be two sets of entities for which some attribute can be measured on a ratio scale. We must show that the statement "The mean of the X_is is greater than the mean of Y_js" is meaningful. To do so, let M and M' be two measures for the attribute in question. Then we want to show that the means preserve the relation. In mathematical terms, we must demonstrate that

$$\frac{1}{n}\sum_{i=1}^{n} M(x_i) > \frac{1}{m}\sum_{j=1}^{m} M(y_j) \quad \text{if and only if} \ \frac{1}{n}\sum_{i=1}^{n} M'(x_i) > \frac{1}{m}\sum_{j=1}^{m} M'(y_j)$$

The ratio scale gives us the extra information we need to show that the assertion is valid. Thanks to the relationship between acceptable transformations for a ratio scale, we know that $M = aM'$ for some $a > 0$. When we substitute aM' for M in the above equation, we get a statement that is clearly valid.

The same investigation can be done for any statistical technique, using scale and transformation properties to verify that a certain analysis is valid for a given scale type. Table 2.8 presents a summary of the meaningful statistics for different scale types. The entries are inclusive reading downwards. That is, every meaningful statistic of a nominal scale type is also meaningful for an ordinal scale type; every meaningful statistic of an ordinal scale type is also meaningful for an interval scale type, etc. We will return to the appropriateness of analysis when we discuss experimental design and analysis in Chapter 4, and again when we investigate the analysis of software measurement data in Chapter 6.

2.4.2 Objective and Subjective Measures

When measuring attributes of entities, we strive to keep our measurements objective. By doing so, we make sure that different people produce the same measurement results, regardless of whether they are measuring product, process, or resource. This consistency of measurement is very important. Subjective measures depend on the environment in which they are made. The measures can vary with the person measuring, and they reflect the judgment of the measurer. What one judge considers bad, another may consider good, and it may be difficult to reach consensus on attributes such as process, product, or resource quality.

Nevertheless, it is important to recognize that subjective measurements can be useful, as long as we understand their imprecision. For example,

TABLE 2.8 Summary of Measurement Scales and Statistics Relevant to Each

Scale Type	Defining Relations	Examples of Appropriate Statistics	Appropriate Statistical Tests
Nominal	Equivalence	Mode	Nonparametric
		Frequency	
Ordinal	Equivalence	Median	Nonparametric
	Greater than	Percentile	
		Spearman r_S	
		Kendall τ	
		Kendall W	
Interval	Equivalence	Mean	Non-parametric
	Greater than	Standard deviation	
	Known ratio of any intervals	Pearson product-moment correlation	
		Multiple product-moment correlation	
Ratio	Equivalence	Geometric mean	Nonparametric and parametric
	Greater than	Coefficient of variation	
	Known ratio of any intervals		
	Known ratio of any two scale values		

Source: Siegel S. and Castellan N.J. Jr., *Nonparametrics Statistics for the Behavioral Sciences*, 2nd Edition. McGraw-Hill, New York, 1988.

suppose we want to measure the quality of requirements before we turn the specification over to the test team, who will then define test plans from them. Any of the techniques shown in Figure 2.2 would be acceptable. For example, we may ask the test team to read and rate each requirement on a scale from 1 to 5, where '1' means "I understand this requirement completely and can write a complete test script to determine if this requirement is met," to '5:' "I do not understand this requirement and cannot begin to write a test script." Suppose the results of this assessment look like the chart in Table 2.9.

Even though the measurement is subjective, the measures show us that we may have problems with our interface requirements; perhaps the interface requirements should be reviewed and rewritten before proceeding to test plan generation or even to design. It is the general picture that is important, rather than the exactness of the individual measure, so the subjectivity, although a drawback, does not prevent us from gathering useful information about the entity. We will see other examples throughout

TABLE 2.9 Results of Requirements Assessment

Requirement Type	1 (good)	2	3	4	5 (bad)
Performance requirements	12	7	2	1	0
Database requirements	16	12	2	0	0
Interface requirements	3	4	6	7	1
Other requirements	14	10	1	0	0

this book where measurement is far from ideal but still paints a useful picture of what is going on in the project.

