CHALLENGES OF SOFTWARE PROJECT MANAGEMENT

March 21, 2017 Version 2.0

Capers Jones, VP and CTO, Namcook Analytics LLC
Email: Capers.Jones3@gmail.com
Web: www.Namcook.com

Abstract

Project management in every industry is a challenging occupation. But the challenges and hazards of software project management are greater than most other industries. This fact is proven by the large number of software project cancellations and the high frequency of software project cost and schedule overruns. Software projects run late and exceed their budgets more than any other modern industry except for defense projects.

Academic training for software project managers is not very good in 2017. Some technology companies have recognized the challenges of software project management and created effective in-house training for new software project managers. These companies with effective software project management training include IBM and a number of telecom companies such as AT&T, ITT, Motorola, Siemens, and GTE. Some other technology companies such as Google and Apple also have effective software project management training. (Many of the companies with good software management training build complex physical devices run by software. Most are also more than 75 years old and have had software measurement programs for more than 40 years.)

A few companies even market effective software project management training. One of these is a subsidiary of Computer Aid Inc. called the Information Technology Metrics and Productivity Institute (ITMPI). The non-profit Project Management Institute (PMI) also offers effective training for software project managers.

INTRODUCTION

Several years ago a survey of engineering technology company CEO's (computers. Telecom, electronics, medical devices, autos, aircraft) found that they regarded their software organizations as the least professional of any of the corporate engineering organizations. This was due to the fact that software projects had higher cancellation rates, longer schedule delays, and higher cost overruns than any of the other engineering organizations.

Lyman Hamilton, a former Chairman of the ITT Corporation, gave an internal speech to ITT executives in which he mentioned that newly hired software engineers just out of college needed about three years of internal training before being entrusted with critical projects. Other kinds of engineers such as mechanical and electrical engineers only needed about 12 months of internal training.

Hamilton was troubled by several major software failures of projects that were terminated without being completed. He was also troubled by the dissatisfaction expressed by customers in the quality of the software the corporation produced. He was further dissatisfied by the inability of internal software executives to explain why the problems occurred, and what might be done to eliminate them.

It is interesting that the failing projects were all large systems in the 10,000 function point size range. Failures in this range are common, and managerial problems are usually a key factor.

Problems, failures, and litigation are directly proportional to the overall size of software applications measured using function point metrics. Table 1 shows the approximate distribution of software project results circa 2017:

Table 1: Normal Software Results based on Application Size Circa 2017

Note: Costs are based on $10,000 per month

Size in Function Points	Schedule in calendar months	Total Staffing	Productivity in Function Points per Staff Month	Cost in U.S. Dollars	Odds of Project Failure	Odds of Outsource Litigation
1	0.02	1	50.00	$200	0.10%	0.00%
10	0.40	1	25.00	$4,000	1.00%	0.01%
100	3.50	2	14.29	$70,000	2.50%	0.25%
1,000	15.00	6	11.11	$900,000	11.00%	1.20%
10,000	**35.00**	**50**	**5.71**	**$17,500,000**	**31.00%**	**7.50%**
100,000	**60.00**	**575**	**2.90**	**$345,000,000**	**47.50%**	**23.00%**

As can be seen from table 1 large software projects are distressingly troublesome and have frequent total failures also a high risk of litigation. Poor project management is a key contributing factor.

Some leading companies have recognized the difficulty of successful software project management and taken active steps to improve the situation. Some of these companies include IBM, AT&T, ITT, Motorola, GTE, and Siemens. Google, Apple, and Microsoft have also attempted to improve software management although Microsoft has perhaps been too rigid in some management topics such as employee appraisals.

The companies that are most proactive in software project management tend to build complex engineered products such as computers, medical devices, aircraft controls, and switching systems that depend upon software to operate. The companies also tend to be mature companies founded over 75 years ago and having effective software measurement programs that are more than 40 years old. Most were early adapters of function point metrics and also early adapters of parametric software estimation tools.

The software benchmarks studies carried out by Namcook Analytics LLC often finds a significant number of serious software project management problems and issues. Table 2 summarizes 40 problems noted in a benchmark study for a Fortune 500 technology corporation:

Table 2: Corporate Software Risk Factors Found by a Namcook Benchmark Study

1	**Project management: no formal training for new managers**
2	**Project management: no annual benchmark studies**
3	**Project management: no annual training in state of the art methods**
4	**Project management: no training in software cost estimating**
5	**Project management: no training in software quality estimating**
6	**Project management: no training in software risk analysis**
7	**Project management: no training in cyber-attack deterrence**
8	**Project management: no training in function point metrics**
9	**Project management: no training in schedule planning**
10	**Project management: lack of accurate productivity measurements**
11	**Project management: lack of accurate quality metrics**
12	**Project management: incomplete milestone and progress tracking**
13	**Project management: historical data "leaks" by over 50%**
14	**Project management: managers continue to use inaccurate manual estimates**
15	**Project management: no widespread use of accurate parametric estimation**
16	Quality control: no use of requirements models or QFD
17	Quality control: no use of automated proofs for critical features
18	Quality control: no use of cyber-attack inspections
19	Quality control: no use of formal design inspections
20	Quality control: no use of formal code inspections
21	Quality control: no use of static analysis tools
22	Quality control: no use of mathematical test case design (cause effect graphs)
23	Quality control: no use of test coverage tools
24	Quality control: defect potentials about 4.75 bugs per function point
25	Quality control: defect removal efficiency (DRE) below 90.00%
26	Maintenance: no use of complexity analysis or cyclomatic complexity
27	Maintenance: no use of renovation tools or work benches
28	Maintenance: no use of code restructuring tools
29	Maintenance: inconsistent use of defect tracking tools
30	Maintenance: no use of inspections on enhancements
31	No reuse program: requirements
32	No reuse program: design
33	No formal reuse program: source code
34	No reuse program: test materials
35	No reuse program: documentation
36	No reuse program: project plans
37	No formal corporate reuse library
38	No corporate contracts with third party reuse companies

39 Office space: small open offices; high noise levels, many interruptions
40 Insufficient meeting/breakout space for team meetings; no large meetings

Some 15 of the 40 problems or about 38% were software project management problems. This distribution is not uncommon.

The author of this report is a frequent expert witness in litigation for software projects that either failed without being delivered or operated so poorly after delivery that the clients sued the vendors. It is interesting that project management problems were key factors in every lawsuit. Inaccurate estimation, poor tracking of progress, and poor quality control are endemic problems of the software industry and far too common even in 2017. These problems have been part of every breach of contract case where the author worked as an expert witness.

IMPROVING SOFTWARE PROJECT MANAGEMENT TOOLS AND TRAINING

From consulting and benchmark studies carried out among top-tier technology corporations they all have taken effective steps to improve and professionalize software project management. Some of these steps include at least the following 10 steps:

Table 3: Ten Steps to Effective Software Project Management

1. Formal internal training for new project managers (10 days).
2. Annual training for project managers and technical staff (5 days).
3. Guest lectures from top software professionals (3 days).
4. Acquisition and use of parametric estimation tools.
5. Acquisition and use of effective progress and milestone tracking tools.
6. Use of formal project offices for applications > 5,000 function points.
7. Use of and measurement of effective quality control methods.
8. Elimination of bad software metrics and adoption of effective metrics.
9. Commissioning annual software benchmark studies.
10. Formal "best practice" analysis of tools, methods, reuse, and quality.

Let us now consider each of these 10 steps in sequence.

Initial Education for New Project Managers

In many technology companies project managers are often selected from the ranks of technical software engineering personnel. If this is so, they usually had close to zero management training at the university level. IBM recognized this as a problem back in the 1950's and introduced an effective training program for newly hired or newly appointed project managers.

This training is given to all project managers, but this report only covers software training topics. Since new project management training lasts for 10 days, there were 10 topics covered:

Table 4: New Software Project Manager Curriculum

	Project Management Courses	Days	Value
1	Software Milestone Tracking	1	10
2	Sizing key software deliverables	1	10
3	Software Project Planning	1	10
4	Cyber-attack defenses	1	10
5	Software Risk Management	1	10
6	Software Cost Estimating: Automated	1	10
7	Measurement and Metrics of Software	1	10
8	Software Quality and Defect Estimating	1	10
9	Human Resource Policies	1	9
10	Appraisals and Employee Relations	1	9

Eight of the 10 topics are technical and deal with actual project issues such as cyber-attacks and risks. Two of the 10 topics deal with human resources and appraisals, which are of course a critical part of any manager's job.

(Microsoft has received criticism for their appraisal system, which uses mathematical curves and requires that only a certain percentage of employees can be appraised as "excellent." The problem with this is that technology companies such as Microsoft tend to have more excellent employees than ordinary companies do, so this curve tended to cause voluntary attrition among capable employees who ended up on the wrong side of the "excellent" barrier.)

