Chapter 12: Evaluating products, processes and resources

1. A set of facets with which to start could be:
   - phase in the development cycle
   - programming language
   - application domain
   - intended audience
   - type of book

   It is very difficult to know when enough facets have been defined. The set of facets can be tested by use to see if they are at the right level of detail, if they cover all relevant aspects, and if there are enough of them. There are enough when each book can be accurately described and when any two books with the same values for all facets really are very similar. The degree of similarity between two such books depends on the level of detail of the classification. At the lowest level of detail (for example, if author and title are both facets), all book descriptions will be unique. An example of such a detailed cataloguing system is the Library of Congress system for books. In the LOC system, every book has a unique identifier. The LOC system is also a faceted system because every part of the identifier actually refers to some particular attribute, or facet, of the book.

2. The cost of reusing software includes the cost of developing that software to begin with (the cost of "producer reuse") and the cost of adapting that software into future systems (the cost of "consumer reuse"). Producer and consumer reuse take place in different projects, so the total cost of reuse must take into account costs in more than one project.

3. Some examples:
   - For each time the component was reused, a description of the modifications that had to be made to the component – this would help potential reusers know what types of modifications might have to be made in order to make the component work in their system.
   - For each time the component was reused, the effort required to modify it – this would help potential reusers know how much effort they might plan on expending to reuse the component.
   - Error history – this would give an idea of the quality of the component.
   - Total number of times the component was reused – this would give managers and analysts an idea of the most heavily used components, with the aim of improving those components with the largest impact.

4. Some possibilities are:
   - plan further training for that developer,
   - assign that developer to more appropriate tasks in the future
   - change hiring practices so that developers with similar deficiencies will not be hired in the future

5. Some questions you might use to answer the last part of the question:
   - Will doing these things differently lower the cost of the next project?
   - Will doing these things differently make the next project shorter?
   - Will doing these things differently improve the quality of the next product?
   - Will doing these things differently decrease the risk of unwanted "surprises" during the next project, especially late in the project?
   - Will doing these things differently decrease the amount of stress and frustration experienced by the personnel in the next project?
   - Will doing these things differently improve our relationship with the customer on the next project?
   - Will doing these things differently cause other problems on the next project that we didn't have on this project?

   It is not likely that you will be able to answer any of these questions definitively because many improvements must be tried before it is known whether they will work or not. This type of evaluation is part of the job of case studies and experiments.

6. One example from Boehm's model is accuracy, which could be measured objectively as the percentage of test cases passed. On the other hand, communicativeness might be measured subjectively by usability testers. An example from the ISO 9126 model is testability, which is sometimes measured objectively by measuring "ambiguous phrases" in the requirements document. There are specific lexical rules for ambiguous phrases, such as the use of "should" or "can." The idea is that if the requirements document is ambiguous, then testing will be harder. A subjective example from Dromey's model might be reusability, measured using developer ratings.

7. The inconsistency between the one-letter codes used on most aeronautical charts and those expected by the software system is really where the error lies. Such an error has to do with defining the system boundary and recognizing the interaction between the system within the boundary and systems that lie outside the boundary. Boehm's model addresses some attributes that, had they been explicitly tested for in this system, might have revealed the error (e.g. consistency, robustness/integrity). However, the model does not explicitly deal with the system's relationship to its environment. The same can be said for the ISO 9126 and the Dromey models, although some of the portability attributes, in particular conformance in the ISO model, might have been used to test for such errors. Another way to view this error is as a usability problem. That is, with a very narrow view of the system boundary, the pilot can be said to have caused the failure because he entered invalid input. All the quality models address usability, and two of them list an attribute communicativeness which seems to relate to this problem. Communicating to the user the effects of his/her actions could have ameliorated the effects of the error. It is unlikely that measurement could have helped to avoid this particular problem, but it can be used during usability and other types of testing to measure the number and severity of errors which occur, in order to evaluate the reliability and usability of the system and how likely it is to permit a severe error. Also, safety analysis techniques, such as FMEA and HAZOPS, applied to the design or to the system before delivery might have caught the problem before it took lives.

8. Boehm's model is different from the other two because it includes the views of different types of developers and users. That is, those who are going to port or maintain the system can be viewed as both developers and users, and their viewpoints are included in Boehm's model. In addition, Boehm's model really begins with a model of usability, so the user view takes priority. The ISO 9126 model takes a completely user view, in that both levels of characteristics are user-oriented, although the model does not provide ways to measure these characteristics, from either a developer or user point of view. Also, the ISO 9126 model is the only one that addresses security, which may be a user concern.

9. It makes sense to have a general model because it does facilitate the comparison of different systems by standardizing terminology. However, although probably any product can utilize the ISO 9126 model to some extent, many products will be better evaluated by a tailored version of the model which eliminates some factors that are irrelevant and adding others that are relevant to that particular product. More unusual products will require more tailoring of the model, and it is conceivable that there are products for which ISO 9126 could be said not to apply at all.

10. Security is a component of functionality in the ISO 9126 model, so ensuring the security of the system is considered one of the functions that the system is meant to provide. This means that the security needs of the system must be specified along with the other functional requirements, and then the resulting product must be tested against those security requirements to see if they are satisfied. This approach implies that security is something that either exists in the system or it doesn't. If it doesn't, the system does not function as specified. Another view of security is that it is a continuum, i.e. that a system can exhibit some level of security, just as it can exhibit some level of reliability. Requirements can then be specified for the level of security of the system, rather than specific security functions. With this view of security, it makes more sense to relate security to reliability in the ISO 9126 quality model.

11. The effectiveness of the review technique could be evaluated with an experiment in which two projects are compared. In one project, the technique would be used and in the other it would not. The quality of the resulting requirements documents and other downstream products could be measured to evaluate the effect of the technique. Ideally, to control all variables, all aspects of the two projects would have to be identical: same personnel, problem, customer, budget, schedule, etc. Practically speaking, this is not possible, but it is important that the two projects be as similar as possible. A good compromise would be to compare two projects with personnel of similar backgrounds, the same application domain, the same computing environment (hardware, supporting software, etc.), approximately the same size, etc. Another approach would be to use the same personnel on both projects, to make sure that differences between personnel do not confound the results. In this case, the first project would

have to be the one not using the review technique. Then, the personnel would have to be taught the technique, then given another similar (but not the same) project to work on.

12. For example, a goal that would help fulfill the "software project planning" KPA would be to "expend between 10% and 15% of each project's budget on planning activities." This goal is measurable using effort data. The growth, and then stability, of this percentage over time would reflect the growing maturity of the process. Another example would be the "organization process definition" KPA, which could have the corresponding goal of "100% of all projects use the organization's process definition." This is also measurable, and the growth in the measure towards 100 would indicate increasing process maturity in this area.

13. One way to test this hypothesis is to examine data from a large set of projects, which varied in terms of team cohesion and resulting quality, but were similar in other respects. Team cohesion could be measured subjectively by surveying project team members. The survey could include such questions as "How much (on a scale of 1 to 5) did you enjoy working with the other team members?," "How likely are you to choose to work with the same team again?," etc. Product quality could be measured objectively using defect data, or subjectively by surveying users about their satisfaction with the products.