3 Characteristics and Quality Attributes of Embedded Systems

LEARNING OBJECTIVES

- ✓ *Learn the characteristics describing an embedded system*
- ✓ *Learn the non-functional requirements that needs to be addressed in the design of an embedded system*
- ✓ *Learn the important quality attributes of the embedded system that needs to be addressed for the operational mode (online mode) of the system. This includes Response, Throughput, Reliability, Maintainability, Security, Safety, etc.*
- ✓ *Learn the important quality attributes of the embedded system that needs to be addressed for the non-operational mode (offline mode) of the system. This includes Testability, Debug-ability, Evolvability, Portability, Time to prototype and market, Per unit cost and revenue, etc.*
- ✓ *Understand the Product Life Cycle (PLC)*

No matter whether it is an embedded or a non-embedded system, there will be a set of characteristics describing the system. The non-functional aspects that need to be addressed in embedded system design are commonly referred as quality attributes. Whenever you design an embedded system, the design should take into consideration of both the functional and non-functional aspects. The following topics give an overview of the characteristics and quality attributes of an embedded system.

3.1 CHARACTERISTICS OF AN EMBEDDED SYSTEM

Unlike general purpose computing systems, embedded systems possess certain specific characteristics and these characteristics are unique to each embedded system. Some of the important characteristics of an embedded system are:

1. Application and domain specific
2. Reactive and Real Time
3. Operates in harsh environments
4. Distributed
5. Small size and weight
6. Power concerns

3.1.1 Application and Domain Specific

If you closely observe any embedded system, you will find that each embedded system is having certain functions to perform and they are developed in such a manner to do the intended functions only. They cannot be used for any other purpose. It is the major criterion which distinguishes an embedded system from a general purpose system. For example, you cannot replace the embedded control unit of your microwave oven with your air conditioner's embedded control unit, because the embedded control units of microwave oven and airconditioner are specifically designed to perform certain specific tasks. Also you cannot replace an embedded control unit developed for a particular domain say telecom with another control unit designed to serve another domain like consumer electronics.

3.1.2 Reactive and Real Time

As mentioned earlier, embedded systems are in constant interaction with the Real world through sensors and user-defined input devices which are connected to the input port of the system. Any changes happening in the Real world (which is called an Event) are captured by the sensors or input devices in Real Time and the control algorithm running inside the unit reacts in a designed manner to bring the controlled output variables to the desired level. The event may be a periodic one or an unpredicted one. If the event is an unpredicted one then such systems should be designed in such a way that it should be scheduled to capture the events without missing them. Embedded systems produce changes in output in response to the changes in the input. So they are generally referred as Reactive Systems.

Real Time System operation means the timing behaviour of the system should be deterministic; meaning the system should respond to requests or tasks in a known amount of time. A Real Time system should not miss any deadlines for tasks or operations. It is not necessary that all embedded systems should be Real Time in operations. Embedded applications or systems which are mission critical, like flight control systems, Antilock Brake Systems (ABS), etc. are examples of Real Time systems. The design of an embedded Real time system should take the worst case scenario into consideration.

3.1.3 Operates in Harsh Environment

It is not necessary that all embedded systems should be deployed in controlled environments. The environment in which the embedded system deployed may be a dusty one or a high temperature zone or an area subject to vibrations and shock. Systems placed in such areas should be capable to withstand all these adverse operating conditions. The design should take care of the operating conditions of the area where the system is going to implement. For example, if the system needs to be deployed in a high temperature zone, then all the components used in the system should be of high temperature grade. Here we cannot go for a compromise in cost. Also proper shock absorption techniques should be provided to systems which are going to be commissioned in places subject to high shock. Power supply fluctuations, corrosion and component aging, etc. are the other factors that need to be taken into consideration for embedded systems to work in harsh environments.

3.1.4 Distributed

The term distributed means that embedded systems may be a part of larger systems. Many numbers of such distributed embedded systems form a single large embedded control unit. An automatic vending machine is a typical example for this. The vending machine contains a card reader (for pre-paid vending systems), a vending unit, etc. Each of them are independent embedded units but they work together

to perform the overall vending function. Another example is the Automatic Teller Machine (ATM). An ATM contains a card reader embedded unit, responsible for reading and validating the user's ATM card, transaction unit for performing transactions, a currency counter for dispatching/vending currency to the authorised person and a printer unit for printing the transaction details. We can visualise these as independent embedded systems. But they work together to achieve a common goal.

