Reliability Growth Models for Defect Prediction

Norman Schneidewind
Naval Postgraduate School, USA

INTRODUCTION

In order to continue to make progress in software measurement as it pertains to reliability, there must be a shift in emphasis from design and code metrics to metrics that characterize the risk of making requirements changes. By doing so, the quality of delivered software can be improved because defects related to problems in requirements specifications will be identified early in the life cycle. An approach is described for identifying requirements change risk factors as predictors of reliability problems. This approach can be generalized to other applications with numerical results that would vary according to application.

BACKGROUND

Several projects have demonstrated the validity and applicability of applying metrics to identify fault prone software at the code level (Khoshgoftaar & Allen, 1998; Khoshgoftaar, Allen, Halstead, & Trio, 1996a; Khoshgoftaar, Allen, Kalaichelvan, & Goel, 1996b; Schneidewind, 2000). This approach is applied at the requirements level to allow for early detection of reliability and maintainability problems. Once high-risk areas of the software have been identified, they would be subject to detailed tracking throughout the development and maintenance process (Schneidewind, 1999).

Much of the research and literature in software metrics concerns the measurement of code characteristics (Nikora, Schneidewind, & Munson, 1998). This is satisfactory for evaluating product quality and process effectiveness once the code is written. However, if organizations use measurement plans that are limited to measuring code, the plans will be deficient in the following ways: incomplete, lack coverage (e.g., no requirements analysis and design), and start too late in the process. For a measurement plan to be effective, it must start with requirements and continue through to operation and maintenance. Since requirements characteristics directly affect code characteristics and hence reliability, it is important to assess their impact on reliability when requirements are specified. As will be shown, it is feasible to quantify the risks to reliability of requirements changes—either new requirements or changes to existing requirements.

Once requirements attribute that portend high risk for the operational reliability of the software are identified, it is possible to suggest changes in the development process of the organization. To illustrate, a possible recommendation is that any requirements change to mission critical software—either new requirements or changes to existing requirements--would be subjected to a *quantitative* risk analysis. In addition to stating that a risk analysis would be performed, the policy would specify the risk factors to be analyzed (e.g., number of modifications of a requirement or *mod level*) and their threshold or critical values. The validity and applicability of identifying critical values of metrics to identify fault prone software at the code level have been demonstrated (Schneidewind, 2000). For example, on the space shuttle, rigorous inspections of requirements, design documentation, and code have contributed more to achieving high reliability than any other process factor. The objective of these policy changes is to prevent the propagation of high-risk requirements through the various phases of software development and maintenance. The payoff to the organization would be to reduce the risk of mission critical software *not* meeting its reliability goals during operation.

Definitions

- **D_{ir}:** *Cumulative* defects (e.g., discrepancy reports: program executes incorrect path due to excessive requirements conflicts), corresponding to risk variable r during operating time interval i.
- **r_i:** Requirements risk (e.g., risk of excessive size, excessive conflicting issues)
- **rmin:** Minimum value of r_i
- **rmax:** maximum value of r_i

Risk Variables:

- **Sloc:** *Cumulative* source lines of code (sloc): number of source lines of code written for a Shuttle release.
- **Issues:** *Cumulative* number of possible conflicts among requirements (e.g., an aggregation of requirements to provide greater Shuttle thrust conflicts with the fact that more thrust requires more engine weight). An aggregation of conflicting requirements issues causes reliability risk.
- **t:** Operating time (e.g., execution, wall clock, calendar time)

- **t_c:** Critical value of t = maximum value of t
- **i:** Operating time interval
- **a and b:** Coefficients of D_{ir} obtained through regression analysis
- **Decision maker:** Software quality control manager

Research Ideas on Models

Models are representations of states, objects, and events. They are less complicated than reality, hence easier to use. This is due to the fact that only relevant properties are included in the model, as explained by Ackoff, Gupta, & Minas, 1962).

Models of Problem Situations (Ackoff et al., 1962)

$V = f(X_i, Y_j)$, where

V = measure of the value of the decision that is made (i.e., action that is taken)

Example: Probability that software will survive for an operational time > t (i.e., reliability R (t))

X_i = the variables that are subject to control by the decision. The decision variables define alternative courses of action.

Example: amount of time T allocated to testing software

Discrete or continuous decision variables

Example: $D_{ri} = a\ r_i^b$ has the continuous variable defect count D_{ri}

Y_j: the factors (variable or constant) that affect performance, which may or may not, be subject to control by the decision maker. These are called parameters.

Example: Failure rate parameters α and β in reliability model

f = functional relationship between independent variables and parameters X_i and Y_j and dependent variable V.

Example: $D_{ri} = a\ r_i^b$

A model has two essential characteristics: At least one of the decision variables X_i is subject to control by the decision maker.

Example: Amount of time T allocated to testing software Second, the value V must be a measure of alternative courses of action.

Example: Probability that software will survive for an operational time > t (i.e., reliability R (t)). The decision maker sets the value of t, and, hence, the value of R (t).

Models with the above properties are called decision models.

Some models contain constraints:

Example: R (t) > R* (t), where R* (t) is the minimum allowable reliability.

Models as approximations of the real world

Therefore, we start with a small and simple defect prediction model and build upon this to achieve more complex and accurate models.

Churchman, Ackoff, and Arnoff, (1957) offers the following advice concerning how to view models:

Viewed generically, a model is a representation of some subject of inquiry such as a process, like the software defect reduction process. The model is used for prediction (e.g., predict defect count as a function of risk variables) and control (i.e., control software quality by managing defect occurrence). The primary purpose is explanatory rather than descriptive. For example, we could want to show how defect count varies as a function of risk variables and operating time interval. A great advantage of such models is the ability to manipulate the model without having the change the system that is being modeled. Thus, we would not want to change the software quality control system just to experiment with how defects vary with requirements risk! Rather, we might change the system depending on the results of our model experiments.

Box, Hunter, and Hunter (1978) states the following with respect to experimental model building:

1. The experimenter must construct a flexible model, subject to change, in case the model assumptions turn out to be erroneous. For example, we may find that defect count does not vary exponentially with requirements risk variables. With this outcome, we want a model that could be changed easily, for example, to a linear model.
2. Frequently, the mechanism underlying the model is not completely understood, or is too complicated, to allow an exact model to be postulated from theory. In this case, a simplified model with a response over a limited range of the variables is appropriate.