UNIT-3

EMBEDDED FIRMWARE DEVELPOMENT ENVIRONMENT

PART – A(2 marks)

1. *What is EDLC?*

Embedded Product Development Life Cycle (EDLC) is an analysis – Design – Implementation based standard problem solving approach for embedded product development. EDLC defines the interaction and activities among various groups of a product development sector including project management, system design and development, system testing, release management and quality assurance.

2. *What is Model?*

The Life cycle of a product development is commonly referred as Models and a Model defines the various phases involved in a product's life cycle. The Embedded product life cycle model contains the phases: Need, Conceptualization, Analysis, Design, Development and testing, Deployment, Support, Upgrades and Retirement/Disposal.

3. *Define conceptualization phase.*

Conceptualization phase is the phase dealing with product concept development. It includes activities like product feasibility analysis, cost benefit analysis, product scoping and planning for next phases.

4. *Name three categories of product development.*

- The 3 categories are
- New or custom product development
- Product Re-engineering
- Product maintenance

5. *What are the models used in EDLC?*

The models are:

- The linear or waterfall models
- Iterative or incremental or fountain models
- Prototyping or Evolutionary Models
- Spiral Models

6. *Define Spiral Model.*

Spiral Model is the EDLC model combining linear and prototyping model to give the best possible risk minimization in product development.

7. *Define data flow model.*

Data flow is a type of process network model. In data flow, a program is specified by a directed graph. The nodes of the graph represent computational functions that map input data into output data. Data is represented by a circle and data flow is represented using arrows.

8. *Define deployment phase.*

The deployment phase deals with the launching of the product. Product deployment notification, training plan execution, product installation, product post implementation review, etc., are the activities performed during deployment phase.

9. *Define product design space and development phase.*

Design phase – It deals with the implementation aspects of the required functionalities for the product

Development Phase – It transforms the design into a realizable product. The detailed specifications generated during the design phase are translated into hardware and firmware during the development phase.

10. What are the differences between data flow model and state machine model?

Both data flow and finite state machine are models of computation. The data flow model of computation is used in signal processing design and modeling of DSP algorithms. On the other hand FSMs have been developed to solve a different class of problems, namely sequential control. FSMs are an appropriate modeling approach for control-dominant applications. Mixing data flow with FSMs is a good solution for representing a system, which requires both signal processing and control. This integrated is very useful to eradicate the drawbacks.

UNIT-3

PART-B(16 marks)

1. Discuss the Objective of EDLC?

EMBEDDED PRODUCT DEVELOPMENT LIFE CYCLE (EDLC)

EDLC is an Analysis – Design – Implementation based problem solving approach for the product development.

- Analysis – what product needs to be developed?
- Design – Good approach for building it
- Implementation – To develop it

ESSENTIALITY OF EDLC:

The necessity and importance of EDLC is to understand the scope and complexity of the work involved in any embedded product development. EDLC defines the interaction and activities among various groups of a product development sector including project management, system design & development, system testing, release management & quality assurance. EDLC standards are needed to design for the product development, which provides the uniformity in development approaches.

OBJECTIVES OF EDLC:

- The aim of any product development is the Marginal benefit. Generally, marginal benefit is expressed as "Return on Investment" (ROI). The investment for a product development includes initial investment, manpower investment & infrastructure investment etc.
- A developed product needs to be acceptable by the end user that it has to meet the requirements of the end user in terms of quality, reliability & functionality.
- EDLC helps in ensuring all these requirements by following 3 objectives:
- Ensure that high quality products are delivered to end user.
- Risk minimization & defect prevention in product development through project management
- Maximize the productivity

Ensuring High quality for products:

- The primary definition of quality in any embedded product development is return on investment achieved by the product. In order to survive in market, quality is the most important factor to be taken care of while developing the product.
- Qualitative attributes depends on the budget of the product, because budget allocation is very important.
- Budget allocation might have done after analyzing the market, trends & requirements of product, competition etc.
- EDLC must ensure that the development of the product has taken account of all the quantitative & qualitative attributes of the embedded system.

