VARIATIONS IN SOFTWARE DEVELOPMENT BY FUNCTION POINT SIZE

Version 3.0 – January, 2018

Abstract

Building small applications and building large systems are very different. Small software projects of a 100 function points can be built by a few developers and a few other personnel. These are low in risk and usually delivered on time. Large systems in the 10,000 function point size range require many skilled specialists such as business analysts, architects, and data base analysts who may not be needed for small projects. These are extremely risk and almost never on time or within budget. Worse, quality control is often poor so post-release bugs are troublesome for users.

This paper shows typical results for three size ranges: 100, 1,000, and 10,000 function points. The data comes from benchmark studies by the author and colleagues at Namcook Analytics LLC. Our Software Risk Master (SRM) tool was used to provide the quantitative data in the paper and in the following summary table:

The Impact of Size on Software Project Costs and Schedules

(Assumes IT financial application and Java programming language)

Size in	**Small**	**Medium**	**Large**
Function Points	**100**	**1,000**	**10,000**
Size in LOC	5,330	53,300	533,000
Schedule months	5.75	13.80	33.11
Team staff size	1.25	6.50	66.66
Effort months	7.19	89.72	2,207.00
Function points Per month	13.90	11.15	4.53
Costs	$71,930	$897,250	$22,075,000
$ per function point	$719.30	$897.25	$2,207.50

Overall risks	**7.83%**	**14.60%**	**29.22%**

The differences in function point size leads to very different kinds of development practices and to very different productivity rates at the low end compared to the high end. For example for some large systems finding and fixing bugs and creating paper documents cost more than the code itself.

Successful results for large systems requires early sizing and estimating using tools such as Software Risk Master (SRM) and careful progress and cost tracking using tools such as the Automated Project Office (APO).

Capers Jones
Vice President and CTO, Namcook Analytics LLC

Email: Capers.Jones3@gmail.com
Web www.Namcook.com

Variations in Software Development by Function Point Size

In many industries building large products is not the same as building small products. Consider the differences in specialization and methods required to build a rowboat versus building an 80,000 ton cruise ship.

A rowboat can be constructed by a single individual using only hand tools. But a large modern cruise ship requires more than 350 workers including many specialists such a pipe fitters, electricians, steel workers, painters, and even interior decorators and a few fine artists.

Software follows a similar pattern: Building large systems in the 10,000 to 100,000 function point range is more or less equivalent to building other large structures such as ships, office buildings, or bridges. Many kinds of specialists are utilized and the development activities are quite extensive compared to smaller applications.

Table 1 illustrates the variations in development activities noted for the six size plateaus using the author's 25-activity checklist for development projects:

Table 1: Development Activities for Six Project Size Plateaus

	1	**10**	**100**	**1000**	**10,000**	**100,000**
	Function	**Function**	**Function**	**Function**	**Function**	**Function**
Activities Performed	**Point**	**Points**	**Points**	**Points**	**Points**	**Points**
01 Requirements	X	X	X	X	X	X
02 Prototyping				X	X	X
03 Architecture					X	X
04 Project plans				X	X	X
05 Initial design		X	X	X	X	X
06 Detail design			X	X	X	X
07 Design reviews					X	X
08 Coding	X	X	X	X	X	X
09 Reuse acquisition	X	X	X	X	X	X
10 Package purchase					X	X
11 Code inspections				X	X	X
12 Ind. Verif. & Valid.						
13 Change control				X	X	X
14 Formal integration				X	X	X
15 User documentation			X	X	X	X
16 Unit testing	X	X	X	X	X	X
17 Function testing			X	X	X	X
18 Integration testing				X	X	X
19 System testing				X	X	X
20 Beta testing					X	X
21 Acceptance testing				X	X	X
22 Independent testing						
23 Quality assurance						X

24 Installation/training				X	X	X
25 Project management			X	X	X	X
Activities	4	5	9	18	22	23

Below the plateau of 1000 function points (which is roughly equivalent to 100,000 source code statements in a procedural language such as COBOL) less than half of the 25 activities are normally performed. But large systems in the 10,000 to 100,000 function point range perform more than 20 of these activities.

To illustrate these points table 2 shows quantitative variations in results for three size plateaus, 100, 1,000, and 10,000 function points:

Table 2: Powers of Ten for 100, 1,000, and 10,000 Function Points

Size in Function Points		**100**	**1,000**	**10,000**
Examples		**Medium**	**Smart**	**Local**
		update	**Phone app**	**System**
Team experience		**Average**	**Average**	**Average**
Methodology		**Agile**	**Iterative**	**Hybrid**
Sample size for this table		**150**	**450**	**50**
CMMI levels (0 = CMMI not used)		**0**	**1**	**1**
Monthly burdened costs		**$10,000**	**$10,000**	**$10,000**
Major Cost Drivers (rank order)	**1**	**Coding**	**Bug repairs**	**Bug repairs**
	2	**Bug repairs**	**Coding**	**Paperwork**
	3	**Management**	**Paperwork**	**Coding**
	4	**Meetings**	**Management**	**Creep**
	5	**Paperwork**	**Meetings**	**Meetings**
	6	**0 integration**	**Integration**	**Integration**
	7	**0 creep**	**Creep**	**Management**
Programming language		Java	Java	Java
Source statements per function point		53.00	53.00	53.00
Size in logical code statements (SRM default for LOC)		5,300	53,000	530,000
Size in logical KLOC (SRM default for KLOC)		5.30	53.00	530.00
Size in physical LOC (not recommended)		19,345	193,450	1,934,500

