How and Why to use Software Risk Master (SRM)

for Risk, Cost, Schedule, Staffing, and Quality Estimates

Version 5.0 – August 21, 2017

Web: www.Namcook.com

Introduction

Software Risk Master (SRM) is a new kind of software project parametric estimation tool based on pattern matching. SRM includes integral high-speed sizing of applications in less than two minutes. SRM sizes applications in a total of 23 metrics including IFPUG, COSMIC, NESMA, and FISMA function points; SNAP points; story points; use-case points, and both logical and physical lines of code. Because of SRM's integral and automatic sizing, users to not have to know application size in order to get accurate estimates. SRM is the only parametric tool with automatic sizing based on pattern matching. Since SRM sizing can take place prior to full requirements, no other known sizing method can be used as early as SRM sizing.

SRM is designed to be fast and easy to use. Most of the input data is in the form of multiple-choice questions and users only need to pick the answers that match their projects. A few SRM questions require actual information but it is the kind of information that should be known very early even before projects start. For example users are asked to provide a name of for project and to provide a start date for the project being estimated.

SRM Inputs needing user data

SRM questions needing actual data from users are:

1. Estimate *confidentiality level*, if any (**Company Confidential** is default)
2. *Project name* (required)
3. *Project start date* (optional; current date is default)
4. *Project planned delivery date* (optional)
5. *Person providing the data* (required)
6. *Company or organization name* (optional)
7. *Country or countries where project occurs* (optional)
8. *Local monthly costs* for development and maintenance (defaults provided)
9. *Local work hours per month* (SRM provides defaults for 52 countries.)
10. *Unpaid overtime per month* (SRM provides defaults.)
11. *Percentage of reusable materials*, if any (SRM provides defaults)

12. *Probable number of application users* (for maintenance estimates)
13. *Probable number of application installations,* (for maintenance estimates)
14. *Number of maintenance repair sites* (for maintenance estimates)
15. *Currency preferred* – U.S. dollars, Yen, Pounds, and Yuan are supported.
16. *Special costs* – Patents; hiring costs; equipment. overseas travel, litigation, etc.

SRM provides default values for costs, reuse, currency, and probable work hours. Users can replace any or all default values with their own local values.

SRM Multiple-Choice Inputs

The multiple-choice questions provide the inputs that SRM actually uses to create size, schedule, staffing, cost, and quality estimates. The SRM multiple choice questions include the following:

1. Project ***nature*** or whether the project is new or an enhancement.
2. Project ***scope***, which can range from a small module through a large system.
3. Project ***class***, which can be internal software or software to be delivered to customers.
4. Project ***type*** such as a web application, embedded software, commercial software, etc.
5. Project *methodology* such as agile, DevOps, waterfall, hybrid, etc.
6. Project ***programming languages***, such as Ruby, SQL, Java (SRM has 87 languages).
7. Project ***programming language levels*** (SRM provides defaults for all languages.).
8. Project *CMMI level*, from 1 to 5 with 0 if the CMMI is not used at all.
9. Project *team experience* ranging from experts through average to novice.
10. Project *management experience* ranging from experts through average to novice.
11. Project *client experience* ranging from experts through average to novice.
12. Project ***problem complexity*** ranging from simple to very complex.
13. Project ***code complexity*** ranging from simple to very complex.
14. Project ***data complexity*** ranging from simple to very complex.
15. Project planned *defect removal stages* such as static analysis and test types

NOTE: The input factors SRM uses for automatic sizing of software applications are highlighted in **blue**.

From start to finish an SRM estimate usually takes about 10 minutes or less. The first time SRM is used it may take up to 15 minutes, but experienced users can create estimates in perhaps 5 to 7 minutes.

It should be noted that the SRM estimating input questions capture the same kind of data that needs to be collected when projects are completed for historical benchmarks. In fact SRM can operate as both an early estimating tool and also a very accurate benchmark data collection tool.