2.4.3 Measurement in Extended Number Systems

In many situations we cannot measure an attribute directly. Instead, we must derive a measure in terms of the more easily understood *sub-attributes*. For example, suppose that we wish to assess the *quality* of the different types of transport available for traveling from our home to another city. We may not know how to measure quality directly, but we know that quality involves at least two significant sub-attributes, journey time and cost per mile. Hence, we accumulate data in Table 2.10 to describe these attributes.

Intuitively, given two transport types, A and B, we would rank A superior to B (i.e., A is of higher quality than B) if

$$\text{journey time } (A) < \text{journey time } (B) \textbf{ AND } \text{cost per mile } (A) < \text{cost per mile } (B)$$

Using this rule with the data collected for each journey type, we can depict the relationships among the candidates as shown in Figure 2.13. In the figure, an arrow from transport type B to transport type A indicates the superiority of A to B. Thus, Car is superior to both Train and Plane because, in each case, the journey time is shorter *and* the cost per mile is less. Figure 2.13 also shows us that Car, Train, and Plane are all superior to Coach.

Notice that in this relation Train and Plane are incomparable; that is, neither is superior to the other. Train is slower but cheaper than Plane.

TABLE 2.10 Transportation Attributes

Option	Journey Time (h)	Cost Per Mile ($)
Car	3	1.5
Train	5	2.0
Plane	3.5	3.5
Executive coach	7	4.0

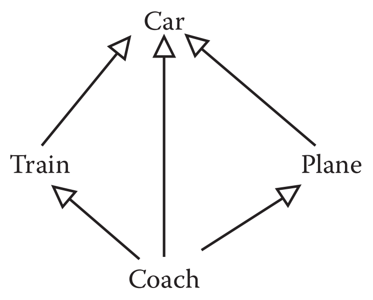

FIGURE 2.13 Quality relationships based on rule and collected data.

It would be inappropriate to force an ordering because of the different underlying attributes. We could impose an ordering only if we had additional information about the relative priorities of cost and timing. If cost is more important to us, then Train is preferable; if speed is more important, we would prefer Plane.

Now suppose we wish to use the representation condition to define a measure to characterize the notion of journey quality given by the above relation system. It is easy to prove that there is *no* possible measure that is a single-valued real number. Suppose that such a measure exists, then Plane would be mapped to some real number m(Plane), while Train would be mapped to some real number m(Train). Then, exactly one of the following must be true:

1. $m(\text{Plane}) < m(\text{Train})$
2. $m(\text{Plane}) > m(\text{Train})$
3. $m(\text{Plane}) = m(\text{Train})$

If the first statement were true, then the representation condition implies that Plane must be superior to Train. This is false, because Train is cheaper. Similarly, the second statement is false because Train is slower than Plane. But the third statement is also false, since it implies an equality relation that does not exist.

The reason we cannot find a measure satisfying the representation condition is because we are looking at too narrow a number system. When we have genuinely incomparable entities, we have a *partial order*, as opposed to what is called a *strict weak order*, so we cannot measure in the set of real numbers. (A *strict weak order* has two properties: it is asymmetric and negatively transitive. By *asymmetric*, we mean that if the pair (x,y) is in the relation, then (y,x) is not in the relation. A relation is *negatively transitive* if, whenever (x,y) is in the relation, then for every z, either (x,z)

or (z,y) is in the relation.) What we need instead is a mapping into *pairs* of real numbers, that is, into the set . In the transport example, we can define a representation in the following way. First, we define a measure m that takes a transport type into a pair of elements:

$$m(\text{Transport}) = (\text{Journey time, Cost per mile})$$

Then, we define the actual pairs:

$$m(\text{Car}) = (3, 1.5)$$

$$m(\text{Train}) = (5, 2)$$

$$m(\text{Plane}) = (3.5, 3.5)$$

$$m(\text{Coach}) = (7, 4)$$

The numerical binary relation over that corresponds to the empirical superiority relation is defined as:

$$(x,y) \text{ superior to } (x',y') \quad \text{if } x < x' \quad \text{and} \quad y < y'$$

The numerical relation preserves the empirical relation. That too is only a partial order in because it contains incomparable pairs. For example, the pair (5,2) is not superior to (3.5,3.5); nor is (3.5,3.5) superior to (5,2).

EXAMPLE 2.26

Suppose we wish to assess the *quality* of four different C compilers. We determine that our notion of quality is defined in terms of two sub-attributes: *speed* (average KLOC compiled per second) and *resource* (minimum Kbytes of RAM required). We collect data about each compiler, summarized in Table 2.11.