Another typical example of a distributed embedded system is the Supervisory Control And Data Acquisition (SCADA) system used in Control & Instrumentation applications, which contains physically distributed individual embedded control units connected to a supervisory module.

3.1.5 Small Size and Weight

Product aesthetics is an important factor in choosing a product. For example, when you plan to buy a new mobile phone, you may make a comparative study on the pros and corns of the products available in the market. Definitely the product aesthetics (size, weight, shape, style, etc.) will be one of the deciding factors to choose a product. People believe in the phrase "Small is beautiful". Moreover it is convenient to handle a compact device than a bulky product. In embedded domain also compactness is a significant deciding factor. Most of the application demands small sized and low weight products.

3.1.6 Power Concerns

Power management is another important factor that needs to be considered in designing embedded systems. Embedded systems should be designed in such a way as to minimise the heat dissipation by the system. The production of high amount of heat demands cooling requirements like cooling fans which in turn occupies additional space and make the system bulky. Nowadays ultra low power components are available in the market. Select the design according to the low power components like low dropout regulators, and controllers/processors with power saving modes. Also power management is a critical constraint in battery operated application. The more the power consumption the less is the battery life.

3.2 QUALITY ATTRIBUTES OF EMBEDDED SYSTEMS

Quality attributes are the non-functional requirements that need to be documented properly in any system design. If the quality attributes are more concrete and measurable it will give a positive impact on the system development process and the end product. The various quality attributes that needs to be addressed in any embedded system development are broadly classified into two, namely 'Operational Quality Attributes' and 'Non-Operational Quality Attributes'.

3.2.1 Operational Quality Attributes

The operational quality attributes represent the relevant quality attributes related to the embedded system when it is in the operational mode or 'online' mode. The important quality attributes coming under this category are listed below:

1. Response
2. Throughput
3. Reliability
4. Maintainability
5. Security
6. Safety

3.2.1.1 Response Response is a measure of quickness of the system. It gives you an idea about how fast your system is tracking the changes in input variables. Most of the embedded systems demand fast response which should be almost Real Time. For example, an embedded system deployed in flight control application should respond in a Real Time manner. Any response delay in the system will create potential damages to the safety of the flight as well as the passengers. It is not necessary that all embedded systems should be Real Time in response. For example, the response time requirement for an electronic toy is not at all time-critical. There is no specific deadline that this system should respond within this particular timeline.

3.2.1.2 Throughput Throughput deals with the efficiency of a system. In general it can be defined as the rate of production or operation of a defined process over a stated period of time. The rates can be expressed in terms of units of products, batches produced, or any other meaningful measurements. In the case of a Card Reader, throughput means how many transactions the Reader can perform in a minute or in an hour or in a day. Throughput is generally measured in terms of 'Benchmark'. A 'Benchmark' is a reference point by which something can be measured. Benchmark can be a set of performance criteria that a product is expected to meet or a standard product that can be used for comparing other products of the same product line.

3.2.1.3 Reliability Reliability is a measure of how much % you can rely upon the proper functioning of the system or what is the % susceptibility of the system to failures.

Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR) are the terms used in defining system reliability. MTBF gives the frequency of failures in hours/weeks/months. MTTR specifies how long the system is allowed to be out of order following a failure. For an embedded system with critical application need, it should be of the order of minutes.

3.2.1.4 Maintainability Maintainability deals with support and maintenance to the end user or client in case of technical issues and product failures or on the basis of a routine system checkup. Reliability and maintainability are considered as two complementary disciplines. A more reliable system means a system with less corrective maintainability requirements and vice versa. As the reliability of the system increases, the chances of failure and non-functioning also reduces, thereby the need for maintainability is also reduced. Maintainability is closely related to the system availability. Maintainability can be broadly classified into two categories, namely, 'Scheduled or Periodic Maintenance (preventive maintenance)' and 'Maintenance to unexpected failures (corrective maintenance)'. Some embedded products may use consumable components or may contain components which are subject to wear and tear and they should be replaced on a periodic basis. The period may be based on the total hours of the system usage or the total output the system delivered. A printer is a typical example for illustrating the two types of maintainability. An inkjet printer uses ink cartridges, which are consumable components and as per the printer manufacturer the end user should replace the cartridge after each '*n*' number of printouts to get quality prints. This is an example for '**Scheduled or Periodic maintenance**'. If the paper feeding part of the printer fails the printer fails to print and it requires immediate repairs to rectify this problem. This is an example of '**Maintenance to unexpected failure**'. In both of the maintenances (scheduled and repair), the printer needs to be brought offline and during this time it will not be available for the user. Hence it is obvious that maintainability is simply an indication of the availability of the product for use. In any embedded system design, the ideal value for availability is expressed as

$$A_i = \text{MTBF}/(\text{MTBF} + \text{MTTR})$$

where A_i = Availability in the ideal condition, MTBF = Mean Time Between Failures, and MTTR = Mean Time To Repair