Risk minimization & defect prevention through management:

- The project management is essential in product development and it needed more significance.
- Project management adds an extra cost on the budget but it is essential for ensuring the development process is going in the right direction and the schedules of the development activity are meeting.

- Projects which are complex and requires timeliness should have a dedicated & skilled project management part & hence they are said to be “Highly” bounded to project management.
- Project management is essential for predictability, coordination and risk minimization
- The time frame may be expressed in number of person days (PDS)
- Predictability – Analyze the time to finish the product.

Coordination – Developers are needed to do the job

Risk management – 1. Backup of resources to overcome critical situation
2. Ensuring defective product is not developed

Increased Productivity:

Productivity is a measure of efficiency as well as Return on Investment. Different ways to improve the productivity are:

- Saving the manpower effort will definitely result increased productivity
- Use of automated tools wherever is required
- Work which has been done for the previous product can be used, when there is a presence of similarities between the previous and present product. This is called as “Re-usable effort”.
- Usage of resource persons with specific set of skills, which exactly matches the requirements of the product. This reduces the time in training the resource. Example: resource with expertise in zigbee wireless technology for developing a wireless interface for the product. This kind of resource does not need training.

2. What are the different phases of EDLC?

- A life cycle of a product development is commonly referred as the “Model”
- A sample model contains 5 Phases
    - ➢ Requirement Analysis
    - ➢ Design
    - ➢ Development and test
    - ➢ Deployment
    - ➢ Maintenance
- Complexity depends on number of phases involved in EDLC

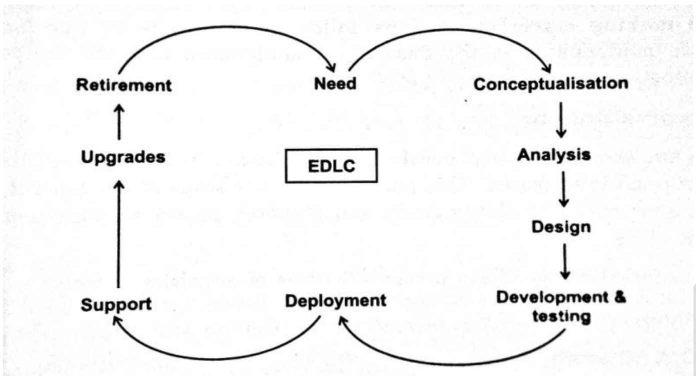


Fig. Classical Embedded product Development Life Cycle

<u>Need:</u>

Any embedded product may evolve as an output of a need. Need may come from an individual or from public or from company (generally from an end user or client). Need initiate the concept proposal. The human resource management and funding agency reviews the concept proposal and provides the approval. Then it goes to a product development team. The product development need can be visualized in any one of the following three needs.

a) ***<u>New or Custom Product Development:</u>***

The need for a new product which does not exist in the market or a product which acts as a competitor to an existing product in the current market will lead to the development of completely new product. Example: Various manufacturers act as competitors in developing the mobile phones.

b) ***<u>Product Re-engineering:</u>***

The process of making changes in an existing product design and launching it as a new version is known as re-engineering a product. It is termed as product upgrade. Re-engineering an existing product comes as a result of the following needs.

- Change in business requirements
- User interface enhancements
- Technology upgrades

c) ***<u>Product Maintenance</u>****:*

The technical support provides to the end user for an existing product in the market is "Product Maintenance need". It has two categories 1. Corrective Maintenance deals with making corrective actions following a failure or non-functioning. 2. Preventive maintenance is the scheduled maintenance to avoid the failure or non-functioning of the product.

<u>Conceptualization:</u>

This is the product concept development phase and it begins immediately after a concept proposal is approved. This phase defines the scope of the concept, performs cost benefit analysis and feasibility study and prepares project management and risk management plans.

