called “Software Process Assessment (SPA) metrics” in support of software process improvement (SPI) (see Table 53).

Table 53

NASA GSFC Software Process Improvement (SPI) Metrics

Class	Subclass	Metric
Process	Personnel Resources Form	Effort by Phase
Product	Software Analysis	Complexity Readability Size
	Changes to Code	Component Origination (% Complete) Component Origination (Amount Reused) Change Report (Type) Change Report (Date) Change Report (Effort)

Rico (1998) identified five forms of defect density metrics for what he called “Quality Metrics” in direct support of software process improvement (SPI) measurement (see Table 54).

Table 54

Defect Density Metrics for Software Process Improvement (SPI)

Source	Metric Name	Metric Algorithm
IEEE	Defect Density	$\frac{\text{Defects}}{\text{KSLOC}}$
IBM (Michael Fagan)	Defect Removal Effectiveness	$\frac{\text{Inspection Defects}}{\text{Inserted Defects}} \times 100\%$
IBM (NASA Space Shuttle)	Early Detection Percentage	$\frac{\text{Major Inspection Defects}}{\text{Inserted Defects}} \times 100\%$
Dunn	Effectiveness	$\frac{\text{Defects}}{\text{Current Phase} + \text{Post Phase}} \times 100\%$
Motorola	Total Defect Containment Effectiveness	$\frac{\text{Pre-Release Defects}}{\text{Pre-Release} + \text{Post-Release Defects}}$
Motorola	Phase Containment Effectiveness	$\frac{\text{Phase Errors}}{\text{Phase Errors} + \text{Phase Defects}}$

Rico (1998) identified three metric classes and 63 individual metrics for what he called “software product metrics” in support of software process improvement (SPI), relational design metrics, object oriented design metrics, and universal/structural design metrics (see Table 55).

Table 55

Universal/Structural Design Metrics for Software Process Improvement (SPI)

Relational Design Metrics	Object Oriented Design Metrics	Universal/Structural Design Metrics
Attribute Consistency	System Size in Classes	Cyclomatic Complexity
Use of Domains	Number of Hierarchies	Fan In
Unnecessary Keys	Number of Independent Classes	Fan Out
Foreign Key Indexes	Number of Single Inheritance	Dependencies
Keys on Similar Attributes	Number of Multiple Inheritance	Design Changes
Relationships (Multiple Paths)	Number of Internal Classes	Historical Defectiveness
Relationships (Infinite Loops)	Number of Abstract Classes	Current Defectiveness
Relationships (Implied)	Number of Leaf Classes	Software Science
Unnecessary Denormalization	Average Depth of Inheritance	Static Graph Theoretic Complexity
Index Consistency	Average Width of Inheritance	Generalized Graph Theoretic Complexity
Missing Indexes (Defined Relationships)	Average Number of Ancestors	Dynamic Graph Theoretic Complexity
Missing Indexes (Implied Relationships)	Measure of Functional Abstraction	Unit Test Case Determination
Excessive Indexes	Measure of Attribute Abstraction	Design Structure
Unnecessary Indexes	Data Access Metric	Data Flow Complexity
No Unique Identifier	Operation Access Metric	
Unique Constraint	Number of Methods	
Use of Surrogate Keys	Class Interface Size	
Redundant Attributes	Number of Inline Methods	
Repeating Groups	Number of Polymorphic Methods	
Homonym Attributes	Number of Attributes	
Missing Tables	Number of Abstract Data Types	
Inconsistent Attribute Definition	Class Size in Bytes	
Inconsistent Constraint Definition	Direct Class Coupling	
Incorrect Relationship Definition	Direct Attribute Based Coupling	
Disabled Constraints		

Rico (1996) identified six metric classes and 39 individual metrics for what he called “Software Inspection Process metrics” in support of software process improvement (SPI), publishing a comprehensive set of software inspection process cost models (see Table 56).

Table 56

Software Inspection Process Metrics for Software Process Improvement (SPI)

Metric Class	Metric	Model	
Planning	Rico	$SLOC / (Rate * 2) * (Team Size * 4 + 1)$	
	Hewlett Packard	$SLOC / (Rate * 2) * 25$	
	AT&T	$50 * KSLOC$	
	Bell Northern Research	$3 * KSLOC * 4 * 8$	
	Tom Gilb	$SLOC / (Rate * 2) * (5.76 * Team Size)$	
Overall	Total Defects		
	Defects per Hour		
	Major Defects		
	Major Defects per Hour		
	Minor Defects		
	Minor Defects per Hour		
	Defect Removal Efficiency		
	Total Hours		
	Duration		
	People		
	Review Rate	Overview Rate	
		Preparation Rate	
		Inspection Rate	
Rework Rate			
Substage Duration	Planning Hours		
	Overview Hours		
	Preparation Hours		
	Inspection Hours		
	Rework Hours		
	Followup Hours		
Substage Interval	Planning/Overview Interval		
	Overview/Preparation Interval		
	Preparation/Inspection Interval		
	Inspection/Rework Interval		
	Rework/Followup Interval		
	Planning/Preparation Interval		
	Planning/Inspection Interval		
	Planning/Rework Interval		
	Planning/Followup Interval		
	Inspection/Followup Interval		
Substage Efficiency	Preparation Efficiency		
	Inspection Efficiency		
	Inspection Gain Rate		
	Inspection Suppression		