

Example 11: How Software Risk Master (SRM) Evaluates Software Complexity

Java Language for all 3 Cases

Iterative development for all 3 cases

High, Average, and Low Complexity Levels

\$7,500 per month for all 3 Cases

Problem complexity = difficulty of algorithms and logic

Code complexity = difficulty due to branches and poor comments (cyclomatic complexity)

Data complexity = number and interactions among files and data sets

Complexity varies with team and personal experience levels

A project can be very complex to novices; very simple to experts

High complexity increases function points; degrades quality; lowers productivity

2017 is the 30th anniversary of IFPUG function point metrics

| | High Complexity | Average Complexity | Low Complexity | |
|----------------------|--------------------|-----------------------|-------------------|--|
| | | | | Complexity scores = 1 (minimum) to 11 (maximum) |
| Problem complexity | 9 | 6 | 3 | Logical difficulty of application |
| Code Complexity | 9 | 6 | 3 | Cyclomatic complexity of code |
| Data complexity | 9 | 6 | 3 | Number of files, data elements, relationships |
| Function Points | 1,120 | 1,000 | 880 | High complexity increases function points |
| Language level | 5.50 | 6.00 | 6.50 | |
| LOC per FP | 58.18 | 53.33 | 49.23 | High complexity increases LOC per function point |
| Logical code lines | 58,182 | 53,333 | 49,231 | High complexity increases code size |
| Project Risks | | | | |
| Cancellation | 24.37% | 13.20% | 9.85% | |
| Negative ROI | 30.87% | 16.74% | 12.48% | |
| Cost overrun | 26.81% | 14.79% | 10.84% | |

| | | | | | |
|--|-------------------------------------|---------------|---------------|--------------|--|
| | Schedule slip | 32.50% | 17.95% | 13.14% | |
| | Unhappy customers | 19.50% | 11.44% | 7.88% | |
| | Litigation | 10.72% | 5.82% | 4.33% | |
| | Technical debt/high COQ | 27.40% | 14.88% | 11.08% | |
| | Cyber attacks | 16.70% | 9.07% | 6.75% | |
| | Financial Risk | 35.97% | 19.53% | 14.54% | |
| | High warranty repairs | 25.26% | 13.72% | 10.21% | |
| | Poor maintainability | 18.84% | 10.23% | 7.62% | |
| | RISK AVERAGE | 24.45% | 13.40% | 9.88% | High complexity increases all risks |
| | | | | | |
| | | | | | |
| | Total Defects in Application | 6,600 | 3,000 | 2,100 | High complexity raises defect potentials |
| | | | | | |
| | Pre-Test Defect Removal % | 42.00% | 70.00% | 84.00% | High complexity lowers defect removal efficiency |
| | Defects Removed | 2,772 | 2,100 | 1,764 | |
| | Defects Remaining | 3,828 | 900 | 336 | |
| | | | | | |
| | Test Defect Removal % | 66.00% | 82.00% | 93.00% | |
| | Defects Removed | 2,526 | 738 | 312 | |
| | Defects Remaining | 1,302 | 162 | 24 | |
| | | | | | |
| | Bad fix injection % | 11.00% | 7.00% | 4.00% | High complexity raises bad-fix injection rates |
| | Bad fixes (new bugs in repairs) | 143 | 11 | 1 | |
| | | | | | |
| | Defects detected but not repaired | | | | |
| | prior to delivery to customers | 433 | 35 | 2 | High complexity increases unrepaired defects |

| | | | | | |
|--|--|--------------------|------------------|------------------|--|
| | | | | | |
| | | | | | |
| | Cumulative Defect Removal % | 80.28% | 94.60% | 98.88% | All projects should top 96% defect removal efficiency (DRE) DRE developed by IBM circa 1973 |
| | Total Defects Removed | 5,298 | 2,838 | 2,076 | |
| | | | | | |
| | Total Defects Delivered | 1,445 | 173 | 24 | High complexity increases delivered defects |
| | | | | | |
| | High-Severity Defects Delivered | 260 | 24 | 3 | High complexity raises defect severity levels |
| | | | | | |
| | Security Flaws Delivered | 35 | 3 | 0 | High complexity raises security flaws delivered with software |
| | | | | | |
| | | | | | |
| | Average monthly cost | \$7,500 | \$7,500 | \$7,500 | |
| | | | | | |
| | OVERALL PROJECT | | | | |
| | Development Schedule (months) | 16.98 | 13.80 | 12.45 | High complexity stretches out schedules |
| | Staff (technical + management) | 10 | 7 | 6 | High complexity increases staffing |
| | Development Effort (staff months) | 170 | 99 | 75 | High complexity increases effort months |
| | Development Costs | \$1,273,683 | \$739,492 | \$560,032 | High complexity increases costs |
| | | | | | |
| | DEVELOPMENT ACTIVITES | | | | |
| | Requirements Effort (staff months) | 16.00 | 8.00 | 7.00 | |
| | Design effort (staff months) | 22.00 | 15.00 | 10.00 | |
| | Coding effort (staff months) | 45.00 | 24.00 | 20.00 | |
| | Testing effort (staff months) | 50.00 | 28.00 | 23.00 | |

| | | | | | |
|--|--------------------------------------|--------------------|-----------------------|------------------|--|
| | Documentation effort (staff month) | 15.00 | 6.00 | 6.00 | |
| | Management effort (staff months) | 22.00 | 9.00 | 9.00 | |
| | TOTAL EFFORT (Staff months) | 170.00 | 90.00 | 75.00 | |
| | | | | | |
| | Function points per month | 5.88 | 11.11 | 13.33 | |
| | Work hours per FP | 22.44 | 11.88 | 9.90 | |
| | LOC per month | 342.25 | 592.59 | 656.41 | |
| | | | | | |
| | Total Cost of Development | \$1,275,000 | \$675,000 | \$562,500 | |
| | Total Cost of Maintenance | \$1,650,000 | \$625,000 | \$110,000 | High complexity raises maintenance costs |
| | Total Cost of Enhancement | \$575,000 | \$300,000 | \$200,000 | High complexity raises enhancement costs |
| | TOTAL COST OF OWNERSHIP (TCO) | \$3,500,000 | \$1,600,000 | \$872,500 | High complexity raises TCO |
| | | | | | |
| | TCO per Function Point | \$3,500.00 | \$1,600.00 | \$872.50 | High complexity raises TCO \$ per function point |
| | TCO per K Lines of Code | \$60.16 | \$30.00 | \$17.72 | High complexity raises TCO \$ per KLOC |
| | | | | | |
| | | | | | |
| | | | END OF EXAMPLE | | |