Continuing Education for Software Project Managers

A study of software education methods carried out by Namcook Analytics LLC found that in-house education in major companies such as IBM and AT&T was superior to academic or university training for software project managers.

IBM and AT&T both employed more than 100 education personnel. These educators taught in-house courses to software and other personnel, and also taught customer courses to clients. The education groups operated initially as cost centers with no charges for in-house or customer training. More recently they have tended to switch to profit-center operations and do charge for training, at least for some customer training.

Quite a few technology companies have at least 10 days or training for new managers and about a week of training each year for both managers and technical staff. When Capers Jones was a new manager at IBM he took this 10-day training series, and later taught some of the IBM

project management courses on estimation, measurements, quality control, and software risk analysis.

Table 5 shows the rankings of 15 channels of software project management education in order of effectiveness:

Table 5: Ranking Software Management Education Channels

1. In-house education in technology companies
2. Commercial education by professional educators
3. University education – graduate
4. University education - undergraduate
5. In-house education in non-technology companies
6. Mentoring by experienced managers
7. On-the-job training
8. Non-profit education (IEEE, PMI, IFPUG, etc.)
9. Vendor education (management tools)
10. Self-study from work books
11. Self-study from CD-ROMs or DVDs
12. Live Conferences with seminars and tutorials
13. On-line education via the Internet and World Wide Web
14. Project management books
15. Project management journals

A composite software project management curriculum derived from technology companies such as IBM, AT&T, ITT, Microsoft, and Apple is shown in table 6:

Table 6: Software Project Management Curriculum

	Project Management Courses	**Days**	**Value**
1	Software Milestone Tracking	1.00	10.00
2	Early Sizing Before Requirements	1.00	10.00
3	Sizing key deliverables	1.00	10.00
4	Controlling creeping requirements	1.00	10.00
5	Software Project Planning	2.00	10.00
6	Cyber-attack defenses	2.00	10.00
7	Cyber-attack recovery	1.00	10.00
8	Software outsourcing pros and cons	1.00	10.00
9	Optimizing Multi-Country Teams	1.00	10.00
10	Best Practices in Project Management	1.00	10.00
11	Software Risk Management	1.00	10.00
12	Software Cost Estimating: Automated	2.00	10.00
13	Software High-Security Architecture	1.00	10.00
14	Benchmark sources: ISBSG, Namcook, etc.	1.00	10.00

15	Measurement and Metrics of Software	2.00	10.00
16	Software Quality and Defect Estimating	1.00	10.00
17	Software Defect Tracking	1.00	9.75
18	Software Benchmark Overview	1.00	9.75
19	Function Point Analysis: High Speed	1.00	9.75
20	Human Resource Policies	1.00	9.60
21	Software Change Control	1.00	9.50
22	Principles of Software Reuse	1.00	9.40
23	Appraisals and Employee Relations	1.00	9.00
24	Software Cost Tracking	1.00	9.00
25	Software Maintenance & Enhancement	1.00	9.00
26	Methodologies: Agile, RUP, TSP, others	1.00	9.00
27	The Capability Maturity Model (CMMI)	2.00	9.00
28	Overview of Management Tools	1.00	9.00
29	Testing for Project Managers	2.00	8.75
30	Static Analysis for Project Managers	0.50	8.75
31	Inspections for Project Managers	0.50	8.75
32	Project Management Body of Knowledge	1.50	8.70
33	Software Metrics for Project Managers	1.00	8.50
34	Software Cost Estimating: Manual	1.00	8.00
35	Tools: Cost Accounting	1.00	8.00
36	Tools: Project management	1.00	8.00
37	Tools: Human Resources	1.00	8.00
38	Tools: Cost and Quality Estimation	1.00	8.00
39	Function Points for Project Managers	0.50	8.00
40	ISO Standards for Functional Measures	1.00	8.00
41	Principles of Agile for Managers	1.00	7.75
42	Principles of RUP for Managers	1.00	7.75
43	Principles of TSP/PSP for Managers	1.00	7.75
44	Principles of DevOps for managers	1.00	7.75
45	Principles of Containers for managers	1.00	7.75
46	Earned Value Measurement (EVM)	1.00	6.75
47	Principles of Balanced Scorecards	1.00	6.50
48	Six-Sigma for Project Managers	2.00	6.00
49	Six-Sigma: Green belt	3.00	6.00
50	Six-Sigma: Black belt	3.00	6.00
	TOTAL	**60.00**	**8.82**

Needless to say this curriculum would be spread over a multi-year period. It is merely a combination of the kinds of software project management courses available in modern technology companies.

Guest Lectures from Visiting Experts

Over and above software classroom training, some technology companies have occasional internal seminars for all personnel which feature industry experts and famous software researchers. IBM, AT&T, and ITT had large seminars twice a year. One seminar was open only to employees and discussed some proprietary or confidential information such as new products and market expansion. The second large seminar was intended to demonstrate technical excellence to clients and customers, who were also invited to participate.

Among the well-known experts invited to companies such as AT&T, IBM, Siemens, and ITT were Al Albrecht (inventor of function points), Dr. Barry Boehm (inventor of COCOMO), Dr. Fred Brooks, (author of The Mythical Man-Month), Watts Humphrey (creator of the SEI CMMI), Dr. Larry Putnam (inventor of SLIM), Dr. Jerry Weinberg (author of The Psychology of Computer Programming), Bill Gates of Microsoft, Donald Knuth, (pioneer of computer algorithms), Admiral Grace Hopper of the Navy (inventor of COBOL), Ken Thompson (co-developer of UNIX), Linus Torvalds (developer of Linux), and many more.

Usually these events lasted for about half a day of technical topics, and then had either a lunch or a reception for the guests based on whether it was a morning event or an afternoon event. At the reception or lunch the audience could meet and chat informally with the visiting experts.

Because the guest experts were world famous, these corporate seminars were attended by top software executives as well as software engineers and technical staff. They were a good method for bringing together all levels of a company to focus on critical software issues.

Having attended a number of these they were usually very enjoyable and it was good to meet famous software researchers such as Admiral Grace Hopper face to face.

Needless to say this kind of event usually takes place in fairly large companies since they are expensive. However these seminars were also valuable, and benefitted both the executives and technical staffs of IBM, ITT, AT&T, Microsoft, and other companies that have them.

Acquisition and Use of Software Parametric Estimation Tools

IBM discovered in the early 1970's that manual software cost estimates became increasingly optimistic and inaccurate as application size increased from below 100 function points to more than 10,000 function points. Since applications grew rapidly in this era, IBM commissioned Capers Jones to build its first parametric estimation tool in 1973. ITT did the same in 1979 and AT&T commissioned Capers Jones and colleagues to build a custom parametric estimation tool for its electronic switching systems in 1983.

The technology and telecom sectors have been pioneers in the development and usage of parametric estimation tools.

Software has achieved a bad reputation as a troubling technology. Large software projects have tended to have a very high frequency of schedule over-runs, cost overruns, quality problems, and outright cancellations of large systems. While this bad reputation is often deserved, it is important to note that some large software projects are finished on time, stay within their budgets, and operate successfully when deployed.

The successful software projects differ in many respects from the failures and disasters. One important difference is how the successful projects arrived at their schedule, cost, resource, and quality estimates in the first place.

It often happens that projects exceeding their planned schedules or cost estimates did not use state of the art methods for determining either their schedules or their costs. Although the project is cited for overruns, the root problem is inadequate planning and estimation.

From large-scale studies first published in the author's books Software Engineering Best Practices, (Jones 2010) and The Economics of Software Quality, (Jones & Bonsignour 2011) usage of automated parametric estimating tools, automated project scheduling tools, and automated defect and quality estimating tools (all of these are combined in some tools such as Software Risk Master (SRM)) are strongly correlated with successful outcomes.

Conversely, software project failures tended to use casual and manual methods of arriving at initial estimates. Indeed, for many software failures there was no formal estimation at all. In the 15 lawsuits for failure, delays, and poor quality where the author has been an expert witness all of the projects were larger than 10,000 function points and all used manual estimating methods.

The author's book Estimating Software Costs (Jones 2007) discusses the history of the software cost estimating business. His newer book on The Technical and Social History of Software Engineering (Jones 2014) also covered the history and arrival of commercial estimation tools.

The first software parametric cost estimation tools were created by researchers who were employed by large enterprises that built large and complex software systems: IBM, Hughes, RCA, TRW, and the U.S. Air Force were the organizations whose research which led to the development of commercial parametric cost estimating tools.

Some of the estimating pioneers who developed the first parametric estimating tools include in alphabetical order: Dr. Barry Boehm (TRW), Frank Freiman (RCA), Dan Galorath (SEER), Capers Jones (IBM), Dr. Larry Putnam (Air Force), and Dr. Howard Rubin (academic-SUNY).