Size in physical KLOC (not recommended)	19.35	193.45	1,934.50
Client planned schedule in calendar months	5.25	12.50	28.00
Actual Schedule in calendar months	5.75	13.80	**33.11**
Plan/actual schedule difference	0.50	1.30	**5.11**
Schedule slip percent	9.61%	10.43%	**18.26%**
Staff size (technical + management)	1.25	6.50	66.67
Effort in staff months	7.19	89.72	2,207.54
Work hours per month (U.S. value)	132	132	132
Unpaid overtime per month (software norms)	0	8	**16**
Effort in staff hours	949.48	11,843.70	291,395.39
IFPUG Function points per month	13.90	11.15	4.53
Work hours per function point	9.49	11.84	29.14
Logical Lines of code (LOC) per month (Includes executable statements and data definitions)	736.83	590.69	240.09
Physical lines of code (LOC) per month (Includes blank lines, comments, headers, etc.)	2,689.42	2,156.03	876.31
Requirements creep (total percent growth)	1.00%	6.00%	**15.00%**
Requirements creep (function points)	1	60	**1,500**
Probable deferred features to release 2	0.00	0.00	**2,500**
Client planned project cost:	**$65,625**	**$812,500**	**$18,667,600**
Actual total project cost	**$71,930**	**$897,250**	**$22,075,408**
Plan/Actual cost difference	**$6,305**	**$84,750**	**$3,407,808**
Plan/Actual percent difference	**8.77%**	**9.45%**	**15.44%**
Planned cost per function point	**$656.25**	**$812.50**	**$1,866.76**
Actual cost per function point	**$719.30**	**$897.25**	**$2,207.54**

Defect Potentials and Removal %

Defect Potentials	**Defects**	**Defects**	**Defects**
Requirements defects	5	445	6,750
Architecture defects	0	1	27
Design defects	25	995	14,700
Code defects	175	2,150	30,500
Document defects	11	160	1,650
Bad fix defects	15	336	3,900
TOTAL DEFECTS	**231**	**4,087**	**57,527**

Defects per function point	**2.31**	**4.09**	**5.75**
Defect removal efficiency (DRE)	**97.50%**	**96.00%**	**92.50%**
Delivered Defects	**6**	**163**	**4,313**
High-severity defects	**1**	**20**	**539**
Security flaws	**0**	**3**	**81**
Delivered Defects per Function Point	**0.06**	**0.16**	**0.43**
Delivered defects per KLOC	**1.09**	**3.08**	**8.14**

Test Cases for Selected Tests	**Test Cases**	**Test Cases**	**Test Cases**
Unit test	101	1,026	10,461
Function test	112	1,137	11,592
Regression test	50	512	5,216
Component test	67	682	6,955
Performance test	33	341	3,477
System test	106	1,080	11,012
Acceptance test	23	237	2,413
TOTAL	**492**	**5,016**	**51,126**
Test cases per function point	**4.92**	**5.02**	**5.11**
Probable test coverage	**95.00%**	**92.00%**	**87.00%**
Probable peak cyclomatic complexity	**12.00**	**15.00**	**> 25.00**

Document Sizing

Document Sizes	**Pages**	**Pages**	**Pages**
Requirements	40	275	2,126
Architecture	17	76	376
Initial design	45	325	2,625
Detail design	70	574	5,118
Test plans			

	23	145	1,158
Development Plans	6	55	550
Cost estimates	17	76	376
User manuals	38	267	2,111
HELP text	19	191	1,964
Courses	15	145	1,450
Status reports	20	119	1,249
Change requests	18	191	2,067
Bug reports	97	1,048	11,467
TOTAL	**423**	**3,486**	**32,638**
Document set completeness	**96.96%**	**91.21%**	**78.24%**
Document pages per function point	**4.23**	**3.49**	**3.26**
Project Risks	**Risk %**	**Risk %**	**Risk %**
Cancellation	8.80%	14.23%	26.47%
Negative ROI	11.15%	18.02%	33.53%
Cost overrun	9.68%	15.65%	34.00%
Schedule slip	10.74%	18.97%	38.00%
Unhappy customers	7.04%	11.38%	34.00%
Litigation	3.87%	6.26%	11.65%
Technical debt/high COQ	5.00%	16.00%	26.21%
Cyber attacks	7.00%	9.75%	15.30%
Financial Risk	9.00%	21.00%	41.00%
High warranty repairs/low maintainability	6.00%	14.75%	32.00%
RISK AVERAGE	**7.83%**	**14.60%**	**29.22%**
Project Staffing by Occupation Group	**100**	**1,000**	**10,000**
Programmers	1.91	6.23	43.53
Testers	1.85	5.66	38.58
Designers	0.51	2.13	18.00
Business analysts	0.00	2.13	9.00
Technical writers	0.44	1.05	7.00
Quality assurance	0.46	0.98	5.00
1st line managers	1.21	1.85	7.13

