Early Estimation of Software Reliability in Large Telecom Systems

Alex Lubashevsky
Independent Consultant

This article describes early estimation experiences with software reliability of complex real-time telecommunications systems based on size estimation and the process assessment. It emphasizes the importance of estimation for evaluating the feasibility of proposed reliability requirements and providing a rational basis for design and allocation decisions. A number of critical factors for building a reliability estimation model are discussed along with typical sizing processes and estimation tools. The modified U.S. Air Force's Rome Laboratory model was selected as the best practical candidate for the early estimation of software reliability. A data comparison of its results to a traditional estimation model is presented.

Customers rank reliability first on their list of most critical quality attributes, according to a recent survey of the largest customers of complex real-time telecommunications systems and data published in the Army's software metrics newsletter, *Insight* (Spring 2000) [1]. With the cost of some systems exceeding tens or even hundreds of millions of dollars and with a development duration of more than 12 to 18 months, early reliability estimation can significantly contribute to the success (or early rational cancellation) of the project. With recent strong emphasis on speed of development, the decisions made on the basis of early reliability estimation can have the greatest impact on schedules and cost of software projects. Software reliability may also be significantly improved in an early stage by a focused review and inspection process, early defect removal, and thorough test effort.

Software reliability estimation provides a solid foundation to perform meaningful tradeoff studies at project start. It also provides a projection of the software failure rate before systems tests begin or at any point throughout. After estimation, the next logical step is creating a software reliability growth model, which covers the period where reliability improves as the result of thorough peer reviews, testing, and fault correction. Reliability metrics help to predict such critical factors as the initial failure rate, final failure rate, fault density, fault profile, etc.

The final outcomes of software reliability estimation include the following:

- An estimation of the number of faults expected during each phase of the life cycle.
- A constant failure rate estimation at system release.
- Relative measures for practical use and management such as duration of system test and size of the test team.

If software reliability estimation is performed early in the software life cycle, it is possible to determine what improvement, if any, can be made to the software methods, techniques, or organizational structure. As described in this article, our experience also confirmed what many recent articles and publications have suggested: A successful, meaningful estimating strategy must simultaneously use more than one estimating technique [2].

"Software reliability estimation provides a solid foundation to perform meaningful tradeoff studies at project start."

For early software reliability estimations, an estimation team used a number of software reliability models. The team compared the results of these models with each other and with the reliability data provided by cost/effort estimation model KnowledgePLAN, which was selected a few years ago after Bell Labs' studies of more than a dozen different estimation models (the COCOMO tool came out a very close second). This expert system tool contains in its database more than 8,000 actual projects that along with size, cost, effort, and other attributes have data about the total number of inherent faults, their distribution among major phases of life cycle, the severity of faults, and potential defect removal efficiency [3].

As with any other existing cost estimation tool, KnowledgePLAN had to be fine-tuned and thoroughly calibrated for the particular project to be consistent and reliable. At the same time in this environment, the KnowledgePLAN questionnaires on the project/process development activities were used as a *shell* for the company-wide software process assessments, together with some parts of the Software Engineering Institute's Capability Maturity Model® (CMM®). These assessments were used for internal benchmarking of current development practices and helped to easily identify similar projects, which was essential for a meaningful comparison [4].

Estimating Parameters

AT&T/Lucent's experience in software reliability estimation for a number of large telecommunications projects is evidence that the size of the project (in function points or sometimes in thousands of lines of code [KLOC]) is the most significant single factor for estimating the number of inherent faults [4]. The second most important factor is project complexity, which can be represented by McCabe's cyclomatic complexity measure (problem, code, and data complexity). Those complexity measures strongly depend on the application types of the project [3, 5].

The KnowledgePLAN tool uses simplified McCabe's complexity factors in its questionnaire, as does its predecessor Checkpoint. Also our previous independent studies found that the fault-reduction factor is relatively stable across different projects in the same organization, though the fault-exposure ratio may be dependent on the structure of the program and the degree to which faults are data dependent. However, these effects are often averaged out for programs of substantial size such as the large projects the estimation team often deals with [6].

To increase estimation accuracy and our ability to better control the discovery-of-faults process, the team concluded the following: The software reliability estimation must be performed in early phases of the life cycle by using phase-based models that emphasize the availability of size and corresponding effort for the project during early phases. In our case, the estimation team obtained the size and effort in the early stage of development very often by using the function point method [7]. However, for some legacy projects, the data were derived by the analogy method or by experts' iterative estimations such as Delphi analysis [5].

Also, the number of factors needed for building reliability estimation, which significantly relates to fault density at the early stages such as application type, development environment, and some other software metrics, can be obtained relatively easily from the results of previously performed company-wide software process assessments. After a number of studies and experiments with different types of reliability models, the team selected the U.S. Air Force's Rome Laboratory model [8]. The estimation team chose this model because it allows us to track the influence on software reliability of the various application types by different development organizations, methods, tools, techniques, and other software factors, and it is closely correlated to our development methodology.