Experimenting with SRM for Alternate Development Scenarios

Because all of the SRM input information is usually known before projects start, SRM is the only parametric estimation tool that can be used before requirements are fully defined. SRM estimates can be made three to six months earlier than other estimating techniques. This means that SRM can be used several times to try out possible development improvements and alternate scenarios. Some suggested SRM scenario runs would include:

1. Try the project with and without *static analysis*.
2. Try the project with and without *inspections*.
3. Try the project with and without *prototypes*.
4. Try the project with and without *SEMAT*.
5. Try the project with and without *requirements models*.
6. Try the project with *agile, TSP, DevOps, XP, and other methodologies*.
7. Try the project with *higher-level programming languages such as Objective C*.
8. Try the project with higher or lower volumes of *reusable materials*.
9. Try the project with *different skill levels*.
10. Try the project with *different complexity levels*.
11. Try the project with and without *unpaid overtime*.
12. Try the project in *different countries* such as the U.S. or India
13. Try the project with *larger or smaller team sizes*.
14. Try the project with *more or fewer work hours per month*.
15. Try the project with *internal teams or outsource* teams.
16. Try the project with different patterns of *quality control methods*.

SRM is so quick and easy to use that a dozen alternate scenarios can be tried out in less than an hour.

SRM Risk Estimates and Risk Avoidance

As the name implies, SRM is a very accurate and powerful risk prediction tool. SRM's early sizing and early estimating ability make it a very useful tool that can help identify possible risks and can also help to minimize or eliminate critical software risks such as:

1. ***Cancellation of project due to negative return on investment (ROI)***
2. ***Cyber-attacks due to latent security flaws.***
3. ***Schedule delays.***
4. ***Cost overruns.***
5. ***Requirements creep during development.***
6. ***Possible deferred features caused by schedule slips.***
7. ***Poor quality after release.***
8. ***High maintenance costs due to poor quality control.***
9. ***Inadequate quality control during development..***
10. ***Unhappy clients and unhappy executives.***
11. ***Possible breach of contract outsource litigation.***
12. ***Negative ROI due to delays and cost overruns.***

The goal of SRM is to give project managers, developers, and executives an accurate early picture of risks, schedules, effort, costs, risks, and quality before any money is spent on projects that might be troublesome.

The SRM risk analysis features can also predict these risks very early before projects start and long before serious money is spent on projects that may prove hazardous. By changing input assumptions it is also possible to examine risk reduction scenarios and see the benefits of better quality control or more reusable materials.

Observations on Getting Started with SRM

Before using SRM to estimate new projects it is useful (and enjoyable) to use SRM to model a sample of projects that have already been finished to see how close the SRM estimates are to reality.

Of course for the comparisons to be valid, the historical data needs to be accurate. This is not as easy as it sounds and most "historical data" is incomplete and leaves out key factors such as unpaid overtime, software reuse, and the work of many specialists such as business analysts, technical writers, and quality assurance. These are of course included in SRM. Some SRM estimates will look more expensive than historical data but this is usually due to errors and omissions in the historical data itself.

A few observations on SRM usage patterns are these:

- Try doing two or three estimates for the same project and change the inputs.
- Be cautious not to exaggerate complexity, which is a common issue with early users.

The "big ticket" factors that have a large impact on software project results are the following:

1. *Software reuse percent* – higher reuse means higher productivity, shorter schedules.
2. *Application size* – costs, schedules, risks go up with application size.
3. *Problem, data, and code complexity* – they affect size, schedules, costs, quality.
4. *Local work hours per month* – monthly hours range from 115 to 200 by country.
5. *Unpaid overtime per month* – Unpaid overtime ranges from 0 to 30 hours per month..
6. *Team experience* – experts have better results than novices.
7. *Management experience* – poor management cause major problems.
8. *Client experience* – Novice clients lead to unstable requirements.
9. *Programming languages* – high-level languages do help.
10. *Methodologies* – agile is usually better than waterfall but lags TSP.
11. *CMMI levels* – higher CMMI levels have higher quality.
12. *Quality control* – static analysis, inspections improve quality and lowers costs.
13. *Non-functional requirements* as measured via SNAP metrics.

We hope you will enjoy the experience of using SRM. We at Namcook Analtyics LLC would be pleased to have you contact us about any questions or about your SRM experiences. Thank you for considering Software Risk Master (SRM).