Using the same sort of analysis as above, we can show that it is not possible to find a measure of this attribute in the real numbers that satisfies the representation condition.

These examples are especially relevant to software engineering. The International Standards Organization has published a standard, (ISO/IEC

TABLE 2.11 Comparing Four Compilers

	Speed	Resource
A	45	200
B	30	400
C	20	300
D	10	600

25010:2011), for measuring software quality that explicitly defines *software quality* as the combination of eight distinct sub-attributes. We will discuss the details of this standard in Chapter 10. However, it is important to note here that the standard reflects a widely held view that no single real-valued number can characterize such a broad attribute as quality. Instead, we look at n-tuples that characterize a set of n sub-attributes. The same observation can be made for *complexity* of programs.

EXAMPLE 2.27

Many attempts have been made to define a single, real-valued metric to characterize program complexity. For instance, in Example 2.9, we were introduced to one of the most well known of these metrics, the cyclomatic number. This number, originally defined by mathematicians on graphs, is the basis of an intuitive notion of program complexity. The number corresponds to an intuitive relation, *more complex than*, that allows us to compare program flowgraphs and then make judgments about the programs from which they came. That is, the cyclomatic number is a mapping from the flowgraphs into real numbers, intended to preserve the complexity relation. As we have seen in examining journey quality, if the relation *more complex than* is not a strict weak order, then cyclomatic number cannot be an ordinal scale measure of complexity. (Indeed, a theorem of measurement theory asserts that a strict weak order is a necessary *and* sufficient condition for an ordinal scale representation in $\mathcal{R}$.) We contend that no general notion of complexity can give rise to such an order. To see why, consider the graphs depicted in Figure 2.14. Flowgraph y represents a conditional choice structure, x represents a sequence of two such structures, and z represents a looping construct. Intuitively, it seems reasonable that graph x is more complex than graph y. If *more complex than* produced a strict weak order, we should be able to show that this relation is negatively transitive. That is, we should be able to show that for any z, either x is related to z or z is related to y. But neither of the following statements is obviously true:

x is more complex than z

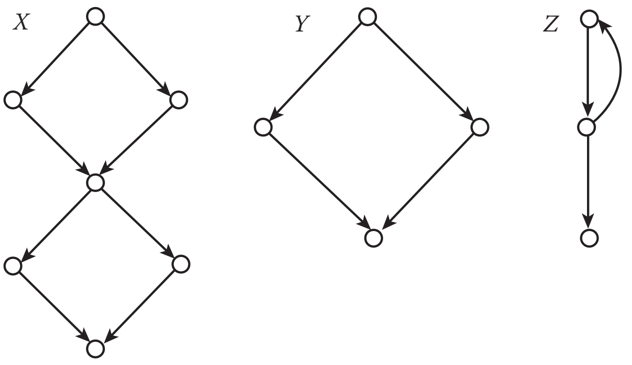

FIGURE 2.14 Three program flowgraphs.

and

z is more complex than y

Some programmers would argue that x is more complex than z; for instance, while others would say that z is more complex than x; we cannot reach consensus. In other words, some of the graphs are not comparable, so the relation is not a strict weak order and the cyclomatic number cannot be on an ordinal scale. Notice that the cyclomatic number for x is 3, for y is 2, and for z is 2, forcing us to conclude that x should be more complex than z. Thus, the cyclomatic number, clearly useful in counting the number of linearly independent paths in the program flowgraph, should not be used as a comprehensive measure of complexity.

In spite of theoretical problems, there are many situations when we must combine sub-attributes to impose a strict ranking, and hence an ordinal scale. That is, we need to define a single real-valued number as a measure. For example, if we are buying a coat, we may take into account the price, quality of material, fit, and color. But in the end, we are forced to determine preference. Consciously or subconsciously, we must define some combination of component measures to arrive at a preference ranking.

EXAMPLE 2.28

We want to buy a contact management program. It is likely that the decision will be based on a collection of attributes, such as price, reliability, and usability. If the program is for a single user's home PC, we may give price a heavier weighting than reliability when ranking the programs. However, if we are buying it for network use in a major organisation, it is likely that reliability would get a larger weighting than price.