Estimating Size and Reliability Early

Size estimation consists of two phases, called *passes*, which require using highly trained and experienced estimators. After a list of new or modified features is prioritized during the proposal stage, the first pass provides a quick rough estimate of size/effort required to develop the features. These are then given to product management to determine the budget for an upcoming release.

The estimate is determined by breaking up the software into smaller pieces, consulting with experts, and forming analogies to previously developed software with similar features. If a separate estimation repository database updated on a quarterly basis contains the data for a similar feature, the preliminary data for the size/effort estimation and often the total number of inherent faults are readily available. If data from the repository are not available, one of the reliability estimation models with some of the default parameters is used. One or two members of the planning group usually generate the first-pass estimate with accuracy within 50 percent.

The second pass is a detailed estimate done near the end of the requirements process to define size/effort of a feature, a subsystem, or system level. The development planning group usually coordinates the entire process. For some projects, the second-pass estimates are based on the judgment of experienced developers (rather than expert estimators) who are using a bottom-up estimation technique based on historical data to outline and estimate the tasks by functional area that will be required to develop a feature.

This group meeting is used to obtain a single estimate from a group of experts after an open discussion. The meeting is a forum for experts to discuss requirements and designs, consider a tradeoff between the reliability of the product and cost and schedules of the project, resolve issues, and work together to create and tune their estimates. The Delphi technique [5] can be combined with the group meeting approach. The group meeting is used to discuss estimation issues and the experts give their opinions. These estimates are discussed again, and the process is repeated until a consensus is reached.

"Size estimation consists of two phases, called passes, which require using highly trained and experienced estimators."

Size/effort estimation and early reliability estimation by analogy was the most popular method. The technique assumes that if two projects are alike in some respects, other similarities can be inferred. The current project is compared with similar completed projects to get a ballpark estimate; experts then factor in the differences between the projects. Estimation by analogy can be applied to a total system, a module, or a task. A historical database is the best tool for estimation by analogy. Here are sources for analog data in the order of decreasing preferences:

- Data from a previous release of the same project.
- Data from a similar project in the same company.
- Data from a similar project in a different company.

Industry data can be used if company data are not available, but they must be calibrated to company data as soon as they are available.

For a completely new project that may have no relevant data or experience on which to base estimates, particularly if the project is moving to a new methodology such as from a traditional to an object-oriented approach, the following estimation strategy is recommended: Create, if possible, an analog by dividing the product into components and actually doing development on a typical component to estimate the remainder of the project. Also, the following additional information associated with the software cost/effort estimation and reliability estimation is recommended:

- Project management tracks baseline, current and completion dates, and the number of detected faults.
- At the end of each software release, postmortem compares actual vs. detailed estimates for the features.
- The estimation process is evaluated on two criteria: responsiveness (in business days) and accuracy, which is defined as the percent of relative error between the detailed estimate and actual data.

The new estimation process described in the next section also consists of two phases (sometimes an additional zero phase is added) and is based on more consistent usage of estimation tools and techniques and less reliance on high human expertise. But this estimation practice strongly concurs with the suggestion that the most important factor in improving estimation is to "hold its software estimators, developers, and managers accountable for their estimates" [9].

A New Estimation Process

Our new estimation process is based on size estimation using function point analysis (FPA), which is usually performed based on complete requirements [7] in combination with estimation tools like KnowledgePLAN and the modified Rome Labs estimation model. (In some cases, FPA is based on particular telecom domains or, even earlier, based on the high-level Feature Definition and Assessment Form (FDAF), the most important estimation phase called zero pass.)

The project size in function points (FP) is one of the significant inputs for the KnowledgePLAN tool. Other input information collected by the KnowledgePLAN's questionnaire describes the type, nature, and complexity of the project, the project management practices, the expertise and morale of the team, together with a few dozen other attributes of the particular project. This enables the tool to choose the project having the closest match from its vast knowledge base of previously collected industry projects.

As output, the tool generates many

Factor	Measure	Range of Values: Rome Labs defs/KLOC	Range of Values: Telecom defs/FPs	Application Phase*	Trade-off Range
A-Application	Difficulty in developing various application types	2 to 14	0.2 to 1.5	AP-T	None fixed
D-Development environment	Development org., methods, tools, techniques, document	.5 to 2.0	0.1 to 1.8	If known at AP, DTLD-T	Largest range
SA-Software anomaly mgmt.	Indication of fault-tolerant design	.9 to 1.1	0.3 to 0.4	Normally, C-T	Small
ST-Software traceability	Traceability of design and code to requests	.9 to 1.0	0.2 to 0. 6	Normally, C-T	Large
SQ-Software quality	Adherence to coding standards	1.0 to 1.1	0.2 to 0.3	Normally, C-T	Small
SL-Software language	Normalizes fault density by language type	N/A	N/A	C-T	N/A
SX -Software complexity	Unit complexity	.8 to 1.5	0.1 to 0.6	C-T	Large
SM-Software modularity	Unit size	.9 to 2.0	0.1 to 0.7	C-T	Large
SR-Software standards review	Compliance with design rules	.75 to 1.5	0.2 to 0.4	C-T	Large