Appendix 1: Examples of 100 Applications Sized using Software Risk Master

Because application size is a key input factor for all parametric estimation tools and for manual estimates as well, the SRM early and rapid sizing feature based on pattern matching is the only known size method that can size any kind of application between a range of 1 function point and over 300,000 function points. Table 1 is a sample of 100 applications sized using SRM in about three hours. Application sizing via pattern matching averages about 1.8 minutes per application sized.

Note that SRM does not need application requirements or application design information. SRM sizing is based on external patterns of project nature, scope, class, type, and complexity. The SRM pattern matching approach is similar to the pattern matching logic used by the Trulia and Zillow real-estate search tools and by the Kelley Blue Book for searching automobile cost information. SRM is the first tool to apply pattern matching to software sizing.

Table 1: Sizes of 100 Software Applications

	Applications *NOTE: SRM sizing takes about 1.8 minutes per application for sizing (patent-pending).*	Size in Function Points IFPUG 4.3	SNAP Non-function Points IFPUG	Size in Logical Code Statements	SNAP Percent
1	IBM Future System FS/1 (circa 1985 not completed)	515,323	108,218	68,022,636	21.00%
2	Star Wars missile defense	352,330	68,800	32,212,992	19.53%
3	World-wide military command and control (WWMCCS)	307,328	65,000	28,098,560	21.15%
4	U.S. Air Traffic control	306,324	70,133	65,349,222	22.90%
5	Israeli air defense system	300,655	63,137	24,052,367	21.00%
6	North Korean Border defenses	273,961	50,957	25,047,859	18.60%
7	Iran's air defense system	260,100	46,558	23,780,557	17.90%
8	SAP	253,500	32,070	18,480,000	12.65%
9	Aegis destroyer C&C	253,088	49,352	20,247,020	19.50%
10	Oracle	229,434	29,826	18,354,720	13.00%
11	Windows 10 (all features)	198,050	21,786	12,675,200	11.00%
12	Obamacare web site (all features)	107,350	33,450	12,345,250	31.16%
13	Microsoft Office Professional 2010	93,498	10,285	5,983,891	11.00%
14	Airline reservation system				23.18%

		38,392	8,900	6,142,689	
15	North Korean Long-Range Missile controls	37,235	4,468	5,101,195	12.00%
16	NSA code decryption	35,897	3,590	3,829,056	10.00%
17	FBI Carnivore	31,111	2,800	3,318,515	9.00%
18	FBI fingerprint analysis	25,075	3,260	2,674,637	13.00%
19	NASA space shuttle	23,153	3,010	2,116,878	13.00%
20	VA Patient monitoring	23,109	6,500	4,929,910	28.13%
21	Data Warehouse	21,895	2,846	1,077,896	13.00%
22	NASA Hubble controls	21,632	2,163	1,977,754	10.00%
23	Skype	21,202	3,392	1,130,759	16.00%
24	Shipboard gun controls	21,199	4,240	1,938,227	20.00%
25	American Express billing	20,141	4,950	1,432,238	24.58%
26	M1 Abrams battle tank operations	19,569	3,131	1,789,133	16.00%
27	Apple I Phone v6 oprations	19,366	2,518	516,432	13.00%
28	IRS income tax analysis	19,013	5,537	1,352,068	29.12%
29	Cruise ship navigation	18,896	2,456	1,343,713	13.00%
30	MRI medical imaging	18,785	2,442	1,335,837	13.00%
31	Google search engine	18,640	2,423	1,192,958	13.00%
32	Amazon web site	18,080	2,350	482,126	13.00%
33	State wide child support	17,850	4,125	952,000	23.11%
34	Linux	17,505	2,276	700,205	13.00%
35	FEDEX shipping controls	17,378	4,500	926,802	25.90%
36	Tomahawk cruise missile	17,311	2,250	1,582,694	13.00%
37	Denver Airport luggage (original)	17,002	2,166	1,554,497	12.74%
38	Inventory management	16,661	2,111	1,332,869	12.67%
39	EBAY transaction controls	16,390	2,110	1,498,554	12.87%
40	Patriot missile controls	16,239	2,001	1,484,683	12.32%
41	IBM IMS data base	15,392	1,939	1,407,279	12.60%

