Chapter 19
Technical Metrics for Software

A Comment
McCall’s quality factors were proposed in the early 1970s. They are as valid today as they were in that time. It’s likely that software built to conform to these factors will exhibit high quality well into the 21st century, even if there are dramatic changes in technology.

Formulation Principles
• The objectives of measurement should be established before data collection begins
• Each technical metric should be defined in an unambiguous manner
• Metrics should be derived based on a theory that is valid for the domain of application (e.g., metrics for design should draw upon basic design concepts and principles and attempt to provide an indication of the presence of an attribute that is deemed desirable)
• Metrics should be tailored to best accommodate specific products and processes [BAS84]

Collection and Analysis Principles
• Whenever possible, data collection and analysis should be automated
• Valid statistical techniques should be applied to establish relationship between internal product attributes and external quality characteristics
• Interpretative guidelines and recommendations should be established for each metric

Attributes
• simple and computable. It should be relatively easy to learn how to derive the metric, and its computation should not demand inordinate effort or time
• empirically and intuitively persuasive. The metric should satisfy the engineer’s intuitive notions about the product attribute under consideration
• consistent and objective. The metric should always yield results that are unambiguous.
Attributes

- **consistent in its use of units and dimensions.** The mathematical computation of the metric should use measures that do not lead to bizarre combinations of unit.

- **programming language independent.** Metrics should be based on the analysis model, the design model, or the structure of the program itself.

- **an effective mechanism for quality feedback.** That is, the metric should provide a software engineer with information that can lead to a higher quality end product.

Analysis Metrics

- **Function-based metrics:** use the function point as a normalizing factor or as a measure of the "size" of the specification.

- **Bang metric:** used to develop an indication of software "size" by measuring characteristics of the data, functional and behavioral models.

- **Specification metrics:** used as an indication of quality by measuring number of requirements by type.

Architectural Design Metrics

- **Architectural design metrics**
  - Structural complexity = g(fan-out)
  - Data complexity = f(input & output variables, fan-out)
  - System complexity = h(structural & data complexity)

- **HK metric:** architectural complexity as a function of fan-in and fan-out.

- **Morphology metrics:** a function of the number of modules and the number of interfaces between modules.

Component-Level Design Metrics

- **Cohesion metrics:** a function of data objects and the locus of their definition.

- **Coupling metrics:** a function of input and output parameters, global variables, and modules called.

- **Complexity metrics:** hundreds have been proposed (e.g., cyclomatic complexity).

Interface Design Metrics

- **Layout appropriateness:** a function of layout entities, the geographic position and the "cost" of making transitions among entities.

Code Metrics

- **Halstead's Software Science:** a comprehensive collection of metrics all predicated on the number (count and occurrence) of operators and operands within a component or program.