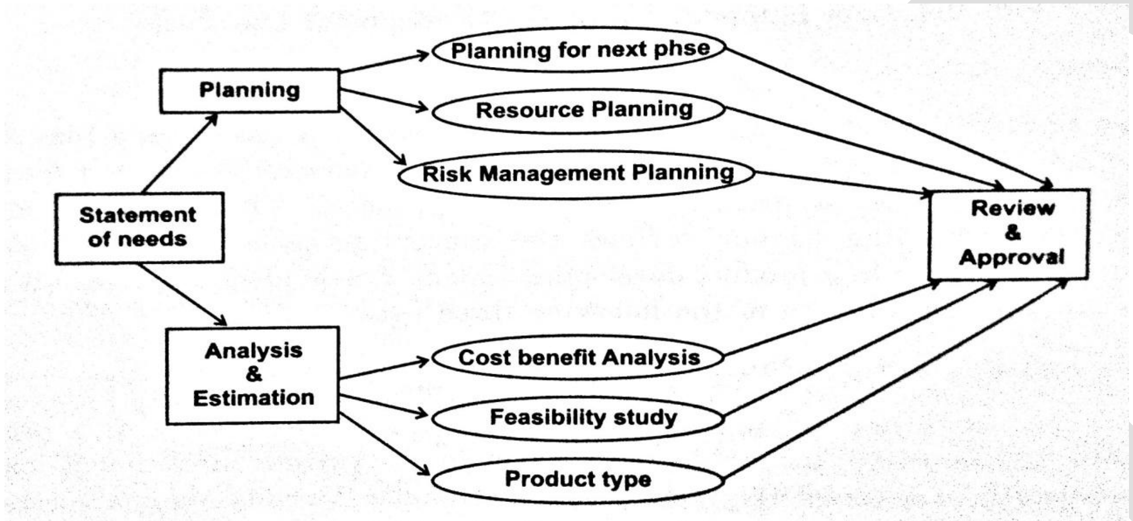


Fig. Activities in Conceptualization phase

The conceptualization phase involves 2 types of activities namely,

1. Planning Activity
2. Analysis and study Activity

- ***Feasibility study*** – examines the need for the product and analyses the technical as well as financial feasibility of the product under construction.
- ***Cost Benefit Analysis (CBA)*** – It examines and total up the equivalent money value of the benefits and costs of the product under consideration and provide an idea about the worthwhile of the product. Some principles of CBA are:

1. Common unit of measurement – Marginal profit is expressed in terms of common currency. Example: Indian Rupee or US dollar
2. Market choice based benefit measurement – value of money
3. Target end users.

- ***Product Scope*** – deals with what is in scope and what is not in the scope.
- ***Planning Activities*** – It contains plans for product development. It also contains recommendation such as
  - The product is feasible and proceeds to the next phase of the product life cycle.
  - If the Product is not feasible, scrap the product.

<u>Analysis:</u>

Requirement analysis phase provides emphasis on determining 'what functions must be performed by the product' rather than how to perform those functions.

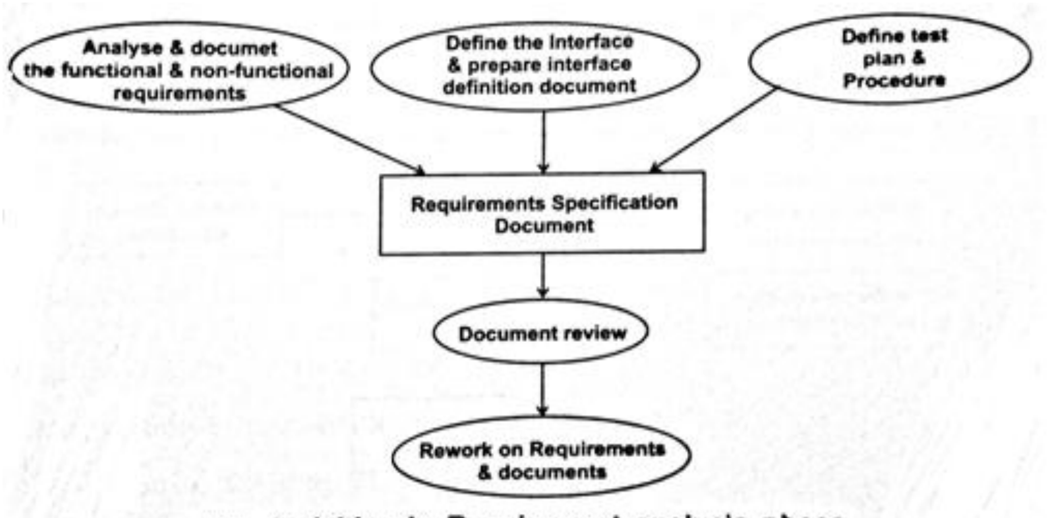

Analyse & documet the functional & non-functional requirements
Define the Interface & prepare interface definition document
Define test plan & Procedure
Requirements Specification Document
Document review
Rework on Requirements & documents

