estimation model, Putnam and Myers [PUT92] suggest a set of equations derived from the software equation. Minimum development time is defined as:

$$t_{\min} = 8.14(\text{LOC/PP})^{0.43} \text{ in months for } t_{\min} > 6 \text{ months} \tag{5.4a}$$

$$E = 180Bt^3 \text{ in person-months for } E \geq 20 \text{ person-months} \tag{5.4b}$$

Note that t in equation (5.4b) is represented in years.

Using equations (5.4) with $P = 12{,}000$ (recommended value for scientific software) for the CAD software discussed earlier in this chapter,

$$t_{\min} = 8.14\ (33{,}200/12{,}000)^{0.43}$$

$$t_{\min} = 12.6 \text{ calendar months}$$

$$E = 180 \times 0.28 \times (1.05)^3$$

$$E = 58 \text{ person-months}$$

The results of the software equation correspond favorably with the estimates developed in Section 5.6.

THE MAKE–BUY DECISION

In many software application areas, it is often more cost effective to acquire, rather than develop, computer software. Software engineering managers are faced with a *make–buy decision* that can be further complicated by a number of acquisition options: (1) software may be purchased (or licensed) *off-the-shelf;* (2) "full-experience" or "partial-experience" software components (see Section 5.4.2) may be acquired and then modified and integrated to meet specific needs, or (3) software may be custom-built by an outside contractor to meet the purchaser's specifications.

The steps involved in the acquisition of software are defined by the criticality of the software to be purchased and the end cost. In some cases (e.g., low-cost PC software), it is less expensive to purchase and experiment than to conduct a lengthy evaluation of potential software packages. For more expensive software products, the following guidelines can be applied:

1. Develop a specification for function and performance of the desired software. Define measurable characteristics whenever possible.
2. Estimate the internal cost to develop and the delivery date.

3a. Select three or four candidate applications that best meet your specification.

3b. Select reusable software components that will assist in constructing the required application.

4. Develop a comparison matrix that presents a head-to-head comparison of key functions. Alternatively, conduct benchmark tests to compare candidate software.

5. Evaluate each software package or component based on past product quality, vendor support, product direction, reputation, and so on.
6. Contact other users of the software and ask for opinions.

In the final analysis, the make–buy decision is made based on the following conditions: (1) Will the delivery date of the software product be sooner than that for internally developed software? (2) Will the cost of acquisition plus the cost of customization be less than the cost of developing the software internally? (3) Will the cost of outside support (e.g., a maintenance contract) be less than the cost of internal support? These conditions apply for each of the acquisition options noted above.

5.8.1 Creating a Decision Tree

The steps described above can be augmented using statistical techniques such as *decision tree analysis* [BOE89]. For example, Figure 5.6 depicts a decision tree for a software-based system, *X*. In this case, the software engineering organization can (1) build system *X* from scratch; (2) reuse existing "partial experience" components to construct the system; (3) buy an available software product and modify it to meet local needs; or (4) contract the software development to an outside vendor.

If the system is to be built from scratch, there is a 70 percent probability that the job will be difficult. Using the estimation techniques discussed earlier in this chapter, the project planner projects that a difficult development effort will cost $450,000. A "simple" development effort is estimated to cost $380,000. The expected value for cost, computed along any branch of the decision tree, is

$$\text{expected cost} = \sum (\text{path probability})_i \times (\text{estimated path cost})_i$$

where *i* is the decision tree path. For the *build* path,

$$\text{expected cost}_{\text{build}} = 0.30\ (\$380\text{K}) + 0.70\ (\$450\text{K}) = \$429\text{K}$$

Following other paths of the decision tree, the projected costs for *reuse, purchase,* and *contract,* under a variety of circumstances, are also shown. The expected costs for these paths are:

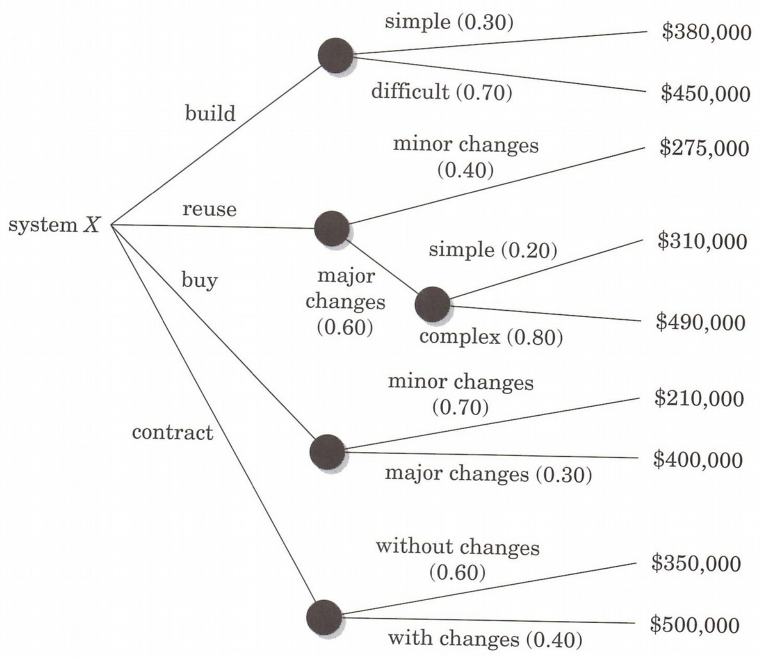
$$\text{expected cost}_{\text{reuse}} = 0.40\ (\$275\text{K}) + 0.60\ [0.20\ (\$310\text{K}) + 0.80\ (\$490\text{K})] = \$382\text{K}$$

$$\text{expected cost}_{\text{buy}} = 0.70\ (\$210\text{K}) + 0.30\ (\$400\text{K})] = \$267\text{K}$$

$$\text{expected cost}_{\text{contract}} = 0.60\ (\$350\text{K}) + 0.40\ (\$500\text{K})] = \$410\text{K}$$

Based on the probability and projected costs that have been noted in Figure 5.6, the lowest expected cost is the *buy* option.

It is important to note, however, that many criteria—not just cost—must be considered during the decision making process. Availability, experience of the developer/vendor/contractor, conformance to requirements, local "politics," and the likelihood of change are but a few of the criteria that may affect the ultimate decision to build, reuse, buy or contract.



URE 5.6. A
sion tree to
port the
e–buy
sion.

5.8.2 Outsourcing

Sooner or later, every company that develops computer software asks a fundamental question: "Is there a way that we can get the software and systems that we need at a lower price?" The answer to this question is not a simple one, and the emotional discussions that occur in response to the question always lead to a single word: "outsourcing."

In concept, outsourcing is extremely simple. Software engineering activities are contracted to a third party who does the work at lower cost, and hopefully, higher quality. Software work conducted within a company is reduced to a contract management activity.

Edward Yourdon [YOU92] discusses the broader implications of the growing trend toward outsourcing when he states:

> If you have been brought up in a culture that glorifies the American software industry as a world leader, I ask simply that you remember that it was only a few years ago that we had the same opinion of our automobile industry. . . . [S]oftware development may well move out of the U.S. into software factories in a dozen countries whose people are well educated, less expensive, and more passionately devoted to quality and productivity.