42	Toyota robotic manufacturing	14,912	1,822	3,181,283	12.22%
43	Android operating system	14,019	1,749	690,152	12.48%
44	Quicken 2015	13,811	1,599	679,939	11.58%
45	State transportation ticketing	12,300	1,461	656,000	11.88%
46	State Motor vehicle registrations	11,240	3,450	599,467	30.69%
47	Insurance claims handling	11,033	2,567	252,191	23.27%
48	SAS statistical package	10,927	1,349	999,065	12.35%
49	Oracle CRM Features	10,491	836	745,995	7.97%
50	DNA Analysis	10,380	808	511,017	7.78%
51	EZPass vehicle controls	4,751	1,300	253,400	27.36%
52	Cat scan medical device	4,575	585	244,000	12.79%
53	Chinese submarine sonar	4,500	522	197,500	11.60%
54	Microsoft Excel 2007	4,429	516	404,914	11.65%
55	Citizens bank on-line	4,017	1,240	367,224	30.87%
56	MapQuest	3,969	493	254,006	12.42%
57	Bank ATM controls	3,917	571	208,927	14.58%
58	NVIDIA graphics card	3,793	464	151,709	12.23%
59	Lasik surgery (wave guide)	3,625	456	178,484	12.58%
60	Sun D-Trace utility	3,505	430	373,832	12.27%
61	Microsoft Outlook	3,450	416	157,714	12.06%
62	Microsoft Word 2007	3,309	388	176,501	11.72%
63	Adobe Illustrator	2,507	280	178,250	11.17%
64	SpySweeper antispyware	2,227	274	109,647	12.30%
65	Norton anti-virus software	2,151	369	152,942	17.16%
66	Microsoft Project 2007	2,108	255	192,757	12.10%
67	Microsoft Visual Basic	2,068	247	110,300	11.94%
68	All-in-one printer	1,963	231	125,631	11.77%
69	AutoCAD	1,900	230	121,631	12.10%

70	Garmin hand-held GPS	1,858	218	118,900	11.73%
71	Intel Math function library	1,768	211	141,405	11.94%
72	PBX switching system	1,658	207	132,670	12.48%
73	Motorola cell phone contact list	1,579	196	144,403	12.41%
74	Seismic analysis	1,564	194	83,393	12.41%
75	Sidewinder missile controls	1,518	188	60,730	12.38%
76	Apple I Pod	1,507	183	80,347	12.15%
77	Property tax assessments	1,492	457	136,438	30.62%
78	Mozilla Firefox (original)	1,450	174	132,564	12.00%
79	Google Gmail	1,379	170	98,037	12.33%
80	Digital camera controls	1,344	167	286,709	12.43%
81	IRA account management	1,340	167	71,463	12.46%
82	Consumer credit report	1,332	345	53,288	25.90%
83	Sun Java compiler	1,310	163	119,772	12.44%
84	All in one printer driver	1,306	163	52,232	12.48%
85	Laser printer driver	1,285	162	82,243	12.61%
86	Microsoft C# compiler	1,281	162	91,096	12.65%
87	Smart bomb targeting	1,267	150	67,595	11.84%
88	Wikipedia	1,257	148	67,040	11.77%
89	Cochlear implant (embedded)	1,250	135	66,667	10.80%
90	Casio atomic watch with compass, tides	1,250	129	66,667	10.32%
91	APAR analysis and routing	1,248	113	159,695	9.06%
92	Computer BIOS	1,215	111	86,400	9.14%
93	Automobile fuel injection	1,202	109	85,505	9.07%
94	Anti-lock brake controls	1,185	107	63,186	9.03%
95	Ccleaner utility	1,154	103	73,864	8.92%
96	Hearing aid (multi program)	1,142	102	30,448	8.93%
97	LogiTech cordless mouse	1,134	96	90,736	8.46%

98	Instant messaging	1,093	89		77,705	8.14%
99	Twitter (original circa 2009)	1,002	77		53,455	7.68%
100	Denial of service virus	866		-	79,197	0.00%
	Averages	**42,682**	**7,739**		**4,250,002**	**14.46%**

Note: sizes assume IFPUG 4.3

Note: All sizes predicted by Software Risk Master (SRM)

Appendix 2: Examples of SRM Estimating Outputs

SRM estimates are fast and easy to perform but quite extensive. SRM estimates include:

1. Risk estimates
2. Development estimates
3. User effort estimates
4. Quality and reliability estimates
5. Three-year maintenance, enhancement, and support estimates
6. Total cost of ownership (TCO) estimates

A few sample topics are shown in this Appendix to give readers an understanding of what SRM can predict. These samples are for a manufacturing order-entry application of 2,500 function points, 325 SNAP points, and 133,333 Java code statements. The team experience is average and the development methodology is agile. Only a few excerpts of a full SRM estimates are shown here.

