# Code metrics

## 1 Characteristics of a metric

- A metric not only needs to have a clear and relevant association with an attribute, its values need to change in accordance with our expectations when the attribute changes
- The problem here is that the attributes we tend to associate with describing software are often rather imprecise, and can mean different things in different contexts - so the rules for combining them might not always be clear cut

## 2 What things do we need to measure?

Some examples of what we want to know about include:

- Code structure
- Control flow through code
- Information flow (and relationships)
- ”object” properties
- Execution of code
- Steps in development
- Specifications
- Design properties
- Testing outcomes

## 3 Attributes and Measures for Code

Two key attributes when thinking about development and evolution are:

- Size
- Relationships between elements

### 3.1 Halstead’s Measures

Focused on counting lexical tokens in programs to extract measure like

- Number of unique operators
- Number of unique operands
- Total occur
- Total occurrences of operands
- Size of a program

### 3.2 Lines of code

Widely used as a size metric as it is easy to measure, however it has limitations:

- Do you count empty lines?
- Do you count comments
- Statements can be expanded onto more lines for readability

## 3.3 Cyclomatic Complexity

Defined as

$$V(G) = e - n + 2$$

or can also be computed as number of decision points+1

This means there is no penalty for writing more legible code

### 3.3.1 Compound conditions

- Compound conditions are a problem for this as different routes could require different numbers of decisions
- The original rule treated any compound condition as a single decision
- There has been a suggestion of treating each element in a compound condition as a separate decision to get an upper bound