### 10.7.1 Reuse and Quality, Overview

If a software system is composed, in part, of reused code, its quality will probably be greater than if it were composed entirely of new code. Here, higher quality means fewer defects remaining undiscovered when a software system is shipped to its users. Reuse enhances the quality of a software product principally because of the increased opportunity it provides for defect discovery. Each time reusable code is used in a new application software system, it passes through the integration and system test processes again. Thus, an additional opportunity is provided for defect discovery and removal. This section focuses on the effect on quality (defined in terms of defect content) due to code reuse and, implicitly, due to the reuse of the requirements and the design from which it came as well.

### 10.7.2 Model of Effect of Reuse on Software Quality

This section develops a model showing the effect of code reuse on software product quality. The model shows the increase in product quality (fewer defects) due to reuse (relative to the quality of the product if it were all new code). This model reflects the fact that multiple uses of the same code affords more opportunities for discovering errors or defects than if that element of code were employed for the first time in the software product.

Assume that the code to be reused in a new software system has gone through the complete development process (whether it is provided from a library or taken from a prior system). Then, this code is presumed to go through integration and system test again during the development of the new application system. However, it is assumed the code does not go through the defect discovery steps of design and code inspections and unit test again.

Let $D_{VR}$ be the latent error density (see Section 11.6) of some (reused) code to be incorporated into a new software system. Let $D_{VN}$ be the latent error density of the new code portion of the new software system. Let both $D_{VN}$ and $D_{VR}$ be measured in errors per KSLOC.

Then, the expression for $D_{Ri}$, the latent defect density in the new software product which includes the *i*th use of the "reused" code:

$$D_{Ri} = D_{VN} \cdot (1 - R) + D_{VR} \cdot R \cdot p^{i-1}$$

where R is the proportion of code reused (on the average over the N planned uses of the reused code). Let:

$$p = 1 - \frac{\text{Defects discovered and removed during the integration and system test process}}{\text{Total number of lifetime defects}}$$

Further, note that $D_{VR} = p \cdot D_{VN}$.

In the case of the first reuse after the creation of the reusable software, i = 1, and:

$$D_{Ri} = D_{R1} = D_{VN} \cdot (1 - R) + D_{VR} \cdot R$$

In the case of the second reuse:

$$D_{Ri} = D_{R2} = D_{VN} \cdot (1 - R) + D_{VR} \cdot R \cdot p^1$$

An example value of p can be derived from data in Gaffney (1984a) that is presented in Table 10-5.

| Phase/Activity | Percent of Lifetime Errors |
|---|---|
| High-Level (Preliminary) Design Inspections | 7.69 |
| Detailed Design Inspections | 19.70 |
| Code Inspections | 23.93 |
| Unit Test | 20.88 |
| Integration Test | 14.27 |
| System Test | 7.92 |
| Latent Error Content | 5.61 |
| Total | 100.00 |

*Table 10-5. Example Values of Error Discovery Percentages*

In this case:

$$p = 1 - \frac{14.27 + 7.92}{100} = 1 - 0.2219 = 0.7781$$

The thinking behind the factor $p^{i-1} \cdot D_{VR}$ in the expression for $D_{Ri}$ is as follows. After the reusable code has been developed for the library or for use in some prior application system, it still has some of the defects (such as 5.61 percent of the errors indicated in the example situation in Table 10-5) that were injected during the development process. Upon