Data base administration	0.00	0.00	3.68
Project Office staff	0.00	0.00	3.19
Administrative support	0.00	0.00	3.68
Configuration control	0.00	0.00	2.08
Project librarians	0.00	0.00	1.72
2nd line managers	0.00	0.00	1.43
Estimating specialists	0.00	0.00	1.23
Architects	0.00	0.00	0.86
Security specialists	0.00	0.00	0.49
Performance specialists	0.00	0.00	0.49
Function point counters	0.00	0.07	0.49
Human factors specialists	0.00	0.00	0.49
3rd line managers	0.00	0.00	0.36
TOTAL STAFF	**6.37**	**20.11**	**148.42**

As can be seen from Table 2 what happens for a small project of 100 function points can be very different from what happens for a large system of 10,000 function points. Note the presence of many kinds of software specialists at the large 10,000 function point size and their absence for the smaller sizes. As application size in function points goes up a number of problems get worse:

Table 3: Problems of Large Software Applications

1. Requirements completeness declines
2. Requirements changes increase
3. Document volumes grow rapidly
4. Document completeness declines
5. Defect potentials increase
6. Defect removal efficiency (DRE) declines
7. Numbers of test cases increase
8. Test coverage declines
9. Cyclomatic complexity goes up
10. Risks of cancellation and delays go up alarmingly
11. Function point counting costs go up
12. Many large applications don’t use function points

The software industry has done well for small projects but not for large systems. Function point metrics have been widely used for small applications but are seldom used above 10,000 function points due to the high cost and lengthy time interval required. There are several forms of high-speed function points such as pattern matching for new projects and automated counts for legacy applications, but manual counts by certified function point personnel remain the most common.

Summary and Conclusions

There are major differences in software development methods, software staffing, software quality, and software productivity between small applications of 100 function points and large systems of 10,000 function points or more. Small projects are generally successful and have fairly good quality and productivity. Large systems fail more often than they succeed and seldom have good quality and productivity.

REFERENCES AND READINGS

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.

Bogan, Christopher E. and English, Michael J.; Benchmarking for Best Practices; McGraw Hill, New York, NY; ISBN 0-07-006375-3; 1994; 312 pages.

Gack, Gary; Managing the Black Hole: The Executives Guide to Software Project Risk; Business Expert Publishing, Thomson, GA; 2010; ISBN10: 1-935602-01-9.

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

IFPUG Counting Practices Manual, Release 6, International Function Point Users Group, Westerville, OH; April 2015; 105 pages.

Jones, Capers; A Guide to Selecting Software Measures and Metrics; CRC Press, 2017.

Jones, Capers; Software Methodologies, a Quantitative Guide, CRC Press, 2017.

Jones Capers; Quantifying Software: Global and Industry Perspectives; CRC Press, 2017.

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

Jones, Capers; Estimating Software Costs; 2nd edition; McGraw Hill; New York, NY; 2007.

Jones, Capers and Bonsignour, Olivier; The Economics of Software Quality; Addison Wesley, Boston, MA; 2011; ISBN 978-0-13-258220-9; 587 pages.

Jones, Capers; “A Ten-Year Retrospective of the ITT Programming Technology Center”; Software Productivity Research, Burlington, MA; 1988.

Jones, Capers; Applied Software Measurement; McGraw Hill, 3rd edition 2008.

Jones, Capers; Software Engineering Best Practices; McGraw Hill, 1st edition 2010.

Jones, Capers; Assessment and Control of Software Risks; Prentice Hall, 1994; ISBN 0-13-741406-4; 711 pages.

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; Software Assessments, Benchmarks, and Best Practices; Addison Wesley Longman, Boston, MA; 2000 (due in May of 2000); 600 pages.

Jones, Capers; Software Quality – Analysis and Guidelines for Success; International Thomson Computer Press, Boston, MA; ISBN 1-85032-876-6; 1997; 492 pages.

Jones, Capers; The Economics of Object-Oriented Software; Software Productivity Research, Burlington, MA; April 1997; 22 pages.

Jones, Capers; Becoming Best in Class; Software Productivity Research, Burlington, MA; January 1998; 40 pages.

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.

McMahon, Paul; 15 Fundamentals for Higher Performance in Software Development; PEM Systems 2014.

Radice, Ronald A.; High Qualitiy Low Cost Software Inspections; Paradoxicon Publishingl Andover, MA; ISBN 0-9645913-1-6; 2002; 479 pages.

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

Yourdon, Ed; Death March - The Complete Software Developer's Guide to Surviving "Mission Impossible" Projects; Prentice Hall PTR, Upper Saddle River, NJ; ISBN 0-13-748310-4; 1997; 218 pages.