3.2.1.5 Security Confidentiality, 'Integrity', and 'Availability' (The term 'Availability' mentioned here is not related to the term 'Availability' mentioned under the 'Maintainability' section) are the three major measures of information security. Confidentiality deals with the protection of data and application from unauthorised disclosure. Integrity deals with the protection of data and application from unauthorised modification. Availability deals with protection of data and application from unauthorized users. A very good example of the 'Security' aspect in an embedded product is a Personal Digital Assistant (PDA). The PDA can be either a shared resource (e.g. PDAs used in LAB setups) or an individual one. If it is a shared one there should be some mechanism in the form of a user name and password to access into a particular person's profile–This is an example of 'Availability'. Also all data and applications present in the PDA need not be accessible to all users. Some of them are specifically accessible to administrators only. For achieving this, Administrator and user levels of security should be implemented –An example of Confidentiality. Some data present in the PDA may be visible to all users but there may not be necessary permissions to alter the data by the users. That is Read Only access is allocated to all users–An example of Integrity.

3.2.1.6 Safety 'Safety' and 'Security' are two confusing terms. Sometimes you may feel both of them as a single attribute. But they represent two unique aspects in quality attributes. Safety deals with the possible damages that can happen to the operators, public and the environment due to the breakdown of an embedded system or due to the emission of radioactive or hazardous materials from the embedded products. The breakdown of an embedded system may occur due to a hardware failure or a firmware failure. Safety analysis is a must in product engineering to evaluate the anticipated damages and determine the best course of action to bring down the consequences of the damages to an acceptable level. As stated before, some of the safety threats are sudden (like product breakdown) and some of them are gradual (like hazardous emissions from the product).

3.2.2 Non-Operational Quality Attributes

The quality attributes that needs to be addressed for the product 'not' on the basis of operational aspects are grouped under this category. The important quality attributes coming under this category are listed below.

1. Testability & Debug-ability
2. Evolvability
3. Portability
4. Time to prototype and market
5. Per unit and total cost.

3.2.2.1 Testability & Debug-ability Testability deals with how easily one can test his/her design, application and by which means he/she can test it. For an embedded product, testability is applicable to both the embedded hardware and firmware. Embedded hardware testing ensures that the peripherals and the total hardware functions in the desired manner, whereas firmware testing ensures that the firmware is functioning in the expected way. Debug-ability is a means of debugging the product as such for figuring out the probable sources that create unexpected behaviour in the total system. Debug-ability

has two aspects in the embedded system development context, namely, hardware level debugging and firmware level debugging. Hardware debugging is used for figuring out the issues created by hardware problems whereas firmware debugging is employed to figure out the probable errors that appear as a result of flaws in the firmware.

3.2.2.2 Evolvability Evolvability is a term which is closely related to Biology. Evolvability is referred as the non-heritable variation. For an embedded system, the quality attribute 'Evolvability' refers to the ease with which the embedded product (including firmware and hardware) can be modified to take advantage of new firmware or hardware technologies.

3.2.2.3 Portability Portability is a measure of 'system independence'. An embedded product is said to be portable if the product is capable of functioning 'as such' in various environments, target processors/controllers and embedded operating systems. The ease with which an embedded product can be ported on to a new platform is a direct measure of the re-work required. A standard embedded product should always be flexible and portable. In embedded products, the term 'porting' represents the migration of the embedded firmware written for one target processor (e.g. Intel x86) to a different target processor (say Hitachi SH3 processor). If the firmware is written in a high level language like 'C' with little target processor-specific functions (operating system extensions or compiler specific utilities), it is very easy to port the firmware for the new processor by replacing those 'target processor-specific functions' with the ones for the new target processor and re-compiling the program for the new target processor-specific settings. Re-compiling the program for the new target processor generates the new target processor-specific machine codes. If the firmware is written in Assembly Language for a particular family of processor (say x86 family), it will be very difficult to translate the assembly language instructions to the new target processor specific language and so the portability is poor.