In 1973 the author and his colleague Dr. Charles Turk at IBM San Jose built IBM's first automated parametric estimation tool for systems software. This tool was called the "Interactive Productivity and Quality" (IPQ) tool. This internal IBM tool was proprietary and not put on the commercial market since it gave IBM competitive advantages. This tool was developed at

IBM's San Jose complex and soon had over 200 IBM users at over 20 locations around the world.

Today in 2017 most technology and telecom companies use parametric estimation tools. These are normally used by project offices or by special estimating teams that provide estimates as a service to specific projects.

Some of the major parametric estimation tools today in 2017 include in alphabetical order:

1. Constructive Cost Model (COCOMO)
2. CostXpert
3. ExcelerPlan
4. KnowledgePlan
5. SEER
6. SLIM
7. Software Risk Master (SRM)
8. True Price

These commercial parametric estimation tools are widely used by technology companies, defense contractors, and other large corporations. They are fairly expensive at costs of over $5,000 per seat for most.

However there is also the constructive cost model (COCOMO) developed by Dr. Barry Boehm. This estimating tool is free. Because COCOMO is free it is widely used by universities and small companies that cannot afford the more expensive commercial estimation tools, but COCOMO does not have as wide usage among U.S. technology companies which need the more detailed estimates provided by the commercial parametric tools.

For example the major features of the author's Software Risk Master (SRM) commercial software estimation tool include:

- Sizing logic for specifications, source code, and test cases
- Sizing logic for function points and lines of code (LOC)
- Sizing logic for story points and use-case points
- Sizing logic for requirements creep during development and post release
- Activity=level estimation for requirements, design, code, testing, etc.
- Sophisticated quality estimates that predict defects and defect removal efficiency (DRE)
- Support for 60 development methods such as agile, containers, DevOps, TSP, spiral, etc.
- Support for development, user costs, and three years of maintenance
- Support for IFPUG function point metrics
- Support for the new SNAP point metric for non-functional requirements
- Support for other function point metrics such as COSMIC, NESMA, FISMA, etc.
- Support for older lines of code (LOC) metrics (both physical and logical)
- Support for modern topics such as cyber-defenses and cyber-attack recovery

- Support for proprietary metrics such as feature points and object points
- Support for software reusability of various artifacts
- Support for 84 modern languages such as Java, Ruby, mySQL, and others
- Support for systems applications such as operating systems and telecom
- Support for IT applications such as finance and insurance
- Support for web-based applications
- Support for cloud applications
- Support for ERP deployment and customization
- Support for including commercial off the shelf software (COTS)
- Support for software portfolio analysis

SRM also supports many advanced functions:

- Quality and reliability estimation
- Numbers of test cases needed and test coverage
- Litigation costs for breach of contract
- Cyber-attack costs and recovery costs
- ERP deployment and modification
- Portfolio sizing and annual portfolio maintenance
- Risk and value analysis
- Measurement modes for collecting historical data
- Cost and time to complete estimates mixing historical data with projected data
- Support for software process assessments
- Support for results from the five levels of the SEI CMMI
- Special estimates such as the odds and costs of outsource litigation
- Special estimates such as venture funding for software startups

The other commercial software parametric estimation tools have similar features. The U.S. Air Force used to perform annual trials of commercial parametric estimation tools. All of the major parametric tools were within about 10% of one another, and all were significantly more accurate than manual estimates for large projects above 1000 function points in size. Manual estimates were often optimistic on cost and schedules by more than 50% above 1000 function points. Parametric estimation tools are almost never optimistic and usually come within 10% of actual results.

Of course another industry problem is that most companies do not have accurate results. Among the author's clients the average accuracy of software project historical data is only 37%. The most common "leaks" from project historical data include business analysts, project managers, software quality assurance, technical writers, project office personnel, configuration control, and integration specialists. Measuring only developers and test personnel such as "design, code, and unit test" or DCUT is a common but fundamentally inaccurate and inadequate for economic or quality analysis.

Acquisition and Use of Progress and Milestone Tracking Tools

In addition to parametric estimation tools there are also many commercial project management tools. The phrase "project management tools" has been applied to a large family of tools whose primary purpose is sophisticated scheduling for projects with hundreds or even thousands of overlapping and partially interdependent tasks and large teams in the hundreds. These tools are able to drop down to very detailed task levels, and can even handle the schedules of individual workers. Microsoft Project and Artemis Views are two samples of project management tools. The new automated project office (APO) tool of Computer Aid is a modern project management tool only recently put on the market. There are also open-source project management and tracking tools such as JIRA.

However, the family of project management tools are general purpose in nature, and do not include specialized software sizing and estimating capabilities as do the software cost estimating tools. Neither do these general project management tools deal with quality issues such as defect removal efficiency. Project management tools are useful, but software requires additional capabilities to be under full management control.

Project management tools are an automated form of several management aids developed by the Navy for controlling large and complex weapons systems: the "program evaluation and review technique" (PERT), critical path analysis, resource leveling, and the classic Gantt charts. Project management tools used for defense projects also support earned-value analysis (EVA) although this is seldom used in the civilian sectors.

Project management tools have no built-in expertise regarding software, as do software cost estimating tools. For example, if you wish to explore the quality and cost impact of an object-oriented programming language such as Objective C a standard project management tool is not the right choice.

By contrast, many software cost estimating tools have built-in tables of programming languages and will automatically adjust the estimate based on which language is selected for the application. Project management tools and software cost estimating tools provide different but complementary functions. Most software cost estimating tools interface directly with common project management tools such as Microsoft Project.

Project Management tools	**Software Cost Estimating tools**
Work-breakdown structures	Sizing logic for function points, code, etc.
Activity-based cost analysis	Quality estimates
Earned-value calculations	Integral risk estimates
Effort by staff members	Staffing predictions (testers, programmers, etc.)

Cost accumulation Cost estimating

Both kinds of tools are useful for large and complex software projects and are generally used concurrently by project office personnel. An average software project using both parametric estimation tools and project management tools would be a significant project of over 1000 function points in size. Small projects below 100 function points are often estimated informally and use only normal corporate cost accounting tools rather than sophisticated project management tools.

The Use of Formal Project Offices (PMO) for Applications > 1,000 function points

When software started most applications were small and below 100 function points in size. During the late 1960's and early 1970's software applications such as operating systems and telephone switching systems grew above 10,000 function points.

These larger applications had development teams that often topped 100 workers and they had over a dozen managers, including some 2nd and 3rd line managers.

Due to the poor training of most software managers in topics such as sizing, planning, estimating, and measurement it soon became obvious that expert help was needed for these critical tasks.

The project office concept is much older than software and appeared in the late 1800's for manufacturing and even for agriculture and commodity training. Many industries used project offices before software became important in the 1950's. IBM was a pioneer in the creation and effective use of software project offices in the late 1960's.

There are a number of levels and kinds of project offices, but for this report the main emphasis is on specific software projects that are fairly large and complex such as 10,000 function points in size.

For these large systems the main roles played by the software project office will be the following:

1. Identifying important standards for the project such as ISO or OMG standards.
2. Identifying important corporate standards such as the IBM standards for software quality.
3. Identifying and monitoring government mandates such as FAA, FDA, or Sarbanes-Oxley.
4. Early sizing of project deliverables using one or more parametric estimation tools.
5. Early prediction of schedules, costs, and staffing using one or more parametric estimation tools.
6. Early prediction of requirements creep.

7. Early prediction of defect potentials and defect removal efficiency (DRE).
8. Establishing project cost and milestone tracking guidelines for all personnel.
9. Continuous monitoring of progress, accumulated costs, and schedule adherence.
10. Continuous monitoring of planned vs. actual defect removal efficiency (DRE).

Project offices are a useful and valuable software structure. They usually contain from 3 to more than 10 people based on the size of the project being controlled. Among the kinds of personnel employed in software project offices are estimating specialists, metric specialists such as certified function point counters, quality control specialists, and standards specialists.