Other, similar problems can arise. We may need to determine which program is safest, based on a set of criteria. Or we may wish to choose from among a variety of design techniques, based on survey data that captures developer preferences, design quality assessments, and cost of training and tools. Each of these instances presents a problem in making a decision with multiple criteria. There is an extensive literature on multi-criteria decision theory, and the measurement theory that relates to it. We discuss this type of analysis in Chapter 6, when we address data analysis techniques. However, here we must look at how to combine measures in a way that remains true to the spirit of measurement theory.

2.4.4 Derived Measurement and Meaningfulness

When we measure a complex attribute in terms of simpler sub-attributes, we are measuring indirectly. In doing so, we must adhere to the same basic rules of measurement theory that apply to direct measures. We must pay particular attention to issues of scale types and meaningfulness.

Scale types for derived measures are similar to those for direct ones. Our concerns include the uniqueness of the representation, as well as the admissible transformations for each scale type. We call an admissible transformation a rescaling, and we define rescaling in the following way. Suppose that we measure each of n sub-attributes with measure M_i. Let M be a derived measure involving components $M_1, M_2, \ldots, M_n$. That is, $M = f(M_1, M_2, \ldots, M_n)$ for some function f. We say that M' is a *rescaling* of M if there are rescalings $M_1', M_2', \ldots, M_n'$, of $M_1, M_2, \ldots, M_n$, respectively, such that $M' = f(M_1', M_2', \ldots, M_n')$.

Strictly speaking, this defines rescaling in the *wide sense*. Rescaling in the *narrow sense* requires us to verify that $M' = f(M_1, M_2, \ldots, M_n)$.

EXAMPLE 2.29

Density d is a derived measure of mass m and volume V. The specific relationship is expressed as

$$d = m/V$$

Every rescaling of d is of the form $d' = \alpha d$ (for $\alpha > 0$). To see why, we must demonstrate two things: that a function of this form is a rescaling, and that every rescaling has this form. For the first part, we have to find rescalings m' and V' of m and V, respectively, such that $\alpha d = m'/V'$. Both m and V are ratio scale measures, so αm and V are acceptable rescalings of m and V, respectively. Since

$$\alpha d = \alpha\left(\frac{m}{V}\right) = \frac{\alpha m}{V}$$

therefore we have a rescaling.

To show that every rescaling is of the appropriate form, notice that since m and V are ratio scale measures, every rescaling of m must be of the form $\alpha_1 m$ for some α_1 and every rescaling of V must be of the form $\alpha_2 V$ for some α_2. Therefore, every rescaling of d has the form

$$\frac{\alpha_1 m}{\alpha_2 V} = \frac{\alpha_1}{\alpha_2}\left(\frac{m}{V}\right) = \frac{\alpha_1}{\alpha_2} d = \alpha d \quad \left(\text{where } \alpha = \frac{\alpha_1}{\alpha_2}\right)$$

Now, we can define scale types for derived scales in exactly the same way as for direct scales. Example 2.29 shows us that the scale for density d is ratio, because all the admissible transformations have the form $d \leftarrow \alpha d$. In the same way, we can show that the scale type for a derived measure M will generally be no stronger than the weakest of the scale types of the M_is. Thus, if the M_is contain a mixture of ratio, interval, and nominal scale types, then the scale type for M will at best be nominal, since it is weakest.

EXAMPLE 2.30

A derived measure of testing efficiency T is D/E, where D is the number of defects discovered and E is effort in person-months. Here D is an absolute scale measure, while E is on the ratio scale. Since absolute is stronger than ratio scale, it follows that T is a ratio scale measure. Consequently, the acceptable rescalings of T arise from rescalings of E into other measures of effort (person-days, person-years, etc.)

Many of the measures we have used in our examples are assessment measures. But derived measures proliferate as prediction measures, too.

EXAMPLE 2.31

In Example 2.12, we saw that many software resource prediction models predict effort E (in person-months) by using an equation of the form

$$E = aS^b$$

where S is a measure of software size, and a and b are constants. Some researchers have doubted the meaningfulness of these derived effort measures. For example, DeMillo and Lipton looked at the Walston and Felix model. Walston and Felix assert that effort can be predicted by the equation

$$E = 5.2S^{0.91}$$

where S is measured in lines of code (see Perlis et al. 1981). DeMillo and Lipton contend that the prediction equation is an example of a meaningless measure. They assert that "both E and S are expressed as a ratio scale… but the measurement is not invariant under the transformation SS and so is meaningless" (DeMillo and Lipton 1981). In fact, this argument is relevant only when we consider scales defined in the *narrow* sense. In the more usual wide sense, it is easy to show that the equation is meaningful and that the scale type for effort is ratio. However, demonstrating scale type and meaningfulness is very different from asserting that the relationship is valid.