Fig. Activities in Requirement analysis phase

- ***Analysis and documentation*** – this activity consolidates the business needs of the product under development and analyses the purpose of the product. The various requirement are:
  - Functional Capabilities
  - Operational and non-operational quality parameters
  - Data Requirements
  - User manuals
  - Operational requirements
  - Maintenance requirements
- ***Interface Definition And Documentation*** – This activity should clearly analyze on the physical interface as well as data exchange through the interfaces and should document it. It is for proper flow of data throughout the life cycle.
- ***Defining test plan and Procedures*** – Master test plan consist of various type of testing in a product development. They are: Unit testing, Integration testing and System testing.

<u>Design:</u>

This deals with the entire design of the product taking the requirements into consideration and focuses on how the functionalities can be delivered. Preliminary design consists of the list are,

- Inputs and Outputs are defined here

- Product will look like a black box
- Preliminary design document is sent for review to the client.
- After the client review, detailed design architecture is generated.
- This design also needs to be reviewed and get approved by the end user.

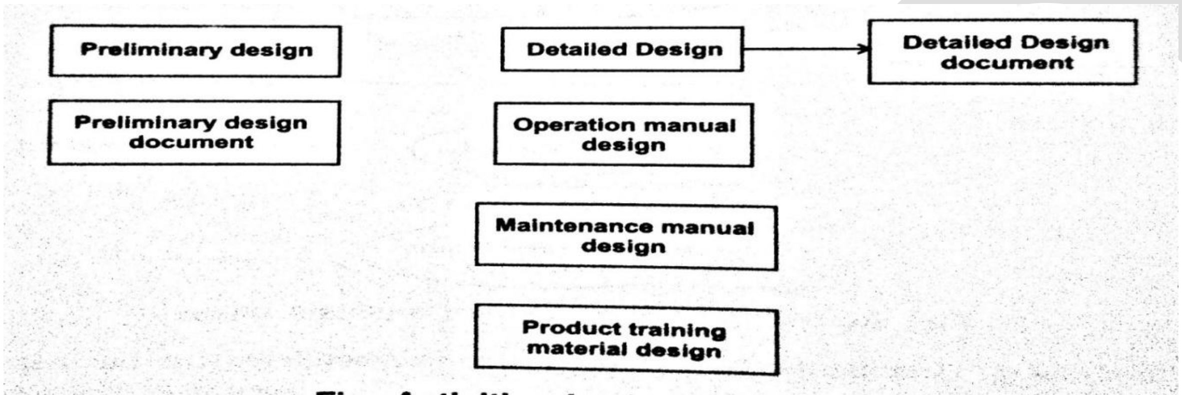

Preliminary design
Preliminary design document
Detailed Design
Detailed Design document
Operation manual design
Maintenance manual design
Product training material design

Fig. Activities in design phase

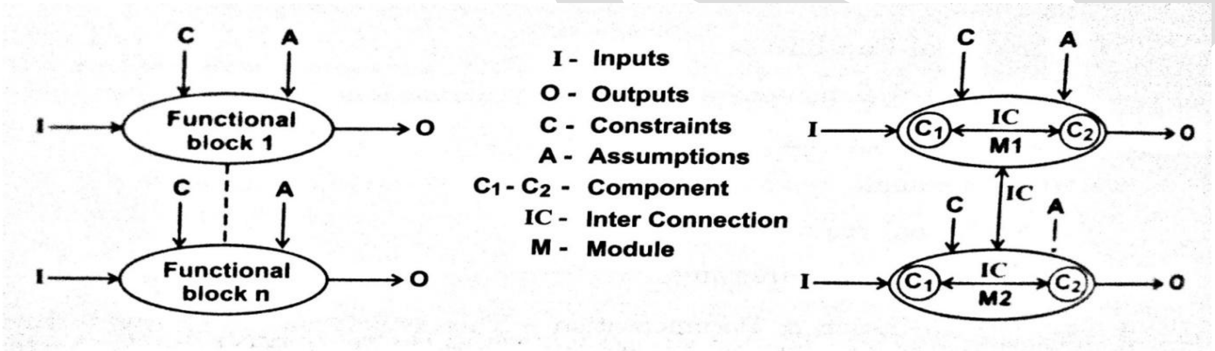


Fig. Preliminary design illustration Fig. Detailed design illustration

Development and testing:

- Development phase transforms the design into realizable product.
- Design is transformed into hardware and firmware.
- Look and feel of the device is very important.

Testing phase can be divided into,

➢ ***Unit Testing*** – independent testing of Hardware and firmware

➢ ***Integration Testing*** - testing after integrating Hardware and firmware

➢ ***System Testing*** – Testing of whole system on functionality and non-functionality basis

➢ ***User acceptance testing*** – testing of the product against the criteria mentioned by the end user or client.

➢ ***Test reports***.

Deployment:

A process of launching fully functional model into the market is called deployment. It is also known as first customer shipping. The essential tasks performed during the deployment phase are,

- ➢ Notification of product deployment
- ➢ Execution of training plan
- ➢ Product installation
- ➢ Product Post – Implementation Review

<u>Support:</u>

This deals with the operation and maintenance of the product. Support should be providing to the end user to fix the bugs of the product. The various activities involved in the support phase are:

- ➢ Setup a dedicated support wing. Example: Customer Care.
- ➢ Identification of bugs and Areas of improvement.

<u>Upgrades:</u>

It deals with

- ➢ Releasing of new version for the product which is already exist in the market.
- ➢ Releasing of major bug fixes. (Firmware up gradation)

<u>Retirement/Disposal:</u>

Everything in the world changes, the technology you feel as the most advanced and best today may not be the same tomorrow. Due to this reason, the product cannot sustain for a long time in the market. It has to be disposed on right time before it causes the loss. The disposition of the product is essential due to the following reason.

1. Rapid technological advancements.
2. Increased user needs.

3. Mention and explain the approaches of EDLC?

MODELLING OF EDLC: (EDLC APPROACHES)

The various approaches in modelling the EDLC are:

- **Linear or water fall model:**
  - In this model, each phase of EDLC is executed in sequence and the flow is unidirectional with output of one phase serving as the input to the next phase. The feedback of each pulse is available locally and only after they are executed.