Project offices usually have several parametric estimation tools available and also both general and specialized project tracking tools. Shown below are samples of some of the kinds of tools and information that a project office might utilize in a technology company for a major software application:

Table 7: Tools for Software Projects

	Tasks	Tools Utilized
1	Architecture	**QEMU**
2	Automated test	**HP QuickTest Professional**
3	Benchmarks	**ISBSG, Namcook SRM, Davids, Q/P Management**
4	Coding	**Eclipse, Slickedit**
5	Configuration	**Perforce**
6	Cost estimate	**Software Risk Master (SRM), SLIM, SEER, COCOMO**
7	Cost tracking	**Automated project office (APO), Microsoft Project**
8	Cyclomatic	**BattleMap**
9	Debugging	**GHS probe**
10	Defect tracking	**Bugzilla**
11	Desgn	**Projects Unlimited, Visio**
12	Earned value	**DelTek Cobra**
13	ERP	**Microsoft Dynamics**
14	Function points 1	**Software Risk Master (SRM)**
15	Function points 2	**Function point workbench**
16	Function points 3	**CAST automated function points**
17	Graphics design	**Visio**
18	Inspections	**SlickEdit**
19	Integration	**Apache Camel**
20	ISO tools	**ISOXpress**
21	Maintenance	**Mpulse**
22	Manual test	**DevTest**
23	Milestone track	**KIDASA Softare Milestone Professional**
24	Progress track	**Jira, Automated Project Office (APO)**
25	Project mgt.	**Automated project office (APO)**
26	Quality estimate	**Software Risk Master (SRM)**
27	Requirements	**Rational Doors**

28	Risk analysis	**Software Risk Master (SRM)**
29	Source code size 1	**Software Risk Master (SRM)**
30	Source code size 2	**Unified code counter (UCC)**
31	SQA	**NASA Goddard ARM tool**
32	Static analysis	**OptimMyth Kiuwin, Coverity, Klocwork**
33	Support	**Zendesk**
34	Test coverage	**Software Verify suite**
35	Test library	**DevTest**
36	Value analysis	**Excel and Value Stream Tracking**

ISO and other standards used for project

IEEE 610.12-1990	**Software engineering terminology**
IEEE 730-1999	**Software assurance**
IEEE 12207	**Software process tree**
ISO/IEC 9001	**Software quality**
ISO/IEC 9003	**Software quality**
ISO/IEC 12207	**Software engineering**
ISO/IEC 25010	**Software quality**
ISO/IEC 29119	**Software testing**
ISO/IEC 27034	**Software security**
ISO/IEC 20926	**Function point counting**
OMG Corba	**Common Object Request Broker Architecture**
OMG Models	**Meta models for software**
OMG funct. pts.	**Automated function points (legacy applications)**
UNICODE	**Globalization and internationalization**

Professional certifications used on project

Certification used for project = 1
Certification not used for project = 0

Note: Some team members have multiple certifications.

Certification - Apple	**0**
Certification - Computer Aid Inc.	**0**
Certification - Computer Associates	**0**
Certification - FAA	**0**
Certification - FDA	**0**
Certification - Hewlett Packard	**0**
Certification - IBM	**1**

Certification - Microsoft	**1**
Certification - Oracle	**0**
Certification - PMI	**1**
Certification - QAI	**0**
Certification - Red Hat	**0**
Certification - RSBC	**0**
Certification - SAP	**0**
Certification - Sarbanes-Oxley	**0**
Certification - SEI	**0**
Certification - Sun	**0**
Certification - -Symantec	**0**
Certification - TickIT	**0**
Certification of computing professionals	**0**
Certified configuration management specialist	**1**
Certified function point analyst	**1**
Certified project managers	**1**
Certified requirements engineers	**0**
Certified scrum master	**1**
Certified secure software lifecycle professional	**0**
Certified security engineer	**0**
Certified SEI appraiser	**1**
Certified software architect	**0**
Certified software business analyst	**1**
Certified software development professional	**0**
Certified software engineers	**0**
Certified software quality assurance	**1**
Certified test managers	**0**
Certified testers	**1**
Certified webmaster	**1**
Certified software auditor	**1**
TOTAL	**13**

As can be seen software projects offices add knowledge and rigor to topics where ordinary project managers may not be fully trained or highly experienced.

Use and Measurement of Effective Quality Control Methods

The #1 cost driver for the software industry for more than 50 years has been "the cost of finding and fixing bugs." Since bug repairs are the top cost driver, it is impossible to have an accurate cost estimate without including quality costs. It is also impossible to have an accurate cost estimate, or to finish a project on time, unless it uses state of the art quality control methods.

The #1 reason for software schedule delays and cost overruns for more than 50 years has been excessive defects present when testing starts, which stretches out the planned test duration by over 100% and also raises planned development costs. Without excellent quality control

software projects will always run late, exceed their budgets, and be at risk of cancellation if excessive costs make the ROI a negative value.

It is alarming that in several lawsuits where Capers Jones has been an expert witness depositions showed that project managers deliberately cut back on pre-test removal such as inspections and truncated testing early "in order to meet schedules." From the fact that the projects were late and over budget, cutting back on quality control raised costs and lengthened schedules, but the project managers did not know that this would happen.

Table 8 shows the 15 major cost drivers for software projects in 2017. The cost drivers highlighted in red are attributable to poor software quality:

Table 8: U.S. Software Costs in Rank Order:

1) The cost of finding and fixing bugs
2) The cost of cancelled projects
3) The cost of producing English words
4) The cost of programming or code development
5) The cost of requirements changes
6) The cost of successful cyber-attacks
7) The cost of customer support
8) The cost of meetings and communication
9) The cost of project management
10) The cost of renovation and migration
11) The cost of innovation and new kinds of software
12) The cost of litigation for failures and disasters
13) The cost of training and learning
14) The cost of avoiding security flaws
15) The cost of assembling reusable components

Table 8 illustrates an important but poorly understood economic fact about the software industry. Four of the 15 major cost drivers can be attributed specifically to poor quality. The poor quality of software is a professional embarrassment and a major drag on the economy of the software industry and for that matter a drag on the entire U.S. and global economies.

Poor quality is also a key reason for cost driver #2. A common reason for cancelled software projects is because quality is so bad that schedule slippage and cost overruns turned the project return on investment (ROI) from positive to negative.

Note the alarming location of successful cyber-attacks in 6th place (and rising) on the cost-driver list. Since security flaws are another form of poor quality it is obvious that high quality is needed to deter successful cyber-attacks.

Poor quality is also a key factor in cost driver #12 or litigation for breach of contract. (The author has worked as an expert witness in 15 lawsuits. Poor software quality is an endemic

problem with breach of contract litigation. In one case against a major ERP company, the litigation was filed by the company's own shareholders who asserted that the ERP package quality was so bad that it was lowering stock values!)

If you can't measure a problem then you can't fix the problem either. Software quality has been essentially unmeasured and therefore unfixed for 50 years. A very useful quality metric developed by IBM around 1970 is that of "defect potentials."

Software defect potentials are the sum total of bugs found in requirements, architecture, design, code, and other sources of error. The approximate U.S. average for defect potentials is shown in table 9 using IFPUG function points version 4.3:.

Table 9: Average Software Defect Potentials circa 2017 for the United States

- Requirements 0.70 defects per function point
- Architecture 0.10 defects per function point
- Design 0.95 defects per function point
- Code 1.15 defects per function point
- Security code flaws 0.25 defects per function point
- Documents 0.45 defects per function point
- Bad fixes 0.65 defects per function point
- **Totals 4.25 defects per function point**

Note that the phrase "bad fix" refers to new bugs accidentally introduced in bug repairs for older bugs. The current U.S. average for bad-fix injections is about 7%; i.e. 7% of all bug repairs contain new bugs. For modules that are high in cyclomatic complexity and for "error prone modules" bad fix injections can top 75%. For applications with low cyclomatic complexity bad fixes can drop below 0.5%.

Defect potentials are of necessity measured using function point metrics. The older "lines of code" metric cannot show requirements, architecture, and design defects not any other defect outside the code itself. (As of 2017 function points are the most widely used software metric in the world. There are more benchmarks using function point metrics than all other metrics put together.)

Successful and effective software quality control requires these 15 technical factors:

1. Quality estimation before starting project using parametric estimation tools.
2. Accurate defect tracking from requirements through post-release maintenance.
3. Effective defect prevention such as requirements models and automated proofs.
4. Effective pre-test defect removal such as inspections and static analysis.
5. Effective mathematical test case design using cause-effect graphs or design of experiments.

6. Effective cyber-attack prevention methods such as security inspections.
7. Cyclomatic complexity analysis of all code modules in application.
8. Keeping cyclomatic complexity < 10 for critical software modules.
9. Automated test coverage analysis for all forms of testing.
10. Achieving defect potentials below 3.00 per function point.
11. Achieving > 95% test coverage.
12. Achieving > 97% defect removal efficiency for all applications.
13. Achieving > 99% defect removal efficiency for critical applications.
14. Achieving < 1% bad-fix injection (bad fixes are bugs in bug repairs)
15. Reuse of certified materials that approach zero-defect status.

The bottom line is that poor software quality is the main weakness of the software industry. But poor software quality can be eliminated by better education for project managers and technical staff, and by using quality methods of proven effectiveness. The good news is that high quality is faster and cheaper than poor quality.

Elimination of Bad Metrics and Adoption of Effective Software Metrics

The software industry has the worst metrics and worst measurement practices of any industry in human history. It is one of very few industries that cannot measure its own quality and its own productivity. This is professionally embarrassing.