Note that the same information that SRM predicts before projects start are also used when Namcook Analytics consultants perform benchmark studies for clients. The same topics used by SRM are important for both early estimation and for historical benchmarks.

Risk Predictions

Risks are the first topic predicted by Software Risk Master (SRM). The risk estimates are based on a combination of application size, team experience, methodology, and quality control.

Risks	
Cancellation	**17.18%**
Negative ROI	**21.76%**
Cost overrun	**18.90%**
Schedule slip	**22.91%**
Unhappy customers	**13.74%**
Litigation	**7.56%**
Average Risks	**17.01%**
Financial Risk	**31.72%**

The financial risks are the weighted combination of negative ROI, cost overruns, and schedule slips. Cost overruns and schedule slips are endemic problems of the software industry for applications larger than about 1000 function points in size.

Development Estimates

Because SRM is intended to be used early before project requirements are fully known, SRM uses a set of seven important development activities:

	Staffing	Effort Months	Schedule Months	Project Costs
Requirements	5.10	10.07	1.97	$151,005
Design	6.64	16.11	2.43	$241,608
Coding	15.42	105.86	6.86	$1,587,959
Testing	13.67	80.22	5.87	$1,203,313
Documentation	2.80	7.32	2.61	$109,822
Quality Assurance	2.34	9.15	3.91	$137,277
Management	2.59	32.21	16.12	$483,217
Totals	**16.19**	**260.95**	**23.66**	**$3,914,201**
Function points per staff month:		**9.58**	23.66	Without overlap
			68.1%	Overlap percent
			SRM	
			16.12	**prediction**
			16.40	**Client Plan**
			-0.29	Difference

As can be seen this project had a net productivity rate of 9.58 function points per month. SRM also shows productivity for each activity and productivity using work hours per function point but these are not shown here.

User Effort Estimates

User effort is only important for internal projects where future users of the software are involved with development. User effort is NOT a factor for embedded software such as auto fuel injection or commercial packages with millions of users. However SRM predicts user effort and costs in the interest of showing clients the full costs of software.

User Activities	Staffing	Schedule	Effort	Costs
User requirements team:	3.85	8.06	44.77	$604,407
User architecture team	2.78	1.93	5.37	$75,529
User planning/estimating team	1.89	1.69	5.13	$69,232
User prototype team:	3.13	4.03	12.59	$169,989
User design review team	5.00	3.22	16.12	$217,587
User change control team:	3.33	16.12	53.73	$725,288
User governance team	2.56	10.48	26.86	$362,644
User document review team	6.67	2.01	13.43	$181,322
User acceptance test team:	5.56	2.42	13.43	$181,322
User installation team:	4.35	1.21	5.26	$70,952
Subtotal	**4.20**	**5.12**	**196.69**	**$2,755,733**
Number of Initial Year 1 Users:	100			
Number of users needing training:	100		5.00	$67,500
TOTAL USER COSTS			**201.69**	**$2,722,773**
User function points per staff month				**12.71**
User work hours per function point				**10.39**
User percent of Development Costs				**69.56%**

As can be seen from this example user costs for this internal project were equal to almost 70% of the development team costs. This is too big a value to ignore and leave out of an accurate cost estimate. User costs are extremely important for government software applications for Federal, state, and municipal government agencies. In fact government user costs are often in the range of 90% of development costs. These costs are far to large to leave out of software cost estimates. Surprisingly SRM is the only parametric estimation tool in 2017 that can predict user costs.

We are also the only benchmark organization that collects user-cost benchmark data for development and operational expenses. The sum total of user costs for both development and operation is often larger than the personnel costs for the development and maintenance teams. For CEO and C-level understanding of software economics, user costs need to be part of the cost estimates and part of the historical benchmarks. This is especially true for government software where user costs are always greater than for ordinary commercial companies.