*AP = Concept or Analysis Phase, C = Coding, DTLD = Detailed and Top Level Design, and T = Testing

Table 1: *Summary of the Rome Laboratory Model, RL-TR-92-52*

useful estimation reports on resources, schedule, etc., and also on the total number of defects that will be introduced during various stages of the project. While in the past the estimation team widely used the predicted reliability data on potential defects from the Checkpoint estimation tool (KnowledgePLAN's predecessor), the team decided to compare the estimation tool's results with those from another software reliability model.

For this purpose, the original U.S. Air Force's Rome Laboratory model, RL-TR-92-52 [8] was modified to allow for more than 60 telecommunications applications to be included in the historical database, and also to allow for the usage of the FP method, which is growing in popularity in civil and military applications [4]. Using available industry and internal data from the software process assessment (SPA) and the CMM of the organization developing the software, the major nine factors of the Rome Lab's model were expanded to include new ranges of values in FPs. (See Table 1).

Originally, the output of the Rome Lab's model is a fault density in terms of faults per KLOC. To compute the total estimated number of inherent defects, the fault density should be multiplied by the total predicted number of KLOC. If function points are being used and no KLOC is available for correlation, the backfire method (low accuracy table for the conversion of source lines of code to FPs [3, 4]) is sometimes recommended. Also the Rome Lab's model is very useful for predicting fault density at delivery time; subsequently, this fault density is utilized to predict the total number of inherent faults and the failure rate.

The fault density of the application (A) is predicted by using a baseline fault density established for applications of the same type, adjusted to reflect the influence of the development environment (D) and the software characteristics (S) of the specific application. Once fault density is determined, the failure rate (FR) can be predicted by applying empirical value (EV), established from historical data for the application type, to the fault density. The Rome Lab's model contains empirical data that have a total of 33 data sources representing 59 different projects (some from Software Engineering Laboratory).

Fault Density: $FD = A \times D \times S$
(faults/FPs or LOC)

Estimated number of Inherent Faults:
$N = FD \times SIZE$

Failure Rate: $FR = FD \times EV$
(faults/time)

This model has the following significant benefits:

- It can be used as soon as the software concept is known.
- During the concept phase, it allows *what-if* analysis to be performed to determine the impact of the development environment on fault density. (In our case, the data from the previous SPA for this organization will be reused.)
- During the concept phase, it allows *what-if* analysis to be performed to determine the impact of software characteristics on fault density. (Also the data from the previous SPA for this organization will be reused.)
- It allows for system software reliability allocation because it can be applied uniquely to each application type comprising a large software system.
- The estimation can be easily customized using unique values for the A, D, and SIZE factors based upon historical software data from the specific organization's environment.

The Rome Lab's model consists of nine factors (Table 1, first column) that are used to predict the fault density of the software application. That is, application type factor (which could be real-time control systems, scientific, or information management) with the range of values of two to 14 defects per KLOC, or 0.2 to 1.5 defects per FP. This demonstrates the potential influence of different application types on fault density in an early phase of development. Similar logic applies to the rest of the factors: They show the potential ranges of values for fault density that depend on the type of factors and measures associated with them.

There are parameters in this estimation model that have tradeoff capability (maximum/minimum predicted values). The analyst can determine where some changes can be made in the software engineering process or product to achieve improved fault-density estimation. This tradeoff is valuable only if the analyst has the capability to impact the software development process. (Notice that the tradeoff is fixed for the type of application and is not applicable after you select a particular software language.) The tradeoff analysis can also be used to perform a cost analysis by optimizing the development.

The values of many of the parameters in this model may change as development proceeds. The latest updated values should be used when making an estimation that will become more and more accurate with each successive phase until the final design and implementation.

Table 1 represents the summary of Rome Laboratory's estimation model. The column "Range of Values" shows original and modified telecom values, the latter reflecting the historical reliability data correlated with the data from the SPA assessments range for more than 60 projects. Most of the factors of the original model like software implementation metrics (SX, SM, SR, etc.), requirements and design representation metrics (SA, ST, SQ), and application (A) and development environment (D) correspond almost one-to-one to the factors of the SPA/SPR questionnaire. The only difference is that range of values for fault density is mapped in defects per FPs instead

Severity Level	Predicted	Industry Standard	Actual	Percent of Variance (Actual to Predicted/Industry Standard)
1. System inoperative	87	103	67	22/35
2. Major functions incorrect	408	516	359	14/30
3. Minor functions incorrect	1185	1446	1076	10/25
4. Superficial error	945	1034	742	27/28
Total	2626	3098	2244	17/28

Table 2: *Faults by Severity Level*

of defects per KLOC.

