# Solutions: Chapter 1: Software and Software Engineering

* 1. Classic examples include the use of "digital automobile dashboards" to impart a high tech, high quality images. Appliances that "think;" the broad array of consumer electronics; personal computers (today, differentiated more by their software function than the hardware), industrial instrumentation and machines. All e-commerce applications are differentiated by software.
	2. This is a good problem for classroom discussion (time permitting). Rather than focusing on cliché' ridden (albeit important) issues of privacy, quality of life, etc., you might want to discuss "techno fright" and how software can help to exacerbate or remedy it. Another interesting possibility is to use Neumann's "Risks" column in SEN to key discussion. You might also consider new attempts at software-based ‘cash’ economies, new modes of interactive entertainment, virtual reality, e-commerce, etc.
	3. It takes software so long to be finished, because of following reasons
1. Facilities are not available on line.
2. Development tools do not work as expected.
3. Customer insists on the new requirements, requiring redesign and rework.
4. Product depends on the government regulations that change unexpectedly.
5. Strict requirements for compatibility with existing system require more testing, design, and implementation then expected.
6. Requirements to operate under multiple operating systems take longer to satisfy than expected.
7. Software project risk management takes more time then expected.
8. Dependency on a technology that is still under development lengthens the schedule.

Development costs are high:

1. Unacceptably low quality requires more testing, design and implementation work to correct then expected.
2. Development of the wrong software functions requires redesign and implementation.
3. Development of the wrong user interface results in redesign and implementation.
4. Development of extra software functions that are not required extends the schedule.

We can’t find errors before we give the software to our customer because of the following reasons;

1. Product depends on government regulation, which changes unexpectedly.
2. Product depends on draft technical standards, which change unexpectedly.
3. New development personnel sometimes are added late in the project.
4. Conflicts within teams sometimes results in poor communication and hence poor design
5. Sabotage by project management results in efficient scheduling and ineffective planning.
6. Sometimes the furnished components are poor quality resulting in extra testing, design and integration work and in extra customer –relationship management.

We continue to have difficulty in measuring progress as software is

 developed as;

a) Sometimes the purpose of the project is not clear.

b) There are a lot of business risks involved.

c) If the product built is not fitted well.

d) We need to review our work continuously.

e) A time check has to be maintained.

f) Project team has to be thorough and organized throughout the process.

* 1. Many modern applications change frequently before they are presented to the end user and then after the first versions have been used. The few ways to build software to stop deterioration due to change would be:

Gather the required information.

Designer and customer define the overall objectives for the software.

Identify the known requirements.

After building a prototype the developer uses an existing program fragment, this will help the working program to complete quickly.

To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations.

Documents should be developed in a timely manner, to do this documentation standards are defined and mechanisms are established.

Review works done up to a particular stage.

There should be a backup person for every critical team member.

Check whether the risk aversion steps are being properly applied or not.

Check whether the necessary information for future risk analysis is necessary to collect.

* 1. The same approach to software engineering can be applied for each of the seven categories. Each of these “new challenges” will undoubtedly have effects (for business people, software engineers, and end-users) that cannot be predicted today. However, software engineers can prepare by instantiating a process that is agile and adaptable enough to accommodate dramatic changes in technology and business rules that are sure to come over the next decade.
	2. There are literally dozens of real life circumstances to choose from. For example, software errors that have caused major telephone networks to fail, failures in avionics that have contributed to plane crashes, computer viruses (e.g., Michelangelo) that have caused significant economic losses and attacks on major e-commerce sites.
	3. Process framework is applicable to all the projects; hence the same work tasks are applied for all projects, regardless of their size or complexity. A process framework involves heavy communication with the customer to gather requirements; this activity establishes a plan for the software engineering work that follows. It involves creation of models that will assist the developer and the customer to understand the requirements and design them; it thereby involves construction (code generation and error testing). It finally provides feedback based on the evaluation.
	4. The umbrella activities occur throughout the software process they are applied evenly across the process, the analysis encompass a set of work tasks (eg. requirement gathering, elaboration, negotiation specification and validation). A process framework has a set of umbrella activities that are applicable across the entire software process. These activities include Software project tracking and control, Risk management, Software quality assurance, and formal technical reviews, measurement, Software configuration management, reusability management and work product preparation and production.