Quality Estimates

Because the costs of finding and fixing bugs is the #1 cost driver for the entire software industry, quality predictions are very important. SRM predicts defect prevention, pre-test defect removal, and all common forms of testing.

Software Quality

Defect Potentials	Potential
Requirements defect potential	1,366
Design defect potential	1,743
Code defect potential	2,692
Document defect potential	262
Total Defect Potential	**6,063**
Per function point	2.43
Per KLOC	45.47

Defect Prevention	Efficiency	Remainder	Bad Fixes
JAD	25%	4,578	30
QFD - not used	0%	4,607	0
Prototype	20%	3,685	18
Models - not used	0%	3,704	0
Subtotal	39%	3,704	47

Pre-Test Removal	Efficiency	Remainder	Bad Fixes
Desk check	26%	2,741	55
Pair programming - not used	0%	2,795	56
Static analysis	55%	1,283	26
Inspections	90%	131	3
Subtotal	96%	133	139

Test Removal	Efficiency	Remainder	Bad Fixes
Test Planning			
Unit	31%	92	2
Function	34%	62	2
Regression	13%	56	1
Component	31%	39	2
Perfomance	12%	36	1
System	35%	24	1
Acceptance	16%	21	0
Subtotal	**84%**	**21**	**9**

Defects delivered	**21**	
High severity	**3**	
Security flaws	**1**	
High severity %	**13.05%**	
Deliv. Per FP	**0.009**	
High sev per FP	**0.001**	
Security flaws per FP	**0.001**	
Deliv. Per KLOC	**0.160**	
High sev per KLOC	**0.021**	
Security flaws per KLOC	**0.009**	
Cumulative Removal Efficiency	**99.65%**	**Excellent**

SRM also predicts test cases and test scripts. SRM predicts cost of quality (COQ) and also technical debt but those are not all shown here. In this project quality control costs were over 45% of total development costs.

		Percent of Development
Defect prevention costs	$189,849	4.85%
Pre-test defect removal costs	$490,909	12.54%
Test defect removal costs	$1,088,484	27.81%
Total Defect removal costs	**$1,769,242**	**45.20%**
Defect removal per function point	$707.70	
Defect removal per KLOC	$13,270.14	
Defect removal FP per staff month	**21.19**	
Defect removal work hours per FP	**6.23**	
Percent of development costs	**45.20%**	

One of the key reasons for SRM quality predictions is because high quality projects are actually faster and cheaper than poor quality projects. Not everyone knows this important fact. Effective defect prevention and pre-test defect removal such as static analysis lowers testing costs and shortens development schedules.

Maintenance, Enhancement, Customer Support, Total Cost of Ownership (TCO)

Although SRM predicts three years of maintenance, enhancements, and customer support these are not shown here since they are fairly extensive. Instead this appendix only shows total cost of ownership (TCO) which combines development, user effort, and the three-year maintenance predictions.

(TCO time period is from calendar year 2011 through year end 2015)

	Staffing	Effort	Costs	$ per FP at release	% of TCO
Development	7.48	260.95	$3,914,201	$1,565.68	46.17%
Enhancement	2.22	79.75	$897,169	$358.87	10.58%
Maintenance	2.36	85.13	$877,561	$351.02	10.35%
Support	0.34	12.29	$58,062	$23.22	0.68%
User costs	4.20	196.69	$2,722,773	$1,089.11	32.12%
Additional costs			$7,500	$3.00	0.09%
Cyber attacks (if any)	**0.00**	**0.00**	**$0**	**$0.00**	**0.00%**
Litigation (if any)	**0.00**	**0.00**	**$0**	**$0.00**	**0.00%**
Total TCO	**16.60**	**634.81**	**$8,477,266**	**$3,390.91**	**100.00%**

Function points after 3 years	3,149
SNAP after 3 years	409
Lines of code after 3 years	167,936
TCO function points/staff month	**4.96**
TCO work hours per function point	**26.61**
TCO cost per function point	**$2,692.05**

Note 1: SRM can estimate *cyber-attack defenses* and also *cyber-attack recovery costs*: both are important in 2017. SRM also has a special estimating feature for the cost of litigation for breach of contract, which is distressingly common in outsource contracts.

