Software Development Rules of Thumb

Peter Kulik
Kulik & Lazarus Consulting, Inc.
February, 1996

Many software development rules of thumb are easy to use and highly relevant. Application of rules of thumb in planning can significantly improve project estimation and therefore reduce project risk.

This paper summarizes rules of thumb that have proven highly useful to address the challenges of managing software development.

Rules of thumb for software development can be applied effectively to most software development projects. Applying rules of thumb is usually quite straightforward, and has benefits including:

- Improved project estimation
- Reduced project risk.
- Industry context for project planning
- Foundation for data-based decision-making

In combination with assumptions about staffing and costs, rules of thumb provide an easy and effective way for project teams to estimate key project parameters such as overall duration, duration of the test phase, expected errors to be discovered, etc.

Project risk can be reduced most dramatically during planning – before project execution has begun. Rules of thumb treat software development from a "top-down" perspective, providing a context consistent with industry experience for project planning. An individual organization can decide if they are better or worse than industry experience based on internal factors; usage of rules of thumb ensures this is a conscious decision.

Rules of thumb also enable project planning teams to make decisions based on data. For project planning teams with a strong experiential foundation, rules of thumb are an extremely valuable supplement to this experience. Where the data and experience align, the team's decisions are reinforced. Where data and experience are different, rules of thumb catalyze important examination of underlying factors and root

Copyright © 1996, **Kulik & Lazarus Consulting, Inc.**
http://www.klci.com
513-291-1851

Software Development Rules of Thumb

Peter Kulik, February 1996

1.	Project percentage by phase: [2] Specification 18% Design 19% Coding 34% Test 29%
2.	Projects created primarily from reused software take about 1/4 the time and resources of new software. [2]
3.	"50% to 75% of all design errors can be found with inspections." [2]
4.	"An average project delivers about 350 NCSS per engineering month (engineering time includes all design, implementation, and test to produce all deliverable NCSS for a project)." [2]
5.	Modules with a cyclomatic complexity greater than 10 are more difficult to understand and have higher defect density than modules with smaller values. [2]
6.	"Typical testing without measuring code coverage only exercises around 50% of the code. With code coverage instrumentation, this can be raised to at least 80% without excessive additional effort." [2]
7.	"Projects created primarily from reused software experience only about 1/3 the defect density (defects/size) of those that are new." [2]
8.	"You will find about one defect post-release for every ten defects that you find prerelease during test." [2]
9.	"The time to fix defects in large mature software systems is 4-10 times the time to make fixes before, or shortly after, initial release of a systems." [2]
10.	Defect correction rates:[3] 25% of defects are found and fixed in 2 hours/defect 50% of defects are found and fixed in 5 hours/defect 20% of defects are found and fixed in 10 hours/defect 4% of defects are found and fixed in 20 hours/defect 1% of defects are found and fixed in 50 hours/defect
11.	"HP historical data shows that projects that don't use the certification process experience an average of around 7 defects/KNCSS before release of a product." [3] We have found an average of 7 to 10 defects/KNSS in a variety of organizations.
12.	Based on the Rayleigh distribution, "to reach the 99-percent level...takes about 1.25 the time it takes to reach the 95-percent level...; to reach the 99.9-per cent level...takes about 1.50 the time to reach the 95-percent level." [1]

Table 1: Relevant rules of thumb for project estimation.

causes. For project teams without a strong experience base, rules of thumb provide a framework within which to evaluate project planning assumptions and results.

While rules of thumb provide an important data point to all project planning teams, they are generally a single data point and will not substitute for good project planning techniques. The author has found rules of thumb to be most effective in combination with both quantitative models and software risk management tools and techniques. This portfolio of tools provides potent solutions for planning and optimization of project execution.

Sources of Rules of Thumb

A number of software development organizations track internal software development productivity and key process factors. The resulting database, built up over at least two to three years and many projects, enables companies to:

- Develop internal rules of thumb
- Measure and track improvement over time

Any software development organization can capture metrics from which rules of thumb can be derived. All that is required is consistent measurement of key software development parameters over time and a number of projects.