If you look into various programming languages for application development for desktop applications, you will see that certain applications developed on certain languages run only on specific operating systems and some of them run independent of the desktop operating systems. For example, applications developed using Microsoft technologies (e.g. Microsoft Visual C++ using Visual studio) is capable of running only on Microsoft platforms and will not function on other operating systems; whereas applications developed using 'Java' from Sun Microsystems works on any operating system that supports Java standards.

3.2.2.4 Time-to-Prototype and Market Time-to-market is the time elapsed between the conceptualisation of a product and the time at which the product is ready for selling (for commercial product) or use (for non-commercial products). The commercial embedded product market is highly competitive and time to market the product is a critical factor in the success of a commercial embedded product. There may be multiple players in the embedded industry who develop products of the same category (like mobile phone, portable media players, etc.). If you come up with a new design and if it takes long time to develop and market it, the competitor product may take advantage of it with their product. Also, embedded technology is one where rapid technology change is happening. If you start your design by making use of a new technology and if it takes long time to develop and market the product, by the time you market the product, the technology might have superseded with a new technology. Product prototyping helps a lot in reducing time-to-market. Whenever you have a product idea, you may not be certain about the feasibility of the idea. Prototyping is an informal kind of rapid product development in which the important features of the product under consideration are developed. The time to prototype is also another critical factor. If the prototype is developed faster, the actual estimated development time

can be brought down significantly. In order to shorten the time to prototype, make use of all possible options like the use of off-the-shelf components, re-usable assets, etc.

3.2.2.5 Per Unit Cost and Revenue Cost is a factor which is closely monitored by both end user (those who buy the product) and product manufacturer (those who build the product). Cost is a highly sensitive factor for commercial products. Any failure to position the cost of a commercial product at a nominal rate, may lead to the failure of the product in the market. Proper market study and cost benefit analysis should be carried out before taking a decision on the per-unit cost of the embedded product. From a designer/product development company perspective the ultimate aim of a product is to generate marginal profit. So the budget and total system cost should be properly balanced to provide a marginal profit. Every embedded product has a product life cycle which starts with the design and development phase. The product idea generation, prototyping, Roadmap definition, actual product design and development are the activities carried out during this phase. During the design and development phase there is only investment and no returns. Once the product is ready to sell, it is introduced to the market. This stage is known as the Product Introduction stage. During the initial period the sales and revenue will be low. There won't be much competition and the product sales and revenue increases with time. In the growth phase, the product grabs high market share. During the maturity phase, the growth and sales will be steady and the revenue reaches at its peak. The Product Retirement/Decline phase starts with the drop in sales volume, market share and revenue. The decline happens due to various reasons like competition from similar product with enhanced features or technology changes, etc. At some point of the decline stage, the manufacturer announces discontinuing of the product. The different stages of the embedded products life cycle–revenue, unit cost and profit in each stage–are represented in the following Product Life-cycle graph.