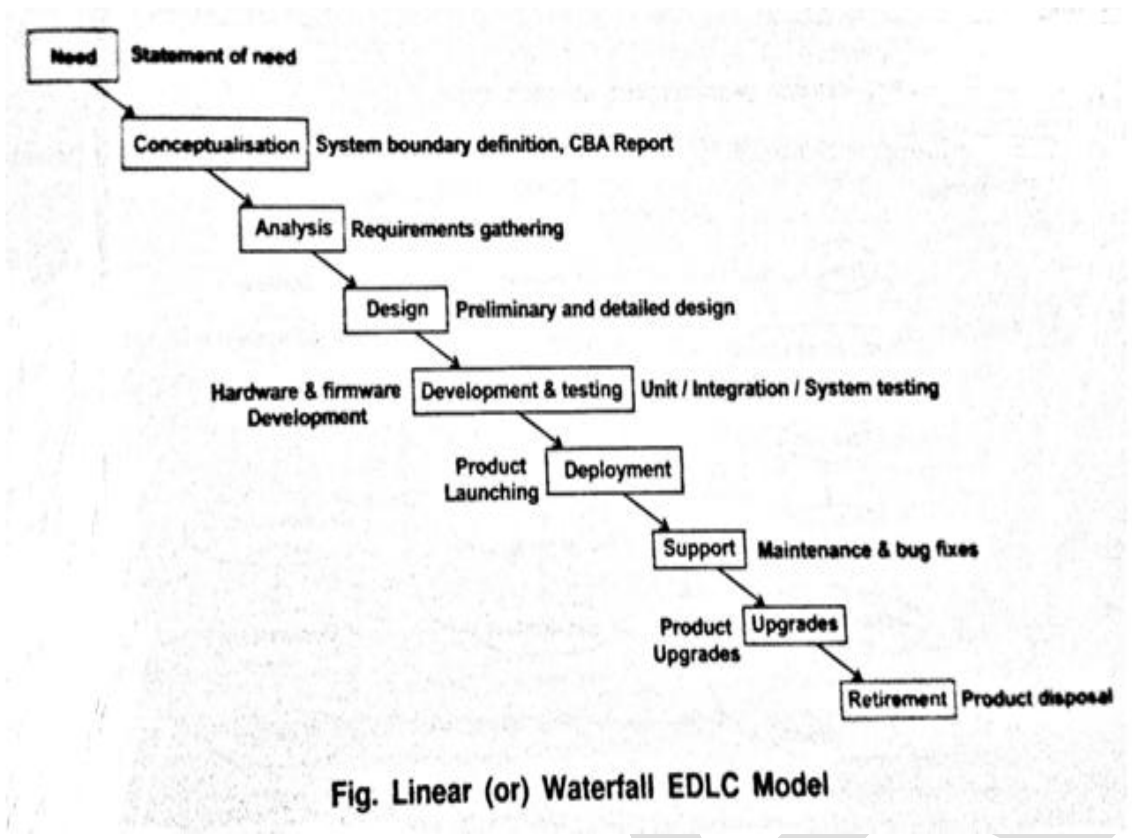


Fig. Linear (or) Waterfall EDLC Model

- Review mechanisms are employed to ensure that the process flow in right direction.
- Bugs are not fixed immediately and they are postponed to support phase.

Advantages:

- Rich documentation
- Easy project management
- Good control over cost & schedule

Drawbacks:

- Analysis can be done without any design
- Risk analysis is performed only once throughout the development
- Bugs are fixed only at support phase

- **Iterative / Incremental / Fountain EDLC Model:**
- The iterative model can be viewed as a cascaded series of linear models.
- Do some analysis, follow some design, the some implementation, evaluate it and based on short comings, cycle back through and conduct more analysis, opt for new design and implementation.
- Repeat the cycles until the requirements are met.

- The incremental model is a superset of iterative model where the requirements are known at the beginning.

