ii. Software doesn't "wear out."

(Question: Justify the implication that the software doesn't wear out but deteriorate. – 3 Marks)

1. Figure 1.1 explains the failure rate as a function of time for hardware.

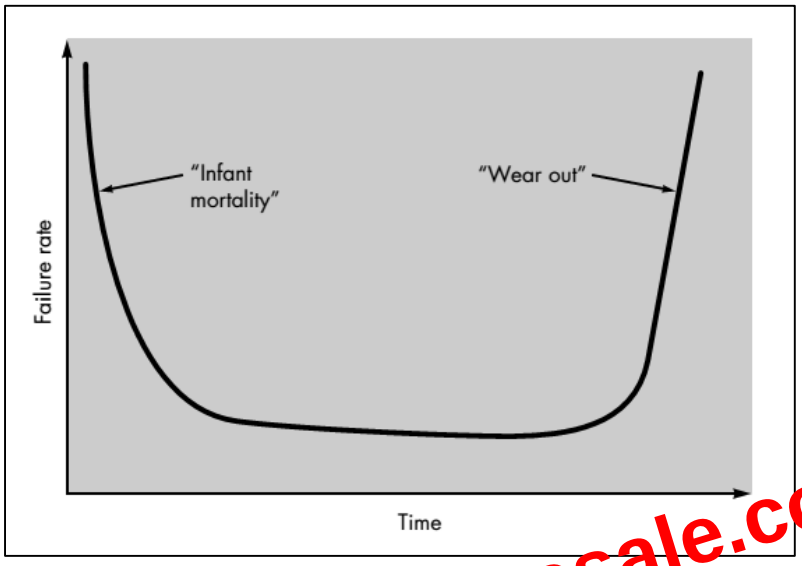

Figure 1.1

2. The relationship, called the "bathtub curve," indicates that hardware exhibits relatively high failure rates early in its life; defects are corrected and the failure rate drops to a steady-state level, ideally quite low for some period of time.
3. With time the failure rate rises again as hardware components suffer from the cumulative effects of dust, vibration, abuse, temperature extremes, and many other environmental maladies.
4. Stated simply, the hardware begins to wear out.
5. Software is not susceptible to the environmental maladies that cause hardware to wear out.
6. The failure rate curve for software should take the form of the "idealized curve" shown in Figure 1.2.
7. Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected (ideally, without introducing other errors) and the curve flattens as shown.
8. The idealized curve is a gross oversimplification of actual failure models for software.
9. The implication is clear—software doesn't wear out. But it does deteriorate!

4. Computer-aided design, system simulation, and other interactive applications have begun to take on real-time and even system software characteristics.

5. Embedded Software

1. Intelligent products have become commonplace in nearly every consumer and industrial market.
2. Embedded software resides in read-only memory and is used to control products and systems for the consumer and industrial markets.
3. Embedded software can perform very limited and esoteric functions, for example: keypad control for a microwave oven.
4. To provide significant function and control capability, for example: digital functions in an automobile such as fuel control, dashboard displays, and braking systems.

6. Personal Computer Software

1. The personal computer software market has burgeoned over the past two decades.
2. Word processing, spreadsheets, computer graphics, multimedia, entertainment, database management, personal and business fi applications, external network, and database access are only a few of hundreds of applications.

7. Web-based Software

1. The Web pages retrieved by a browser are software that incorporates executable instructions and data.

III. Software Engineering

1. Definition:

Software Engineering is the study and application of engineering to the design, development, and maintenance of software. Software engineering can be divided into ten sub disciplines. They are:

1. **Software requirements**: The elicitation, analysis, specification, and validation of requirements for software.
2. **Software design**: The process of defining the architecture, components, interfaces, and other characteristics of a system or component. It is also defined as the result of that process.
3. **Software construction**: The detailed creation of working, meaningful software through a combination of coding, verification, unit testing, integration testing, and debugging.
4. **Software testing**: The dynamic verification of the behavior of a program on a finite set of test cases, suitably selected from the usually infinite executions domain, against the expected behavior.

4. Tools Layer

1. Software engineering tools provide automated or semi-automated for the process and the methods.
2. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called computer-aided software engineering, is established. CASE combines software, hardware, and a software engineering database.

VI. Software Development Generic Process Framework

1. **Software Process:** A software process can be characterized as shown in Figure 1.5. A common process framework is established by defining a small number of framework activities that are applicable to all software projects, regardless of their size or complexity.
2. **Software Product:** A number of task sets—each a collection of software engineering work tasks, project milestones, work products, and quality assurance points—enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team.
3. **Basic Framework Activities:** Finally, umbrella activities—such as software quality assurance, software configuration management, and measurement to overlay the process model.
4. **Umbrella Activities:** Umbrella activities are independent of any one framework activity and occur throughout the process.

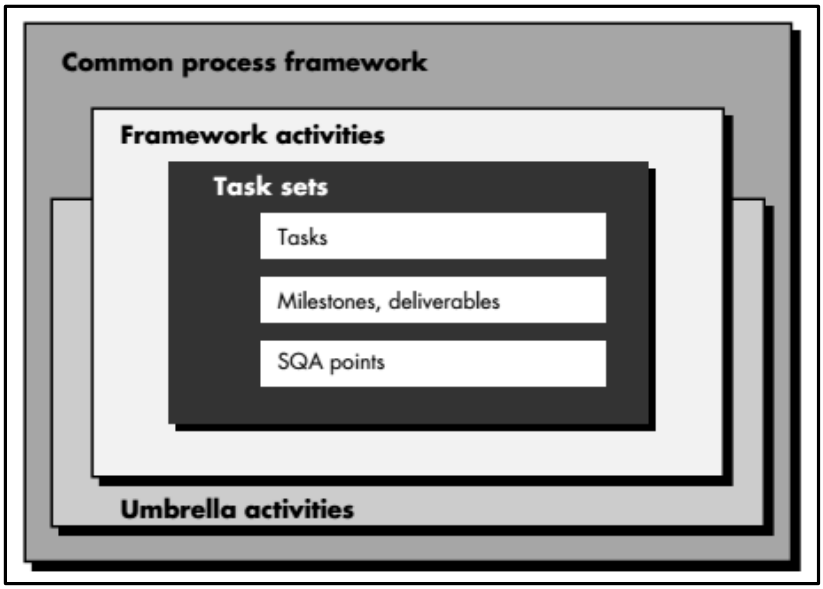

Figure 1.5