Table 2 provides an example of data for one of the critical telecom projects. The predicted and industry standard data were generated using the project quality report of KnowledgePLAN. Actual data came from the final systems/integration test reports. The data from the two different models were compared and correlated with the SPA data for this project and the reliability data produced by the modified Rome Labs model. The percent of variance between actual and predicted faults showed an acceptable range of deviation (from 10 percent to 27 percent, as compared with the acceptable range for this most important early stage of estimation: +/- 50 percent).

Table 2 represents data for a project size of about 1,800 FPs. The fault density (1.24 faults/FP) and total number of inherent faults (2,626) were predicted for the FDAF stage of the life cycle. These data, together with some historical information for similar projects, helped predict the duration of systems test with about 15 percent accuracy and the size of the test team with about 20 percent accuracy. The early predicted faults in Table 2 are presented by severity level (based on historical distribution data for similar projects) and percent of deviation from actual and industry database. The 17 percent deviation of total predicted faults to actual is a significant (almost three times) improvement compared with the first-pass results (50 percent) at the same stage.

Also during this object-oriented project for a real-time telecommunication system, the early reliability estimation data indicated a need for a focused review and inspection process especially during analysis, design, and additional systems test effort to fully cover all test cases written against original user requirements. This helped to increase defect removal efficiency to 92 percent (which is very high for this type of transmission application) and to create a system with software availability exceeding Telcordia standards by more than 10 times. This particular system was in the field for more than one year without any major software outage.

Summary

This article has described a number of experiences of early estimation of software reliability for large real-time telecommunications systems. The benefits of early reliability estimation for the design and allocation decisions, which can have a significant impact on schedule and cost, were also discussed. The article also emphasized the importance of early size and complexity estimation on which a number of software reliability models are based. The past processes of early size and reliability estimation (relying on highly qualified human experts) and new processes (based on the heavy usage of estimation tools) were described in detail. The modified Rome Labs estimation model, based on early size estimation as well as a number of other factors that describe the software development process and its influence on reliability, was introduced for comparison and was shown to be very useful. The example of working with two different models for early reliability estimation and the positive results achieved proved that other projects could significantly benefit from the above-described processes in building reliable systems.

References

1. Insight 4:1. Spring 2000. (Insight is the Army's Software Metrics Newsletter. Available at: <www.ArmySoftware Metrics.org>.)
2. Shepperd, M. J., and C. Schofield. "Estimating Software Project Effort Using Analogies." IEEE Trans. Software Engineering 23.12 (1997).
3. Jones, T.C. Applied Software Measurement. McGraw-Hill, 1991.
4. Lubashevsky, Alex, and L. Bernstein. "Living with Function Points at AT&T." CrossTalk Nov./Dec. 1995.
5. Boehm, B. W. Software Engineering Economics. Prentice Hall, 1981.
6. Musa, J. D. Software Reliability. Measurement, Estimation, Application. McGraw-Hill, 1987.
7. Albrecht, A. J., and J. R. Gaffney. "Software Function, Source Lines of Code, and Development Effort Estimation: A Software Science Validation." IEEE Trans. Software Engineering 9.6 (1983).
8. Rome Laboratory. "Methodology for Software Reliability Estimation and Assessment." Technical Report RL-TR-92-52, Vol. 1 and 2, 1992.
9. Lederer A. L., and J. Prasad. "A Causal Model for Software Cost Estimating Error." IEEE Trans. Software Engineering 24.2 (1998).

About the Author

Alex Lubashevsky is an independent consultant specializing in estimation and reliability. He was previously an estimation project manager with AT&T/Lucent for 17 years. He was responsible for estimating size, effort, interval, and defects for more than 200 telecom and data processing systems inside and outside the company (IRS, DELTA, Prudential, etc). While at AT&T, he also helped to achieve one of the first industry Capability Maturity Model® Level 3 ratings. He originated and helped to develop the first automated CASE tool for early estimation (Bachman Analyzer). Lubashevsky was an early industry supporter of Practical Software and Systems Measurement and a pioneer in the National Software Council and later a board member. He has a master's of science degree in computer science from New York University and a bachelor's of science degree in computer science from the Polytechnic University of Kharkov, Ukraine. He is a member of the International Electrical and Electronics Engineers Computer and Communications Societies, the International Function Point Users Group, and the International Academy of the Information Sciences.

165 Osprey
Hackettstown, NJ 07840
Phone: (908) 813-0208
Fax: (908) 813-0208
E-mail: alexluba1@att.net