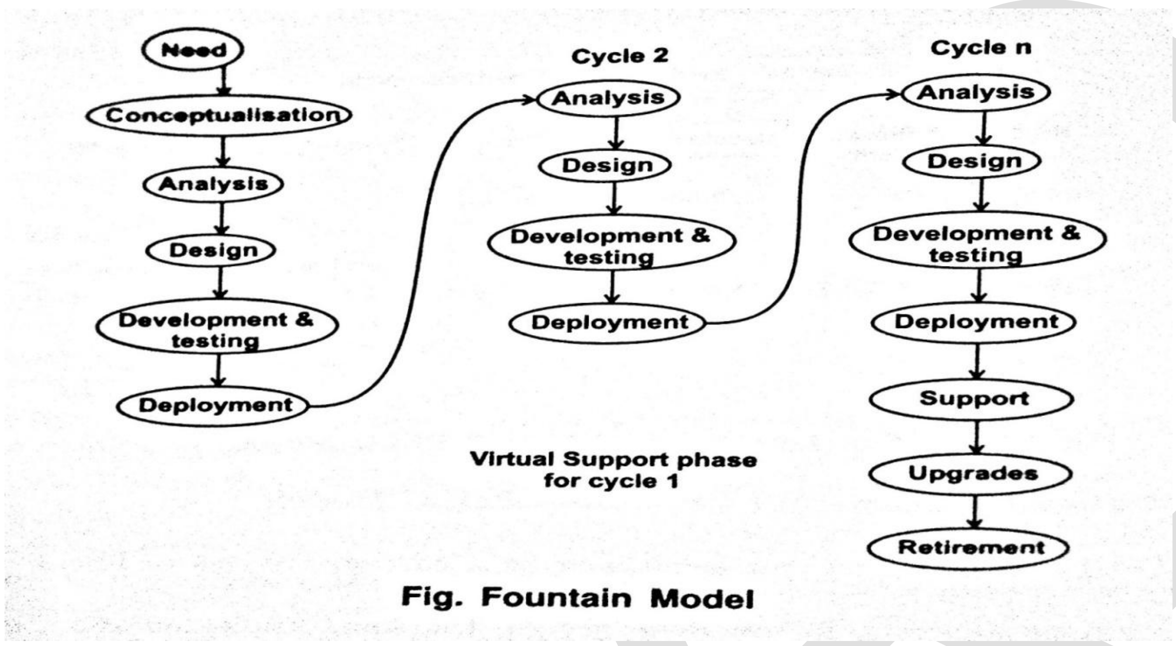


Fig: Fountain model

Advantages:

▪ Feedback is at each function. So each cycle can act as a maintenance phase for previous cycle

▪ Risk is spread across each individual cycle and can be minimized easily

▪ Project management as well as testing is much simpler to the linear model

▪ Product development can be stopped at any stage

Drawbacks:

▪ Extensive review requirement at each cycle

▪ Training is required for each new deployment at the end of each development cycle.

Prototyping / Evolutionary Model:

▪ This model is similar to the iterative model and the product is developed in multiple cycles

▪ The only difference is the model produces more refined prototype of the product at each cycle instead of just adding the functionality at each cycle like in interval model

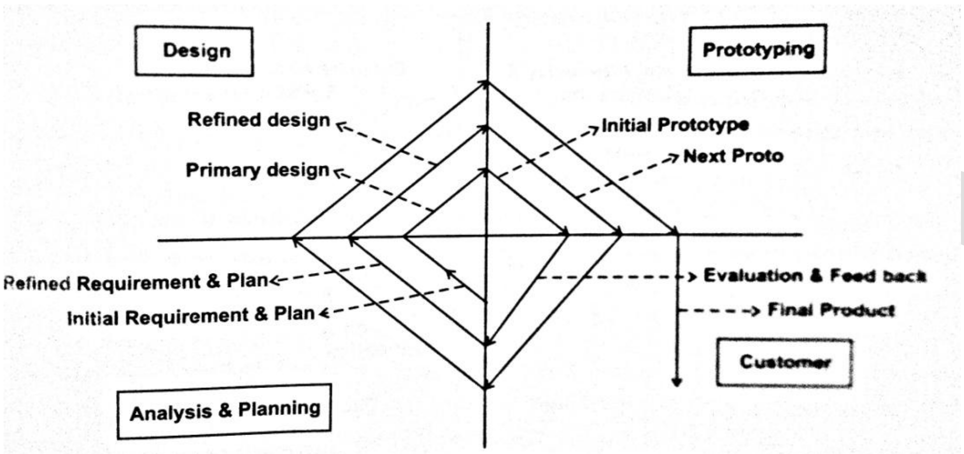


Fig. Prototyping / Evolutionary Model

- The short comings of the protomodel after each cycle are evaluated and it is fixed in the next cycle
- After the initial requirement analysis, design is made and development process is started. Then this prototype is sent to the customer for evaluation. The customer provides the feedback to the developer for further improvement.
- Then the developer repeat the process with some additional features and finally delivered to the outside world

Advantages:

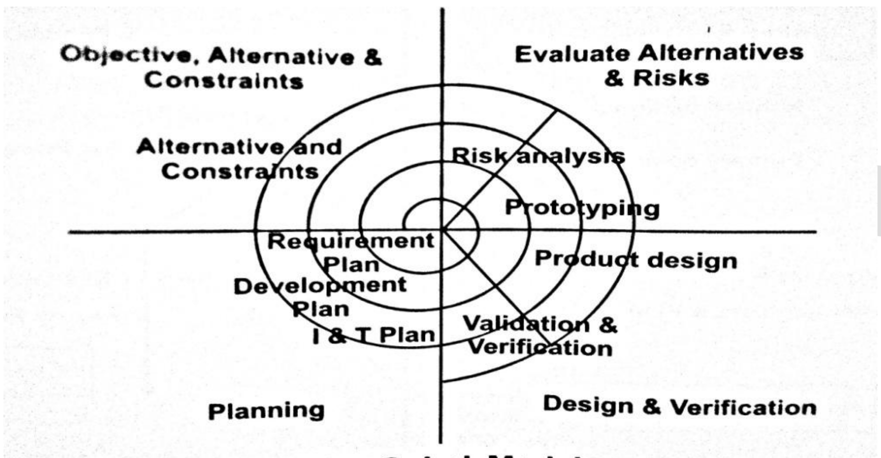
- Feedback after each implementation and fire tuning is also possible
- By using the prototype model, the risk is spread across each proto development cycle & it is under control

