Quality & Productivity Factors

Software quality and programmer productivity can be improved by improving the processes used to develop and maintain software products. Some factors that influence quality and productivity are listed as below:

1. ***Individual ability***:

Production and maintenance of software products are labour-intensive activities. Productivity and quality are thus direct functions of individual’s ability and effort.   
On very large and extremely large projects, the number of programmers is so large that individual differences in programmer productivity will tend to average output. However, the modules developed by weaker programmers may exhibit poor quality and may lag in delivery time.

In programming, the general guideline is to utilize outstanding people. However, it is not always possible to hire exceptional individuals.

1. ***Team communication***:

Increasing product size result in decreasing programmer productivity due to the increased complexity of interactions among program components and due to the increased communication required among programmers, managers and customers.

Increasing the number of team members increases the number of communications paths.

1. ***Product complexity***:

There are three generally acknowledged levels of product complexity:

* + *Application Programs*: It includes scientific and data processing routine written in high-level language *i.e.* COBOL, PASCAL, JAVA, C++ etc.
  + *Utility Programs*: It includes compilers, assemblers, linkage editors and loaders. They may be written in a high-level language such as Pascal or Ada or in assembly language.
  + *System Programs*: It s include data communication packages, real-time process control systems and operating systems routines which are usually written in assembly language or a high-level systems language such as PL/1 or Ada.

Utility programs can be produced at a rate 5 to 10 times and application programs at a rate 5 to 10 times that of systems programs. The effort required to develop and maintain a software product is a non linear function of product size and complexity.

1. ***Appropriate notations***: In software engineering the representation schemes are of fundamental importance. Good notations can clarify the relationships and interaction of interest while poor notations complicate and interfere with good practice.

Appropriate notations provide vehicles of communication among project personnel and introduce the possibility of using automated software tools to manipulate the notations and verify proper usage

1. ***Systematic approaches***: A number systematic approaches to software development and maintenance a discussed in this text. It is quite unreasonable to expect that a single approach to soft development and maintenance will ever be adequate to cover all situations. At this point in the evolution of software engineering, it is often not clear which of the various approaches to software development should be used in which situations.
2. ***Change control***: Requirements can also change due to poor understanding of the problem or external economic and political factors beyond the control of customer or developer.

The deviating effects of constantly changing requirements can be minimized by planning for change and by formalizing the mechanism of change. Planning for a software product must include plans for change control.

1. ***Level of technology***: The level of technology utilized on a software projects accounts for such factors as the programming language, the machine environment, the programming practices and the software tools.

Modern programming languages provide improved facilities for data definition and data usage, improved constructs for specifying control flow, better modularization facilities, user-defined exception handling and facilities for concurrent programming.

Stability and availability of the machine environment exert a strong influence on productivity and quality.

Software tools span the spectrum from elementary tools such as assemblers and primitive debugging aids to fully integrate developed environments that incorporate tools for managing and controlling the software development process.

1. ***Required reliability***: Every software must possess a basic level of reliability: However extreme reliability is gamed only with great care an analysis, design implementation system testing and maintenance of the software product. Both human and machine resources are required to obtain increased reliability.
2. ***Available time***: While it might appear-that a software project requiring six programmer-months of effort can be completed by one programmer in 6 months *or by* six programmers in month. Software projects are sensitive not only to total effort but also to elapsed time and because the effort required for coordination and communication among six programmers will drastically increase project overhead.

On the other hand, using two programmers for 3 months may be more effective than using one programmer for 6 months due to the reinforcement that each programmer gains from the other.

Programmer productivity is also sensitive to the calendar time available for project completion. There is a quantitative evidence to suggest that development time cannot be compressed below about 75 percent of the nominal development time regardless of the personnel and resources expended.

Determining optimum staffing levels and proper elapsed time for various activities in software product development is in important and difficult aspect of cost and resource estimation.

1. ***Problem understanding***: Failure to understand the true nature of the problem to be solved is a common and difficult issue.

Often the software engineer does not understand the application area and has trouble communicating with the customer because of differences in educational backgrounds, viewpoints and technical jargon.

Careful planning, customer interviews, task observation, prototyping, a preliminary version of the user’s manual and precise product specifications can increase both customer and developer understanding of the problem to be solved.

1. ***Stability of requirements***: Extracting information from customers in order to establish user needs demands good communication skills, tact and diplomacy as well as knowledge of the application area. Requirements definition and design activities are conceptual in nature and require good problem-solving skill.

Implementation of software requires concentrated attention to details Debugging often require deductive skills.

Project manager must be patient enough to test the project with all possible test data. He/ She must not get satisfied if the project works well for a smallest set of test data. The software with many defects must not be presented to the customer.

1. ***Required skills***: It is not necessary that every software engineer possess all the requisite skills, but the basic (*i.e.* fetching requirement from customer, managing & monitoring software project, providing instant and robust solution etc.) skills must be present among the members of a programming team.

Software project managers must be effective in dealing with the factors that motivate and frustrate programmers if they are to maintain high product quality, high programmer productivity and high job satisfaction.

1. ***Adequacy of training***: *Why do the entry level programmers undergo a training program while taking up job?* Many educational institutions train their students as computer scientists but not a software engineers. That is, the students know the implementation phase only.

Implementation is only one aspect of software engineering. Software engineering includes not only the product implementation but also the analysis, design, test, verification documentation and maintenance of software products.

As the entry-level programmers are lacking the following skills:

* + Express oneself clearly in English.
  + Develop and validate software requirements and design specifications.
  + Work within application area.
  + Perform software maintenance.
  + Perform economic analyses
  + Work with project management techniques
  + Work in group

1. ***Management’s skills***: Software projects are often supervised by managers who have little, if any knowledge of software engineering. Managers experienced in management of computer hardware projects find software projects management to be difficult due to the differences in design methods, notations and development tools. Often a software engineer must report to hardware engineer. These are indeed an unfortunate situation for both manager and manage.

*O*n the other hand, the practice of "Promoting" technically competent individuals who have little training into software project management often produces equally unsatisfactory results. This may be due to the low social needs of many software people and the need for good social skills in managers.

1. ***Appropriate goals***: The primary goal of software engineering is the development of software products; that are appropriate for their intended use. Every software product should provide optimal levels of generality, efficiency and reliability, an appropriate trade-off between productivity and quality factors can be achieved by adhering to the goals and requirements established for the software product during project planning.
2. ***Rising expectations***: There are two interrelated aspects of rising expectations:
   * *First,* is the concern for how much functionality, reliability and performance can be provided by a given amount of development effort, and
   * *Second,* is the issue of fundamental limitations of software technology?

However the diversity, size and complexity of software applications are growing at a much faster rate than our ability to cope with the increasing demand.