Since rules of thumb are based on individual characteristics of the organization which creates them, they will be slightly different from organization to organization. For example, programmer productivity has been found to vary greatly between individuals. An organization's programming productivity is a function of the productivity of the individuals within the organization. One organization that is lucky enough to have several highly productive programmers can and should use a different productivity rule of thumb than another organization with less productive individuals.

This organization-specific behavior of rules of thumb needs to considered by project teams in organizations substantially different than the rule of thumb's source. For example, many of the rules of thumb in Table 1 are derived from Hewlett-Packard. Organizations substantially different from Hewlett-Packard need to consider the applicability of these rules of thumb. In all but the most extreme cases, the author's experience has been that these rules of thumb are

Copyright © 1996, **Kulik & Lazarus Consulting, Inc.**
http://www.klci.com
513-291-1851

extremely useful – but generally in combination with other methods.

Note that the rules of thumb in Table 1 are based on a Lines-of-Code sizing methodology. Rules of thumb for other sizing methodologies (such as Function Points) do exist, but are beyond the scope of this paper.

Examples

Top-down project planning requires estimation of several key project parameters:

- Overall project duration
- Project development staffing
- Resources required for testing

Overall project duration can be estimated by first estimating the size of a project in terms of lines of code (non-comment source statements (NCSS)), then applying rule of thumb #4 from Table 1 (programmer productivity). Reuse can be factored in by calculating a value for "effective NCSS" in the project using rule of thumb #2 (savings from code reuse). The impact of inspections and code coverage measurements can be considered using rules of thumb #3 (design inspections) and #6 (code coverage tools). The duration of project phases can be estimated by applying rule of thumb #1 (project percentage by phase) to the overall project duration.

Project duration is a function of both productivity and staffing. Project staffing is estimated when applying Rule of Thumb #4 (programmer productivity) to estimate overall project duration, as described in the preceding paragraph.

Required testing resources is a function of the test phase duration and the number of defects expected to be discovered. The latter can be estimated using rule of thumb #11 (expected defect density); in addition, rule of thumb #7 (reused code defect density) can be invoked to reflect the impact of code reuse, and #6 (code coverage tools) to reflect the impact of code coverage tools.

Based on the number of defects expected to be discovered, rule of thumb #10 (defect correction rates) can be used to describe test duration as a function of resources – people and equipment. The number of test systems for parallel testing, number of testers, and – most importantly – the number of developers required to support bug fixing can be planned effectively to maximize the fix rate and optimize the duration of testing.

Conclusion

Rules of thumb are relevant, easy-to-apply, and effective tools for project planning. As a supplement to a project team's experiential foundation, they lay the groundwork for data-based decision-making. Since rules of thumb consider software development from a "top-down" perspective, they do not enable evaluation of individual risk factors for the specific project being managed. In the author's experience, a powerful combination of rules of thumb, quantitative models, and software risk management tools and techniques has proven most effective.

References

1. Putnam, Lawrence H., and Myers, Ware, Measures for Excellence, Yourdon Press, Englewood-Cliffs NJ, 1992.
2. Grady, Robert B., "Practical Rules of Thumb for Software Managers", *Hewlett-Packard Software Engineering Productivity Conference Proceedings*, August 1990, pages 647-651.
3. Grady, Robert B., "The Role of Soft ware Metrics in Managing Quality and Testing", January 30, 1990.

***Peter Kulik** is Managing Partner of **Kulik & Lazarus Consulting, Inc.** With more than 10 years experience in all aspects of software development, he holds an MS in Engineering Management with the thesis "Practical Quantitative Methods for Software Development Process Management", a Certificate in Economics and Finance, and a BS in Electrical Engineering. He can be reached via e-mail at pkulik@klci.com.*

***Kulik & Lazarus Consulting, Inc.** focuses on enabling software development organizations to accelerate completion of their projects. Leveraging more than 25 years practical experience, we use innovative tools to apply leading-edge critical path and risk management methodologies in an action-oriented framework. Our services enable clients to identify, quantify, and proactively address opportunities to improve their project completion dates on projects of five to fifty people. Contact us at 513-291-1851, or via the World Wide Web at http://www.klci.com.*

Copyright © 1996, **Kulik & Lazarus Consulting, Inc.**
http://www.klci.com
513-291-1851