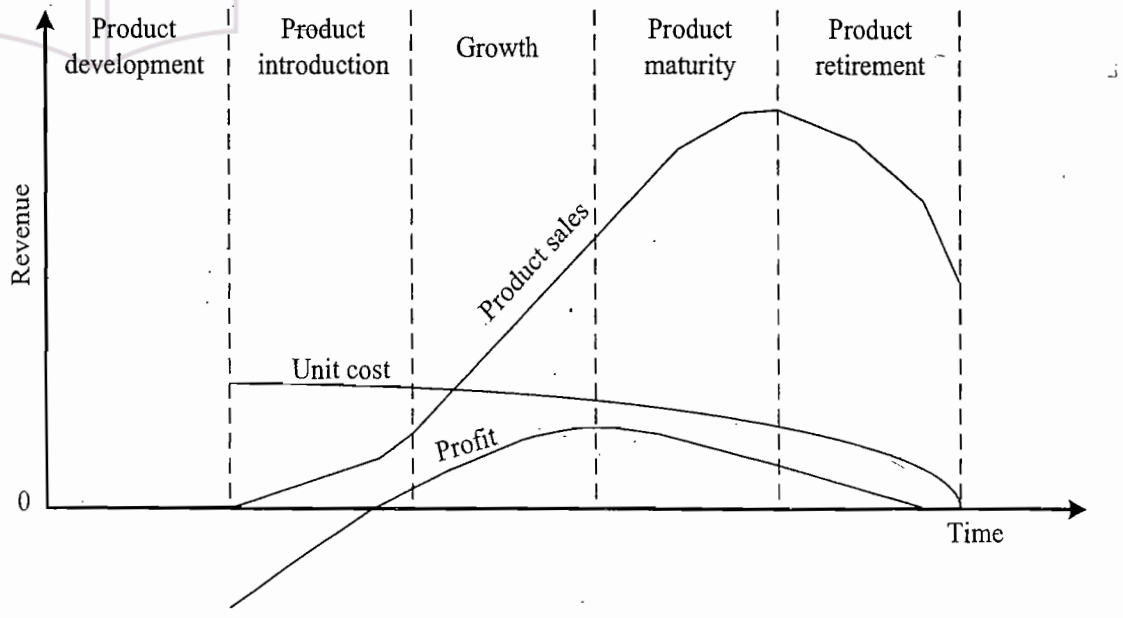

Fig. 3.1 **Product life cycle (PLC) curve**

From the graph, it is clear that the total revenue increases from the product introduction stage to the product maturity stage. The revenue peaks at the maturity stage and starts falling in the decline/retirement stage. The unit cost is very high during the introductory stage (a typical example is cell phone; if

you buy a new model of cell phone during its launch time, the price will be high and you will get the same model with a very reduced price after three or four months of its launching). The profit increases with increase in sales and attains a steady value and then falls with a dip in sales. You can see a negative value for profit during the initial period. It is because during the product development phase there is only investment and no returns. Profit occurs only when the total returns exceed the investment and operating cost.

Summary

- ✓ There exists a set of characteristics which are unique to each embedded system.
- ✓ Embedded systems are application and domain specific.
- ✓ Quality attributes of a system represents the non-functional requirements that need to be documented properly in any system design.
- ✓ The operational quality attributes of an embedded system refers to the non-functional requirements that needs to be considered for the operational mode of the system. Response, Throughput, Reliability, Maintainability, Security, Safety, etc. are examples of operational quality attributes.
- ✓ The non-operational quality attributes of an embedded system refers to the non-functional requirements that needs to be considered for the non-operational mode of the system. Testability, debug-ability, evolvability, portability, time-to-prototype and market, per unit cost and revenue, etc. are examples of non-operational quality attributes.
- ✓ The product life cycle curve (PLC) is the graphical representation of the unit cost, product sales and profit with respect to the various life cycle stages of the product starting from conception to disposal.
- ✓ For a commercial embedded product, the unit cost is peak at the introductory stage and it falls in the maturity stage.
- ✓ The revenue of a commercial embedded product is at the peak during the maturity stage.

Keywords

Quality attributes	:	The non-functional requirements that need to be addressed in any system design
Reactive system	:	An embedded system which produces changes in output in response to the changes in input
Real-Time system	:	A system which adheres to strict timing behaviour and responds to requests in a known amount of time
Response	:	It is a measure of quickness of the system
Throughput	:	The rate of production or operation of a defined process over a stated period of time
Reliability	:	It is a measure of how much % one can rely upon the proper functioning of a system
MTBF	:	Mean Time Between Failures–The frequency of failures in hours/weeks/months
MTTR	:	Mean Time To Repair–Specifies how long the system is allowed to be out of order following a failure
Time-to-prototype	:	A measure of the time required to prototype a design
Product life-cycle (PLC)	:	The representation of the different stages of a product from its conception to disposal
Product life cycle curve	:	The graphical representation of the unit cost, product sales and profit with respect to the various life cycle stages of the product starting from conception to disposal