Some of the troubling and inaccurate metrics used by the software industry are the following:

Cost per defect metrics penalize quality and makes the buggiest software look cheapest. There are no ISO or other standards for calculating cost per defect. Cost per defect does not measure the economic value of software quality. The urban legend that it costs 100 times as much to fix post-release defects as early defects is not true and is based on ignoring fixed costs. Due to fixed costs of writing and running test cases, cost per defect rises steadily because fewer and fewer defects are found. This is caused by a standard rule of manufacturing economics: "*if a process has a high percentage of fixed costs and there is a reduction in the units produced, the cost per unit will go up*." This explains why cost per defects seems to go up over time even though actual defect repair costs are flat and do not change very much. There are of course very troubling defects that are expensive and time consuming, but these are comparatively rare.

Defect density metrics measure the number of bugs released to clients. There are no ISO or other standards for calculating defect density. One method counts only code defects released. A more complete method used by the author includes bugs originating in requirements, architecture, design, and documents as well as code defects. The author's method also includes "bad fixes" or bugs in defect repairs themselves. About 7% of bug repairs contain new bugs. There is more than a 500% variation between counting only released code bugs and counting

bugs from all sources. For example requirements defects comprise about 20% of released software problem reports.

Lines of code (LOC) metrics penalize high-level languages and make low-level languages look better than they are. LOC metrics also make requirements and design invisible. There are no ISO or other standards for counting LOC metrics. About half of the papers and journal articles use physical LOC and half use logical LOC. The difference between counts of physical and logical LOC can top 500%. The overall variability of LOC metrics has reached an astounding 2,200% as measured by Joe Schofield, the former president of IFPUG! LOC metrics make requirements and design invisible and also ignore requirements and design defects, which outnumber code defects. Although there are benchmarks based on LOC, the intrinsic errors of LOC metrics make them unreliable. Due to lack of standards for counting LOC, benchmarks from different vendors for the same applications can contain widely different results. Appendix B provides a mathematical proof that LOC metrics do not measure economic productivity by showing 79 programming languages with function points and LOC in a side-by-side format.

SNAP point metrics are a new variation on function points introduced by IFPUG in 2012. The term SNAP is an awkward acronym for "software non-functional assessment process." The basic idea is that software requirements have two flavors: 1) functional requirements needed by users; 2) non-functional requirements due to laws, mandates, or physical factors such as storage limits or performance criteria. The SNAP committee view is that these non-functional requirements should be sized, estimated, and measured separately from function point metrics. Thus SNAP and function point metrics are not additive, although they could have been. Having two separate metrics for economic studies is awkward at best and inconsistent with other industries. For that matter it seems inconsistent with standard economic analysis in every industry. Almost every industry has a single normalizing metric such as "cost per square foot" for home construction or "cost per gallon" for gasoline and diesel oil. As of 2017 none of the parametric estimation tools have fully integrated SNAP and it may be that they won't since the costs of adding SNAP are painfully expensive. As a rule of thumb non-functional requirements are about equal to 15% of functional requirements, although the range is very wide.

Story point metrics are widely used for agile projects with "user stories." Story points have no ISO standard for counting or any other standard. They are highly ambiguous and vary by as much as 400% from company to company and project to project. There are few if any useful benchmarks using story points. Obviously story points can't be used for projects that don't utilize user stories so they are worthless for comparisons against other design methods.

Technical debt is a new metric and rapidly spreading. It is a brilliant metaphor developed by Ward Cunningham. The concept of "technical debt" is that topics deferred during development in the interest of schedule speed will cost more after release than they would have cost initially. However there are no ISO standards for technical debt and the concept is highly ambiguous. It can vary by over 500% from company to company and project to project. Worse, technical debt

does not include all of the costs associated with poor quality and development short cuts. Technical debt omits canceled projects, consequential damages or harm to users, and the costs of litigation for poor quality.

Use case points are used by projects with designs based on "use cases" which often utilize IBM's Rational Unified Process (RUP). There are no ISO standards for use cases. Use cases are ambiguous and vary by over 200% from company to company and project to project. Obviously use cases are worthless for measuring projects that don't utilize use cases, so they have very little benchmark data. This is yet another attempt to imitate the virtues of function point metrics, only with somewhat less rigor and with imperfect counting rules as of 2015.

Velocity is an agile metric that is used for prediction of sprint and project outcomes. It uses historical data on completion of past work units combined with the assumption that future work units will be about the same. Of course it is necessary to know future work units for the method to operate. The concept of velocity is basically similar to the concept of using historical benchmarks for estimating future results. However as of 2017 velocity has no ISO standards and no certification.

There are no standard work units for velocity and these can be story points or other metrics such as function points or use case points, or even synthetic concepts such as "days per task." If agile projects used function points then they could gain access to large volumes of historical data using activity-based costs; i.e. requirements effort, design effort, code effort, test effort, integration effort, documentation effort, etc. So long as agile continues to use quirky and unstandardized metrics without any certification exams, then agile productivity and quality will continue to be a mystery to clients who will no doubt be dismayed to find as many schedule delays and cost overruns as they had with waterfall.

As already stated in other reports, there are 11 primary metrics and 10 supplementary metrics that allow software projects to be measured with about 1% precision:

The only software metrics that allow quality and productivity to be measured with 1% precision are these 11 primary software metrics and 10 supplemental metrics:

Primary Software Metrics for High Precision

1. Application size in function points including requirements creep.
2. Size of reusable materials (design, code, documents, etc.)
3. Activity-based costs using function points for normalization.
4. Work hours per month including paid and unpaid overtime.
5. Work hours per function point by activity.
6. Function points per month.
7. Defect potentials using function points (requirements, design, code, document, and bad fix defect categories)
8. Defect removal efficiency (DRE) or the percentage of defects removed before release.

9. Delivered defects per function point.
10. Cost of quality (COQ).
11. Total cost of ownership (TCO).

Supplemental Software Metrics for High Precision

1. Software project taxonomy (nature, scope, class type)
2. Occupation groups (business analysts, programmers, testers, managers, QA, etc.)
3. Team experience levels (expert to novice)
4. CMMI level (1 to 5)
5. Development methodology used on application
6. Programming language(s) used on application
7. Complexity levels (problem, data, code complexity)
8. User effort for internal applications
9. Documents produced (type, pages, words, illustrations, etc.)
10. Meeting and communication costs

What is important are the technical features of the metrics themselves and not the numbers of metric users. Even if 50,000 companies use "lines of code" it is still a bad metric that distorts reality and should be viewed as professional malpractice. Following are the characteristics of a sample of current software metrics:

Software Metric Attributes

	Function Points	Lines of Code	Story Points	Use-Case Points
ISO standard?	**Yes**	**No**	**No**	**No**
OMG standard?	**Yes**	**No**	**No**	**No**
Professional associations?	**Yes**	**No**	**No**	**No**
Formal training?	**Yes**	**No**	**No**	**No**
Certification exam?	**Yes**	**No**	**No**	**No**
Automated counting?	**Yes**	**Yes**	**No**	**No**
Required by governments?	**Yes**	**No**	**No**	**No**
Good for productivity?	**Yes**	**No**	**Yes**	**No**
Good for quality?	**Yes**	**No**	**No**	**No**
Good for estimates?	**Yes**	**No**	**Yes**	**No**
Published conversion rules?	**Yes**	**No**	**No**	**No**
Accepted by benchmark groups?	**Yes**	**Yes**	**No**	**No**
Used for IT projects?	**Yes**	**Yes**	**Yes**	**Yes**
Used for web projects?	**Yes**	**Yes**	**Yes**	**Yes**
Used for cloud projects?	**Yes**	**Yes**	**Yes**	**No**

Used for embedded projects?	**Yes**	**Yes**	**No**	**No**
Used for systems software?	**Yes**	**Yes**	**No**	**No**
Used for telecom software?	**Yes**	**Yes**	**No**	**No**
Used for defense software?	**Yes**	**Yes**	**No**	**No**
Productivity Measures				
Activity-based costs?	**Yes**	**No**	**No**	**No**
Requirements productivity?	**Yes**	**No**	**No**	**No**
Design productivity?	**Yes**	**No**	**No**	**No**
Coding productivity?	**Yes**	**Yes**	**No**	**No**
Testing productivity?	**Yes**	**Yes**	**No**	**No**
Quality assurance productivity?	**Yes**	**No**	**No**	**No**
Technical writer productivity?	**Yes**	**No**	**No**	**No**
Project management productivity?	**Yes**	**No**	**No**	**No**
Net productivity of projects	**Yes**	**Yes**	**Yes**	**Yes**
Quality Measures				
Requirements defects?	**Yes**	**No**	**No**	**No**
Architecture defects?	**Yes**	**No**	**No**	**No**
Design defects?	**Yes**	**No**	**No**	**No**
Document defects?	**Yes**	**No**	**No**	**No**
Coding defects?	**Yes**	**Yes**	**No**	**No**
Bad fix defects?	**Yes**	**Yes**	**No**	**No**
Net quality of projects?	**Yes**	**Yes**	**Yes**	**Yes**

As can be seen function point metrics are the only metrics that can be used for all software activities and for both quality and productivity analysis. This paper uses IFPUG function points version 4.3 but other function point variations such as COSMIC, FISMA, and NESMA function points would produce similar but slightly different results.

