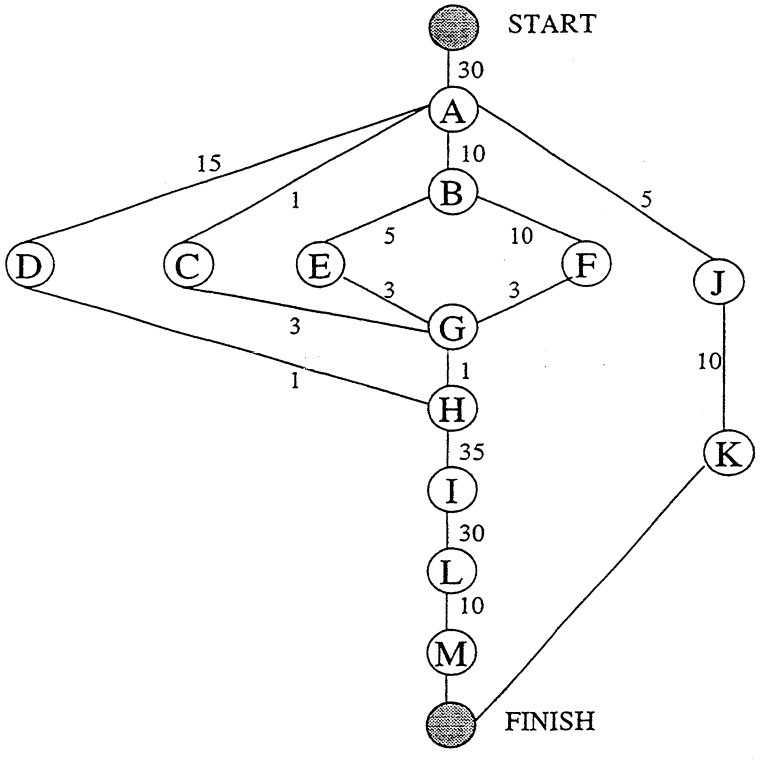
Chapter 3: Planning and managing the project

1.

Activities:	Duration (in minutes):
A. Choose recipe	30
B. Assemble cake ingredients	10
C. Prepare cake pan	2
D. Preheat oven	15
E. Mix dry ingredients	5
F. Mix wet ingredients	10
G. Fold wet ingredients into dry ingredients	3
H. Pour batter into pan	1
I. Bake cake	35
J. Assemble icing ingredients	5
K. Mix icing	10
L. Cool cake	30
M. Apply icing	10

Activity Graph:



Critical Path: A-B-F-G-H-I-L-M 129 minutes

2.

Activity leading to:	Precursors
A	
B	A
C	A
D	A,B
E	A
F	A,C
G	A,E
H	A,C,E,F,G
I	A,B,D
J	A,B,D,E,G,I
K	A..J
L	A..K

Activity leading from .. to:	Earliest start time	Latest start time	Slack
A..B	1	1	0
B..D	4	4	0
B..I	4	5	1
D..I	9	9	0
I..J	11	11	0
A..C	1	5	4
C..F	6	10	4
F..H	9	13	4
A..E	1	4	3
E..G	5	8	3
G..H	8	14	6
G..J	8	11	3
H..K	10	14	4
J..K	13	16	3
J..L	13	13	0
K..L	15	18	3
L	21		

Critical path: A..B..D..I..J..L 20 days

3. **Critical path**: A..B..C..E..D..I..K..L 24 days

4. Test planning can take place in parallel with requirements, design, and coding activities. Low-level design and coding on different parts of the system can take place in parallel. Activity graphs might

hide interdependencies because you cannot represent the fact that the results of a particular activity may require re-doing part of a previous activity. Also, a complex interdependency between two activities may not require that one be completed before the other. There are other types of interdependencies, e.g. information dependencies.

5. Adding personnel requires extra time for training as well as a superlinear increase in communication channels and thus communication costs. These added costs may outweigh the time savings from having extra people working on the project.

6. The Hardand estimate would be $5.25 * (20000)^{.91} = 43{,}062$ person-months. If the size estimate is 10% too low, then it will take 4,332 more person-months to complete. If we express k as a fraction, rather than a percentage, then the relationship between the original estimated size, S_e, and the actual size, S_t, is

$$S_t = S_e/(1-k)$$

and the relationship between the original estimate, E_e, and the actual needed time, E_t, is

$$\begin{aligned} E_t &= 5.25\, S_t^{.91} \\ &= 5.25\, (S_e/(1-k))^{.91} \\ &= (1/(1-k))^{.91} * 5.25\, S_e^{.91} \\ &= (1/(1-k))^{.91} * E_e \end{aligned}$$

If k is 10% (or 0.1), then the above formula yields 47,395 person-months, which is 4,332 more than the original estimate.

7. A utility program usually takes longer to develop than an applications program because the developer has to take into account all of the different environments in which it might be used. For instance, a sort routine must be callable from many different other programs, and it has to return its output in some "generic" way that can be used by an assortment of other components. On the other hand, an applications program is usually very specific. It has a much narrower scope, and the set of possible calling programs is much smaller. Its output is usually tailored for the very small number of other programs that need it. A systems program, being even more general than a utility program, has a larger set of possible callers, and it must be even more generic. So it can take a longer time to specify the set of calling and receiving programs, and it is likely to have a lot of cases to handle, especially in its error-handling.

8. Factors to consider include:

   - how long it would take to build in-house;
   - how soon the software is needed
   - how many programmers would be needed;
   - whether or not the technical and managerial expertise is available in-house;
   - the quality of the purchased software;
   - maintenance costs, both in-house and through the vendor

   It is generally better to build when the expertise is already available in-house, when the organization is willing and able to take on maintenance of the system, or when there is uncertainty about whether or not the purchased software would meet quality requirements. It is generally better to buy when the software is needed sooner than it can be built in-house, or when it can be bought much more cheaply than it can be built.

9. Yes and no. Development time, strictly defined, may be shortened by adding more people, but actual project time would probably increase due to increased training, communication, and management effort. It also depends a great deal on when in the project lifecycle people are added. A project that starts out with more people will probably finish earlier than if it had started out with fewer people. However, adding those people nearer to the end of the project is less likely to be beneficial.

10. The Bailey & Basili model reflects the first factor through the cost factor "customer initiated program design changes." COCOMO Stage 3 addresses the latter through the factor "personnel continuity."

11. Teams of three students each implemented an automatic car wash control system in 90 days as part of a software engineering course. Below are some of the risks, the corresponding exposures, and possible mitigating actions.

Risk	Exposure	Mitigating actions
unable to agree on design	(.30)(10) = 3 days	pick one chief designer who makes all decisions
one member not working	(.10)(30) = 3 days	choose team members; appeal to professor
many bugs found in testing	(.40)(5) = 2 days	inspections

12. Unit size per unit time is an unsatisfactory measure of productivity for several reasons. Unfortunately, it is pretty much the only measure available. Some of the reasons it is inadequate:

    - programmer output varies widely
    - size per time varies according to application language, application domain, difficulty, and newness of problem
    - amount of reuse also affects productivity
    - management style and structure
    - programmer style can vary according to the targets given to them
    - values for size are not available until long after estimates are needed, and size estimates can be very inaccurate

    Consequently, it would be very helpful to have a measure for productivity that overcame all of these shortcomings. Furthermore, we need to understand some of the surrounding issues, such as the effect of productivity targets, management style, and programmer expertise on productivity, no matter how it is measured.