Many times, models of effort involve several levels of derived measurement. That is, a derived measure is defined to be a combination of other measures, both direct and derived.

EXAMPLE 2.32

Halstead developed a theory of software physics (discussed in Chapter 8) that defines attributes as combinations of counts of operators and operands. His equation for software effort, E, is

$$E = V/L$$

where V, the program volume, is on a ratio scale, but L, the estimated program level, appears to be only on an ordinal scale. Thus, E cannot be a ratio scale. However, Halstead claims that E represents the number of mental discriminations necessary to implement the program, which is *necessarily* a ratio scale measure of effort. Therefore, Halstead's effort equation is not meaningful.

The unit in which the measure is expressed can affect the scale of the measure.

EXAMPLE 2.33

Consider another effort measure

$$E = 2.7v + 121w + 26x + 12y + 22z - 497$$

cited by DeMillo and Lipton (1981). E is supposed to represent person-months, v is the number of program instructions, and w is a subjective complexity rating. The value of x is the number of internal documents generated on the project, while y is the number of external documents. Finally, z is the size of the program in words. DeMillo and Lipton correctly point out that, as in Example 2.32, effort should be on a ratio scale, but it cannot be ratio in this equation because w, an ordinal measure, restricts E to being ordinal. Thus, the equation is meaningless. However, E could still be an ordinal scale measure of effort if we drop the pre-condition that E expresses effort in person-months.

In this chapter, we have laid a foundation of principles on which to base valid measurement. The next chapter builds on this foundation by introducing a framework for how to choose measures, based on needs and process.

2.5 SUMMARY

Measurement requires us to identify intuitively understood attributes possessed by clearly defined entities. Then, we assign numbers or symbols to the entities in a way that captures our intuitive understanding about the attribute. Thus, intuitive understanding of that attribute must precede direct measurement of a particular attribute. This intuitive understanding leads to the identification of relations between entities. For example, the attribute height for the entity person gives rise to relations like *is tall*, *taller than*, and *much taller than*.

To measure the attribute, we define corresponding relations in some number system; then measurement assigns numbers to the entities in such a way that these relations are preserved. This relationship between the domain and range relationships is called the *representation condition*.

In general, there may be many ways of assigning numbers that satisfy the representation condition. The nature of different assignments determines the scale type for the attribute. There are five well-known scale types: nominal, ordinal, interval, ratio, and absolute. The scale type for a measure determines what kind of statements we can meaningfully make using the measure. In particular, the scale type tells us what kind of operations we can perform. For example, we can compute means for ratio scale measures, but not for ordinal measures; we can compute medians for ordinal scale measures but not for nominal scale measures.

Many attributes of interest in software engineering are not directly measurable. This situation forces us to use vectors of measures, with rules for combining the vector elements into a larger, derived measure. We define scale types for these in a similar way to direct measures, and hence can determine when statements and operations are meaningful.

In the next chapter, we build on this foundation to examine a framework for measurement that helps us to select appropriate measures to meet our needs.

EXERCISES

1. At the beginning of this chapter, we posed four questions:

   a. How much we must know about an attribute before it is reasonable to consider measuring it? For instance, do we know enough about "complexity" of programs to be able to measure it?

   b. How do we know if we have really measured the attribute we wanted to measure? For instance, does a count of the number of "bugs" found in a system during integration testing measure the quality of the system? If not, what does the count tell us?

   c. Using measurement, what meaningful statements can we make about an attribute and the entities that possess it? For instance, is it meaningful to talk about doubling a design's quality? If not, how do we compare two different designs?

   d. What meaningful operations can we perform on measures? For instance, is it sensible to compute average productivity for a group of developers, or the average quality of a set of modules?

   Based on what you have learned in this chapter, answer these questions.

2. a. List, in increasing order of sophistication, the five most important measurement scale types.

   b. Suppose that the attribute "complexity" of software modules is ranked as a whole number between 1 and 5, where 1 means