Commissioning Annual Software Benchmark Studies

Software benchmarks are collections of data on software costs, schedules, staffing, quality, and technology usage that allow companies to compare results against similar projects in other companies. Usually the benchmarks are "sanitized" and do not reveal the names of the companies or projects themselves.

Major corporations should commission software benchmark studies about once a year in order to judge progress. Some companies such as IBM can produce their own internal benchmarks with high accuracy, but still commission external benchmarks to compare results against other companies.

However most companies are incompetent in collecting historical data. Their data leaks and the average is only about 37% complete. This is why self-reported benchmark data often has higher

productivity than benchmarks collected by professional benchmark consultants. Self-reported data "leaks" and is usually incomplete.

A more fundamental problem is that most enterprises simply do not record data for anything but a small subset of the activities actually performed. In carrying out interviews with project managers and project teams to validate and correct historical data, the author has observed the following patterns of incomplete and missing data, using the 25 activities of a standard chart of accounts as the reference model:

Table 10: Gaps and Omissions Observed in Data for a Software Chart of Accounts

Activities Performed	**Completeness of historical data**
1. Requirements	Missing or Incomplete
2. Prototyping	Missing or Incomplete
3. Architecture	Missing or Incomplete
4. Project planning	Missing or Incomplete
5. Initial analysis and design	Missing or Incomplete
6. Detail design	Incomplete
7. Design reviews	Missing or Incomplete
8. Coding	Complete
9. Reusable code acquisition	Missing or Incomplete
10. Purchased package acquisition	Missing or Incomplete
11. Code inspections	Missing or Incomplete
12. Independent verification and validation	Complete
13. Configuration management	Missing or Incomplete
14. Integration	Missing or Incomplete
15. User documentation	Missing or Incomplete
16. Unit testing	Incomplete
17. Function testing	Incomplete
18. Integration testing	Incomplete
19. System testing	Incomplete
20. Field testing	Missing or Incomplete
21. Acceptance testing	Missing or Incomplete
22. Independent testing	Complete
23. Quality assurance	Missing or Incomplete
24. Installation and training	Missing or Incomplete
25. Project management	Missing or Incomplete
26. Total project resources, costs	**Incomplete**

When the author and his colleagues collect benchmark data, we ask the managers and personnel to try and reconstruct any missing cost elements. Reconstruction of data from memory is plainly inaccurate, but it is better than omitting the missing data entirely.

Unfortunately, the bulk of the software literature and many historical studies only report information to the level of complete projects, rather than to the level of specific activities. Such gross "bottom line" data cannot readily be validated and is almost useless for serious economic purposes."

As of 2017 there are about 40 companies and non-profit organizations that perform software benchmarks of various kinds. Some of the many forms of available software benchmarks include:

1. Compensation studies for software occupation groups
2. Voluntary and involuntary attrition of software personnel
3. Customer satisfaction for quality, reliability, etc.
4. Cyber-attack statistics for hacking, denial of service, data theft, etc.
5. Software productivity using function points per month and work hours per function point
6. Software quality using defect potentials in function points and defect removal efficiency (DRE)

Usually software benchmarks are commissioned by individual business units rather than at the corporate level. Some companies spend over $5,000,000 per year on various kinds of benchmark studies, but may not realize this because the costs are scattered across various business and operating units. Table 9 shows some 40 software benchmark organizations circa 2017. The great majority of these are located in the United States, South America, or Europe. Asia is sparse on software benchmarks except for Japan, South Korea, and Malaysia. South America has large benchmark organizations in Brazil, and other more local benchmark groups in Mexico and Peru.

Table 11: Software Benchmark Providers 2017

1	4SUM Partners
2	Bureau of Labor Statistics, Department of Commerce
3	Capers Jones (Namcook Analytics LLC)
4	CAST Software
5	Congressional Cyber Security Caucus
6	Construx
7	COSMIC function points
8	Cyber Security and Information Systems
9	David Consulting Group
10	Economic Research Center (Japan)
11	Forrester Research
12	Galorath Incorporated

13 Gartner Group
14 German Computer Society
15 Hoovers Guides to Business
16 IDC
17 IFPUG
18 ISBSG Limited
19 ITMPI
20 Jerry Luftman (Stevens Institute)
21 Level 4 Ventures
22 Metri Group, Amsterdam
23 Namcook Analytics LLC
24 Price Systems
25 Process Fusion
26 QuantiMetrics
27 Quantitative Software Management (QSM)
28 Q/P Management Group
29 RBCS, Inc.
30 Reifer Consultants LLC
31 Howard Rubin
32 SANS Institute
33 Software Benchmarking Organization (SBO)
34 Software Engineering Institute (SEI)
35 Software Improvement Group (SIG)
36 Software Productivity Research
37 Standish Group
38 Strassmann, Paul
39 System Verification Associates LLC
40 Test Maturity Model Integrated

Benchmarks are hard to do with accuracy, but useful when done well. When done poorly they add to the confusion about software productivity and quality that has blinded the software industry for more than 50 years.

Formal Best Practice Analysis of Software Tools, Methods, and Quality

The software literature has many articles and some books on software best practices. However these usually lack quantitative data. To the author a "best practice" should improve quality or productivity by at least 10% compared to industry averages. A "worst practice" might degrade productivity and quality by 10%.

With over 26,000 projects to examine, the author has published a number of quantitative tables and reports on software best (and worst) practices. Although we have evaluated about 335 methods and practices, the list is too big for convenience. A subset of 115 methods and practices

shows best practices at the top and worst practices at the bottom. Namcook recommends using as many as possible from the top and avoiding the bottom:

Table 12: Software Technology Stack Scoring

	Methods and Practices in Technology Stack	Value Scores
1	**Benchmarks (validated historical data from similar projects)**	**10.00**
2	**Defect potential < 2.5**	**10.00**
3	**Defect removal efficiency DRE > 99%**	**10.00**
4	**Estimates: Activity-based cost estimates**	**10.00**
5	**Estimates: Parametric estimation tools**	**10.00**
6	**Estimates: Total cost of ownership (TCO) cost estimates**	**10.00**
7	**Formal and early quality predictions**	**10.00**
8	**Formal and early risk abatement**	**10.00**
9	**Inspection of all critical deliverables**	**10.00**
10	**Methods: Patterns and > 85% reuse of key deliverables**	**10.00**
11	**Metrics: Defect potential measures**	**10.00**
12	**Metrics: Defect Removal efficiency (DRE) measures**	**10.00**
13	**Metrics: IFPUG function points**	**10.00**
14	**Metrics: SRM pattern matching sizing**	**10.00**
15	**Pre-requirements risk analysis**	**10.00**
16	**Static analysis of all source code**	**10.00**
17	Automated project office (APO)	9.75
18	Metrics: bad-fix injections	9.70
19	Accurate cost tracking	9.50
20	Accurate defect tracking	9.50
21	Accurate status tracking	9.50
22	Estimates: cost of quality (COQ) estimates	9.25
23	Metrics: COSMIC function points	9.25
24	Metrics: FISMA function points	9.25
25	Metrics: NESMA function points	9.25
26	Metrics: Cost of Quality (COQ) measures	9.00
27	Metrics: Defect detection efficiency (DDE) measures	9.00
28	Reusable test materials	9.00
29	SEMAT usage on project	9.00
30	Test coverage > 96%	9.00
31	Defect removal efficiency DRE > 95%	8.75
32	Methods: Disciplined Agile Delivery (DAD)	8.65
33	Mathematical test case design	8.60
34	CMMI 5	8.50
35	Methods: TSP/PSP	8.50
36	Test coverage tools used	8.50
37	Metrics: requirements growth before and after release	8.50

38	Metrics: deferred features	8.50
39	Methods: Containers	8.40
40	Methods: DevOps	8.40
41	Methods: Hybrid: (agile/TSP)	8.25
42	Automated requirements modeling	8.15
43	Methods: Git	8.10
44	Methods: Mashups	8.10
45	Methods: RUP	8.00
46	Methods: Evolutionary Development (EVO)	8.00
47	Metrics: Automated function points	8.00
48	Reusable requirements	8.00
49	Reusable source code	8.00
50	Methods: Hybrid (waterfall/agile	7.80
51	Static analysis of text requirements	7.80
52	Methods: Kanban/Kaizen	7.70
53	Methods: Iterative development	7.60
54	CMMI 4	7.50
55	Methods: Service oriented models	7.50
56	Metrics: Cyclomatic complexity tools	7.50
57	Requirements change tracking	7.50
58	Reusable designs	7.50
59	Automated proofs of correctness	7.50
60	Methods: Continuous development	7.40
61	Methods: Quality Function Deployment (QFD)	7.35
62	CMMI 3	7.00
63	Methods: Joint Application Design (JAD)	7.00
64	Methods: Spiral development	7.00
65	Requirements change control board	7.00
66	Reusable architecture	7.00
67	Reusable user documents	7.00
68	Methods: Extreme Programming	6.90
69	Metrics: FOG/Flesch readability scores	6.85
70	Defect removal efficiency DRE > 90%	6.50
71	Methods: Agile < 1000 function points	6.50
72	Methods: Correctness proofs - automated	6.25
73	Automated testing	6.00
74	Certified quality assurance personnel	6.00
75	Certified test personnel	6.00
76	Defect potential 2.5 to 4.9	6.00
77	Maintenance: data mining	6.00
78	Metrics: Earned value analysis (EVA)	6.00
79	Six-Sigma for software	5.50
80	ISO risk standards	5.00
81	Metrics: Unadjusted function points	5.00
82	ISO quality standards	4.75