Several other special features not shown in this report include the costs ERP deployment and customization, and the venture capital investments needed for start-up software companies including equity dilution. So many start-up companies run out of money and fail that it is useful to have a predictive tool that can estimate three or four rounds of angel and venture capital funding to alert entrepreneurs that costs are likely to be higher than anticipated.

(This venture feature was used on the notorious Studio 38 software company in Rhode Island after it went bankrupt. SRM predicted that in spite of over $100,000,000 in funding by the State

of Rhode Island there had been no funds provided for maintenance or future releases of the Studio 38 game software. SRM predicted that another $60,000,000 would have been needed to keep Studio 38 in business long enough to become profitable.

If SRM had been used before the State voted to fund Studio 38 the SRM prediction of an 88% chance of failure might have saved the state from a very poor investment that has cost Rhode Island taxpayers over 100 million dollars for zero value.)

Note 2: The row labeled "*additional costs*" are for expenses outside the scope of parametric estimation such as the costs of filing patents for novel software features, the costs of acquiring special hardware devices, or the costs of travel to client sites or off-shore development locations. If these expenses occur users can include them in the overall TCO calculations but they are not predictable by SRM itself, or by any other parametric estimation tools. The essential idea of SRM is to try and either predict or include all costs that have as much as a 1% impact on overall application costs.

As can be seen from these partial examples Software Risk Master (SRM) combines speed and ease of use with extensive estimates of critical software factors that project managers and C-level executives should know before projects start to spend serious money and possibly get in trouble.

SRM is a powerful diagnostic tool that can provide early warnings of potential hazards while there is still plenty of time to correct the issues and get the projects on safe tracks to on-time and within-budget completion.

Multiple Metric Sizing with SRM

In addition to having the earliest and fastest known sizing method for software applications, SRM is also the most versatile since it predicts application size in 23 metrics as of 2017. These are the most common metrics used by the software industry.

It is an unfortunate sociological problem that the software industry has more variations of common topics than any other industry in human history. As of 2017 software has over 3,000 programming languages when only a few are widely used; software has over 60 development methodologies when only a few are effective; and software has over 40 size metrics when a single metric, function points, would be sufficient.

All of these alternatives are because software has the worst measurement practices of any industry in human history. Software is almost the only industry where a majority of companies and managers don't know either their productivity rates or their quality levels. Shown below are the 23 metrics SRM uses for every estimate. IFPUG is the default metric.

	Alternate Metrics	**Size**	**% of IFPUG**
1	**IFPUG 4.3**	**2,500**	**100.00%**
2	Automated code based	2,675	107.00%
3	Automated UML-based	2,575	103.00%
4	Backfired function points	2,500	100.00%
5	Cosmic function points	2,714	108.57%
6	Fast function points	2,425	97.00%
7	Feature points	2,500	100.00%
8	FISMA function points	2,550	102.00%
9	Full function points	2,925	117.00%
10	Function points light	2,413	96.50%
11	IntegraNova models	2,725	109.00%
12	Mark II function points	2,650	106.00%
13	NESMA function points	2,600	104.00%
14	RICE objects	11,071	442.86%
15	SCCQI function points	7,193	287.71%

16	Simple function points	2,438	97.50%
17	SNAP non functional metrics	300	12.00%
18	**SRM pattern matching**	2,500	100.00%
19	Story points	833	33.33%
20	Unadjusted function points	2,225	89.00%
21	Use case points	500	20.00%
		Code	**LOC per FP**
22	Logical code statements	133,333	53.33
23	Physical LOC (blank,comments)	538,888	215.56

There are several kinds of automatic function point metrics shown above. These are interesting and speed up counting compared to traditional manual counts. However they only work on existing applications and can't be used for sizing future projects. Also, automatic function points can only be used with the knowledge and permission of the software owner.

By comparison SRM sizing works on both new future applications and also existing applications. SRM sizing can be done on any software so long as the SRM user can answer the sizing input questions of nature, scope, class, type, programming languages, and complexity.

This means that SRM can size classified military software applications, commercial software such as Windows, Android, and Safari, and also software being developed in hostile countries such as North Korea. SRM is also faster than any other form of sizing since it averages only about 1.8 minutes to size any known software application.