Objective Questions

1. Embedded systems are application and domain specific. State True or False
   (a) True (b) False
2. Which of the following is true about Embedded Systems?
   (a) Reactive and Real Time (b) Distributed (c) Operates in harsh environment
   (d) All of these (e) None of these
3. Which of the following is a distributed embedded system?
   (a) Cell phone (b) Notebook Computer (c) SCADA system (d) All of these
   (e) None of these
4. Quality attributes of an embedded system are
   (a) Functional requirements (b) Non-functional requirements
   (c) Both (d) None of these
5. Response is a measure of
   (a) Quickness of the system (b) How fast the system tracks changes in Input
   (c) Both (d) None of these
6. Throughput of an embedded system is a measure of
   (a) The efficiency of the system (b) The output over a stated period of time
   (c) Both (d) None of these
7. Benchmark is
   (a) A reference point (b) A set of performance criteria
   (c) (a) or (b) (d) None of these
8. Mean Time Between Failures (MTBF) and Mean Time To Repair (MTTR) defines the reliability of an embedded system. State True or False
   (a) True (b) False
9. MTBF gives the frequency of failures of an embedded system. State True or False
   (a) True (b) False
10. Which of the following is true about the quality attribute 'maintainability'?
    (a) The corrective maintainability requirement for a highly reliable embedded system is very less
    (b) Availability of an embedded system is directly related to the maintainability of the system
    (c) Both of these
    (d) None of these
11. The Mean Time Between Failure (MTBF) for an embedded product is very high. This means:
    (a) The product is highly reliable
    (b) The availability of the product is very high
    (c) The preventive maintenance requirement for the product is very less
    (d) All of these
    (e) None of these
12. The Mean Time Between Failure (MTBF) of an embedded product is 4 months and the Mean Time To Repair (MTTR) of the product is 2 weeks. What is the availability of the product?
    (a) 100% (b) 50% (c) 89% (d) 10%
13. Which of the following are the three measures of information security in embedded systems?
    (a) Confidentiality, secrecy, integrity (b) Confidentiality, integrity, availability
    (c) Confidentiality, transparency, availability (d) Integrity, transparency, availability

14. You are working on a mission critical embedded system development project for a client and the client and your company has signed a Non Disclosure Agreement (NDA) on the disclosure of the project-related information. You share the details of the project you are working with your friend. Which aspect of Information security you are violating here?
    (a) Integrity (b) Confidentiality (c) Availability (d) None of these
15. Which of the following is an example of 'gradual' safety threat from an embedded system?
    (a) Product blast due to overheating of the battery (b) UV emission from the embedded product
    (c) Both of these (d) None of these
16. Non operational quality attributes are
    (a) Non-functional requirements (b) Functional requirements
    (c) Quality attributes for an offline product (d) (a) and (c)
    (e) None of these
17. Which of the following is (are) an operational quality attribute?
    (a) Testability (b) Safety (c) Debug-ability (d) Portability
    (e) All of these
18. Which of the following is (are) non-operational quality attribute?
    (a) Reliability (b) Safety (c) Maintainability (d) Portability
    (e) All of these (f) None of these
19. In the Information security context, Confidentiality deals with the protection of data and application from unauthorised disclosure. State True or False
    (a) True (b) False
20. What are the two different aspects of debug-ability in the embedded system development context?
    (a) Hardware & Firmware debug-ability (b) Firmware & Software debug-ability
    (c) None of these
21. For an embedded system, the quality attribute 'Evolvability' refers to
    (a) The upgradability of the product (b) The modifiability of the product
    (c) Both of these (d) None of these
22. Portability is a measure of 'system independence'. State True or False
    (a) True (b) False
23. For a commercial embedded product the *unit cost* is high during
    (a) Product launching (b) Product maturity
    (c) Product growth (d) Product discontinuing
24. For a commercial embedded product the sales volume is high during
    (a) Product launching (b) Product maturity
    (c) Product growth (d) Product discontinuing

Review Questions

1. Explain the different characteristics of embedded systems in detail.
2. Explain quality attribute in the embedded system development context? What are the different Quality attributes to be considered in an embedded system design.
3. What is operational quality attribute? Explain the important operational quality attributes to be considered in any embedded system design.
4. What is non-operational quality attribute? Explain the important non-operational quality attributes to be considered in any embedded system design.
5. Explain the quality attribute *Response* in the embedded system design context.
6. Explain the quality attribute *Throughput* in the embedded system design context.

7. Explain the quality attribute *Reliability* in the embedded system design context.
8. Explain the quality attribute *Maintainability* in the embedded system design context.
9. The availability of an embedded product is 90%. The Mean Time Between Failure (MTBF) of the product is 30 days. What is the Mean Time To Repair (MTTR) in days/hours for the product?
10. Explain the quality attribute *Information Security* in the embedded system design context.
11. Explain the quality attribute *Safety* in the embedded system design context.
12. Explain the significance of the quality attributes *Testability and Debug-ability* in the embedded system design context.
13. Explain the quality attribute *Portability* in the embedded system design context.
14. Explain *Time-to-market*? What is its significance in product development?
15. Explain *Time-to-prototype*? What is its significance in product development?
16. Explain the *Product Life-cycle* curve of an embedded product development.