83	Maintenance: ITIL	4.75
84	Metrics: Mark II function points	4.00
85	Requirements modeling - manual	3.00
86	Metrics: SNAP non-functional metrics	2.50
87	CMMI 2	2.00
88	Estimates: Phase-based cost estimates	2.00
89	Metrics: Story point metrics	2.00
90	Metrics: Technical debt measures	2.00
91	Metrics: Use case metrics	1.00
92	CMMI 0 (not used)	0.00
93	**CMMI 1**	**-1.00**
94	**Methods: Correctness proofs - manual**	**-1.00**
95	**Test coverage < 90%**	**-1.00**
96	**Benchmarks (unvalidated self-reported benchmarks)**	**-1.50**
97	**Testing by untrained developers**	**-2.00**
98	**Methods: Waterfall development**	**-3.00**
99	**Methods: Agile > 5000 function points**	**-4.00**
100	**Cyclomatic complexity > 20**	**-6.00**
101	**Metrics: No productivity measures**	**-7.00**
102	**Methods: Pair programming**	**-7.50**
103	**Methods: Cowboy development**	**-8.00**
104	**No static analysis of source code**	**-8.00**
105	**Test coverage not used**	**-8.00**
106	**Estimates: Manual estimation > 250 function points**	**-9.00**
107	**Inaccurate defect tracking**	**-9.00**
108	**Metrics: Cost per defect metrics**	**-9.00**
109	**Inaccurate status tracking**	**-9.50**
110	**Defect potential > 5.00**	**-10.00**
111	**Defect removal efficiency DRE < 85%**	**-10.00**
112	**Inaccurate cost tracking**	**-10.00**
113	**Metrics: Lines of code for economic study**	**-10.00**
114	**Metrics: No function point measures**	**-10.00**
115	**Metrics: No quality measures**	**-10.00**

Because new practices come out at frequent intervals, companies need to have a formal method and practice evaluation group. At ITT the Applied Technology Group evaluated existing and commercial tools, methods, and practices. The ITT Advanced Technology Group developed new tools and methods beyond the state of the art.

It is of minor historical interest that the Objective C programming language selected by Steve Jobs for all Apple software was actually developed by Dr. Tom Love and Dr. Brad Cox at the ITT Advanced Technology Group. When Alcatel acquired the ITT telecom research labs the ownership of Objective C was transferred to Dr. Love at his new company.

SUMMARY AND CONCLUSIONS ON SOFTWARE PROJECT MANAGEMENT

Software is viewed by a majority of corporate CEO's as the most troublesome engineering technology of the modern era. It is true that software has very high rates of canceled projects and also of cost and schedule overruns. It is also true that poor project management practices are implicated in these problems.

However some companies have been able to improve software project management and thereby improve software results. These improvements need better estimates, better metrics and measures, and better quality control.

Since academic training in software project management is marginal, the best source of project management training is in-house education in large companies followed by professional education companies such as the Information Technology Metrics and Productivity Institute (ITMPI), and then by non-profit associations such as IFPUG, PMI, ASQ, etc.

Suggested Readings on Software Project Management

Abran, A. and Robillard, P.N.; "Function Point Analysis, An Empirical Study of its Measurement Processes"; IEEE Transactions on Software Engineering, Vol 22, No. 12; Dec. 1996; pp. 895-909.

Abrain, Alain; Software Estimating Models; Wiley-IEEE Computer Society; 2015

Abrain, Alain; Software Metrics and Metrology; Wiley-IEEE Computer Society; 2010

Abrain, Alain; Software Maintenance Management: Evolution and Continuous Improvement; Wiley-IEEE Computer Society, 2008.

Baird, Linda M. & Brennan, M. Carol; Software Measurement and Estimation: A Practical Approach; IEEE Computer Society Press, Los Alamitos, CA; John Wiley & Sons, Hoboken NJ; ISBN 0-471-67622-5; 2006; 257 pages.

Boehm, Barry et al.; Software Cost Estimating with Cocomo II; Prentice Hall, Upper Saddle River NJ; 2000; ISBN-10 0137025769; 544 pages.

Boehm, Barry Dr.; Software Engineering Economics; Prentice Hall, Englewood Cliffs, NJ; 1981; 900 pages.

Brooks, Fred: The Mythical Man-Month, Addison-Wesley, Reading, Mass., 1974, rev. 1995.

Bundshuh, Manfred and Dekkers, Carol; The IT Measurement Compendium; Estimating and Benchmarking Success with Functional Size Measurement; Springer, 2008; ISBN-10 3540681876; 644 pages.

Cohn, Mike; Agile Estimating and Planning; Prentice Hall PTR, Englewood Cliffs, NJ; 2005; ISBN 0131479415.

Crosby, Phil; Quality is Free; New American Library, Mentor Books; New York, NY; 1979, 270 pages.

DeMarco, Tom; Why Does Software Cost So Much?; Dorset House, New York, NY; ISBN 0-9932633-34-X; 1995; 237 pages.

Ebert, Christof; Dumke, Reiner; and Schmeitendorf, Andreas; Best Practices in Software Measurement; Springer, 2004; ISBN-10 3540208674; 344 pages.

Fleming, Quentin W. & Koppelman, Joel M.; Earned Value Project Management; 2nd edition; Project Management Institute, NY; ISBN 10 1880410273; 2000; 212 pages.

Galorath, Daniel D. & Evans, Michael W.; Software Sizing, Estimation, and Risk Management: When Performance is Measured Performance Improves; Auerbach, Philadelphia, AP; ISBN 10-0849335930; 2006; 576 pages.

Gack, Gary; Managing the Black Hole: The Executive's Guide to Project Risk; Business Expert Publishing, 2010.

Garmus, David & Herron, David; Function Point Analysis; Addison Wesley, Boston, MA; ISBN 0-201069944-3; 363 pages; 2001.

Garmus, David; Russac Janet, and Edwards, Royce; Certified Function Point Counters Examination Guide; CRC Press; 2010.

Garmus, David & Herron, David; Measuring the Software Process: A Practical Guide to Functional Measurement; Prentice Hall, Englewood Cliffs, NJ; 1995.

Gilb, Tom and Graham, Dorothy; Software Inspections; Addison Wesley, Reading, MA; 1993; ISBN 10: 0201631814.

Glass, R.L.; Software Runaways: Lessons Learned from Massive Software Project Failures; Prentice Hall, Englewood Cliffs; 1998.

Harris, Michaael; Herron, David; and Iwanicki, Stacia; The Business Value of IT: Managing Risks, Optimizing Performance, and Measuring Results; CRC Press (Auerbach), Boca Raton, FL: ISBN 13: 978-1-4200-6474-2; 2008; 266 pages.

Hill, Peter; Jones Capers; and Reifer, Don; The Impact of Software Size on Productivity; International Software Standards Benchmark Group (ISBSG), Melbourne, Australia, September 2013.

Humphrey, Watts; Managing the Software Process; Addison Wesley, Reading, MA; 1989.

International Function Point Users Group (IFPUG); IT Measurement – Practical Advice from the Experts; Addison Wesley Longman, Boston, MA; 2002; ISBN 0-201-74158-X; 759 pages.

Jacobsen, Ivar, Griss, Martin, and Jonsson, Patrick; Software Reuse - Architecture, Process, and Organization for Business Success; Addison Wesley Longman, Reading, MA; ISBN 0-201-92476-5; 1997; 500 pages.

Jacobsen, Ivar et al; The Essence of Software Engineering; Applying the SEMAT Kernel; Addison Wesley Professional, 2013.

Johnson, James et al; The Chaos Report; The Standish Group, West Yarmouth, MA; 2000.

Jones, Capers; The Technical and Social History of Software Engineering; Addison Wesley 2014.

Jones, Capers and Bonsignour, Olivier; The Economics of Software Quality; Addison Wesley, 2011.

Jones, Capers; Software Engineering Best Practices; McGraw Hill, 2009; ISBN 978-0-07-162161-8660 pages.

Jones, Capers; Applied Software Measurement; McGraw Hill, 3rd edition 2008; ISBN 978-0-07-150244-3; 668 pages; 3rd edition (March 2008).

Jones, Capers; Estimating Software Costs; McGraw Hill, New York; 2007; ISBN 13-978-0-07-148300-1.

Jones, Capers; Program Quality and Programmer Productivity; IBM Technical Report TR 02.764, IBM San Jose, CA; January 1977.

Jones, Capers; *Sizing Up Software*; Scientific American Magazine; New York NY; Dec. 1998, Vol. 279 No. 6; December 1998; pp 104-109.

Jones, Capers; Software Assessments, Benchmarks, and Best Practices; Addison Wesley Longman, Boston, MA, 2000; 659 pages.

Jones, Capers; Conflict and Litigation Between Software Clients and Developers; Version 6; Software Productivity Research, Burlington, MA; June 2006; 54 pages.

Jones, Capers; "*Software Project Management Practices: Failure Versus Success*;" Crosstalk, Vol. 19, No. 6; June 2006; pp 4-8.

Jones, Capers; Patterns of Software System Failure and Success; International Thomson Computer Press, Boston, MA; December 1995; 250 pages; ISBN 1-850-32804-8; 292 pages.

Jones, Capers; "*Why Flawed Software Projects are not Cancelled in Time"*; Cutter IT Journal; Vol. 10, No. 12; December 2003; pp. 12-17.

Kan, Stephen H.; Metrics and Models in Software Quality Engineering, 2nd edition; Addison Wesley Longman, Boston, MA; ISBN 0-201-72915-6; 2003; 528 pages.

Kaplan, Robert S & Norton, David B.; The Balanced Scorecard; Harvard University Press, Boston, MA; ISBN 1591391342; 2004.

Love, Tom; Object Lessons – Lessons Learned in Object-Oriented Development Projects; SIG Books Inc., New York NY; ISBN 0-9627477-3-4; 1993; 266 pages.

McCabe, Thomas J.; "A Complexity Measure"; IEEE Transactions on Software Engineering; December 1976; pp. 308-320.

McConnell, Steve; Software Estimation – Demystifying the Black Art; Microsoft Press, Redmond, Wa; ISBN 10: 0-7356-0535-1; 2006.

Parthasarathy, M.A.; Practical Software Estimation – Function Point Methods for Insourced and Outsourced Projects; Addison Wesley, Boston, MA; ISBN 0-321-43910-4; 2007; 388 pages.

Paulk Mark et al; The Capability Maturity Model; Guidelines for Improving the Software Process; Addison Wesley, Reading, MA; ISBN 0-201-54664-7; 1995; 439 pages.

Pressman, Roger; Software Engineering - A Practitioner's Approach; McGraw Hill, New York, NY; 1982.

Putnam, Lawrence H.; Measures for Excellence – Reliable Software On-Time Within Budget; Yourdon Press, Prentice Hall, Englewood Cliffs, NJ; ISBN 0-13-567694-0; 1992; 336 pages.

Putnam, Lawrence & Myers, Ware; Industrial Strength Software – Effective Management Using Measurement; IEEE Press, Los Alamitos CA; ISBN 0-8186-7532-2; 1997; 320 pages.

Robertson, Suzanne and Robertson, James; Requirements-Led Project Management; Addison Wesley, Boston, MA; 2005; ISBN 0-321-18062-3.

Roetzheim, William H. and Beasley, Reyna A.; Best Practices in Software Cost and Schedule Estimation; Prentice Hall PTR, Saddle River, NJ; 1998.

Royce, Walker; Software Project Management – A Unified Framework; Addison Wesley, 1998.

Strassmann, Paul; Governance of Information Management: The Concept of an Information Constitution; 2nd edition; (eBook); Information Economics Press, Stamford, Ct; 2004.

Strassmann, Paul; The Squandered Computer; Information Economics Press, Stamford, CT; 1997.

Stutzke, Richard D.; Estimating Software-Intensive Systems – Projects, Products, and Processes; Addison Wesley, Boston, MA; ISBN 0-301-70312-2; 2005; 917 pages.

Weinberg, Dr. Gerald; Quality Software Management - Volume 2 First-Order Measurement; Dorset House Press, New York, NY; ISBN 0-932633-24-2; 1993; 360 pages.

Wiegers, Karl A; Creating a Software Engineering Culture; Dorset House Press, New York, NY; 1996; ISBN 0-932633-33-1; 358 pages.

Wiegers, Karl E.; Peer Reviews in Software – A Practical Guide; Addison Wesley Longman, Boston, MA; ISBN 0-201-73485-0; 2002; 232 pages.

Whitehead, Richard; Leading a Development Team; Addison Wesley, Boston, MA; 2001; ISBN 10: 0201675267; 368 pages.

Yourdon, Ed; Outsource – Competing in the Global Productivity Race; Prentice Hall PTR, Upper Saddle River, NJ; ISBN 0-13-147571-1; 2004; 251 pages.

Yourdon, Ed: Death March—The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects, Prentice Hall PTR, Upper Saddle River, N.J., ISBN 0-13-748310-4, 1997.

Suggested Web Sites

http://www.IASAhome.org This is the web site for the non-profit International Association of Software Architects (IASA). Software architecture is the backbone of all large applications. Good architecture can lead to applications whose useful life expectancy is 20 years or more. Questionable architecture can lead to applications whose useful life expectancy is less than 10 years, coupled with increasing complex maintenance tasks and high defect levels. The IASA is working hard to improve both the concepts of architecture and the training of software architects via a modern and extensive curriculum.

http://www.IIBA.org This is the web site for the non-profit International Institute of Business Analysis. This institute deals with the important linkage between business knowledge and software that supports business operations. Among the topics of concern are the Business Analysis Body of Knowledge (BABOK), training of business analysts, and certification to achieve professional skills.

http://www.IFPUG.org This is the web site for the non-profit International Function Point Users Group. IFPUG is the largest software metrics association in the world, and the oldest association of function point users. This web site contains information about IFPUG function points themselves, and also citations to the literature dealing with function points. IFPUG also offers training in function point analysis and administers. IFPUG also administers a certification program for analysts who wish to become function point counters.

http://www.ISBSG.org This is the web site for the non-profit International Software Benchmark Standards Group. ISBSG, located in Australia, collects benchmark data on software projects throughout the world. The data is self-reported by companies using a standard questionnaire. About 4,000 projects comprise the ISBSG collection as of 2007, and the collection has been growing at a rate of about 500 projects per year. Most of the data is expressed in terms of IFPUG function point metrics, but some of the data is also expressed in terms of COSMIC function points, NESMA function points, Mark II function points, and several other function point variants. Fortunately the data in variant metrics is identified. It would be statistically invalid to include attempt to average IFPUG and COSMIC data, or to mix up any of the function point variations.

http://www.iso.org This is the web site for the International Organization for Standardization (ISO). The ISO is a non-profit organization that sponsors and publishes a variety of international standards. As of 2007 the ISO published about a thousand standards a year, and the total published to date is approximately 17,000. Many of the published standards affect software. These include the ISO 9000-9004 quality standards and the ISO standards for functional size measurement.

http://www.namcook.com This web site contains a variety of quantitative reports on software quality and risk factors. It also contains a patented high-speed sizing tool that can size applications of any size in 90 seconds or

less. It also contains a catalog of software benchmark providers which currently lists 20 organizations that provide quantitative data about software schedules, costs, quality, and risks.

http://www.PMI.org This is the web site for the Project Management Institute (PMI). PMI is the largest association of managers in the world. PMI performs research and collects data on topics of interest to managers in every discipline: software, engineering, construction, and so forth. This data is assembled into the well known Project Management Body of Knowledge or PMBOK.

http://www.ITMPI.org This is the web site for the Information Technology Metrics and Productivity Institute. ITMPI is a wholly-owned subsidiary of Computer Aid Inc. The ITMPI web site is a useful portal into a broad range of measurement, management, and software engineering information. The ITMPI web site also provides useful links to many other web sites that contain topics of interest on software issues.

http://www.sei.cmu.edu This is the web site for the Software Engineering Institute (SEI). The SEI is a federally-sponsored non-profit organization located on the campus of Carnegie Mellon University in Pittsburgh, PA. The SEI carries out a number of research programs dealing with software maturity and capability levels, with quality, risks, measurement and metrics, and other topics of interest to the software community.

http://www.stsc.hill.af.mil/CrossTalk This is the web site of both the Air Force Software Technology Support Center (STSC) and also the CrossTalk journal, which is published by the STSC. The STSC gathers data and performs research into a wide variety of software engineering and software management issues. The CrossTalk journal is one of few technical journals that publish full-length technical articles of 4,000 words or more. Although the Air Force is the sponsor of STSC and CrossTalk, many topics are also relevant to the civilian community. Issues such as quality control, estimating, maintenance, measurement, and metrics have universal relevance.

END OF DOCUMENT