Drawbacks:

- Deviations from expected cost and schedule due to requirements refinement.
- Increased project management
- Minimal documentation on each prototype may create problems in backward prototype traceability
- Increased configuration management activities

Spiral Model:

- Spiral model is best suited for the development of complex embedded products & situations where the requirements are changing from customer side
- Risk evaluation in each stage helps in reducing risk


Objective, Alternative & Constraints
Evaluate Alternatives & Risks
Alternative and Constraints
Risk analysis
Prototyping
Requirement Plan
Product design
Development Plan
I & T Plan
Validation & Verification
Planning
Design & Verification
Fig. Spiral Model

- Spiral model is best suited for the development of complex embedded products & situations where the requirements are changing from customer side
- Risk evaluation in each stage helps in reducing risk
- It is the combination of linear & prototype models to give the best possible risk minimized EDLC model. The activities in the spiral model present in the 4 quadrants are:
  - Determine objectives, alternatives, constraints
  - Evaluate alternatives, identity & resolve risks
  - Develop & test
  - Plan

4. What are the fundamental issues in Hardware-Software co-design?

- **Co design**

The meeting of system-level objectives by exploiting the trade-offs between hardware and software in a system through their concurrent design

- **Key concepts**

○ **Concurrent:** hardware and software developed at the same time on parallel paths

○ **Integrated:** interaction between hardware and software developments to produce designs that meet performance criteria and functional specifications

Fundamental issues:

The problem statement is hardware software co-design and its issues. Some of the issues are:

Selecting the model:

- A model is a formal system consisting of objects & composition rules. It is hard to make a decision on which model should be followed in a particular system design
- Models are used for capturing & describing the system characteristics

- Designers switch between a variety of models, because the objective varies with each phase

Selecting the architecture:

The architecture specifies how a system is going to implement in terms of the number & types of different components and the interconnection among them. Some type of architecture falls into application specific architecture class, while others fall into either general purpose architecture class or parallel processing class. The commonly used architectures is system design are:

1. The controller Architecture – implements the finite state machine model using a state register holds the present state and the combinational circuits holds the next state & output
2. The data path architecture – suitable for implementing the data flow graph model. A datapath represents a channel between the input and output. The datapath contains registers, counters, memories & ports. Ports connect the datapath to multiple buses.
3. The finite state machine datapath – this architecture combines the controller architecture with datapath architecture. The controller generates the control input, whereas the datapath processes the data.
4. The complex instruction set computing (CISC) – this architecture uses an instruction set for solving complex operations. The use of a single complex instruction in place of multiple simple instructions greatly reduces the program memory access & program memory size requirement. On the other hand, Reduced Instruction Set Computing (RISC) architecture uses the multiple RISC instructions to perform a complex operation. RISC architecture supports extensive pipelining
5. The Very Long Instruction Word (VLIW) – this architecture implements functional units in the datapath
6. Parallel processing architecture – implements multiple concurrent processing elements and each processing element may associate a datapath containing register & local memory. Single Instruction Multiple Data (SIMD) & Multiple Instruction Multiple Data (MIMD) architectures are examples for parallel processing architecture. SIMD – eg: Reconfigurable process, MIMD – eg: Multiprocessor systems

Selecting the language:

A programming language captures a 'Computational Model' and maps it into architecture. A model can be captured using multiple programming languages like C, C++, C#, Java, etc. for software implementations and languages like VHOL, system C, verilog, etc. for hardware implementations. C++ is a good language for capturing an object oriented model.

Partitioning system requirements into hardware & software:

Various hardware software trade-offs are used for making a decision on the hardware-software portioning.

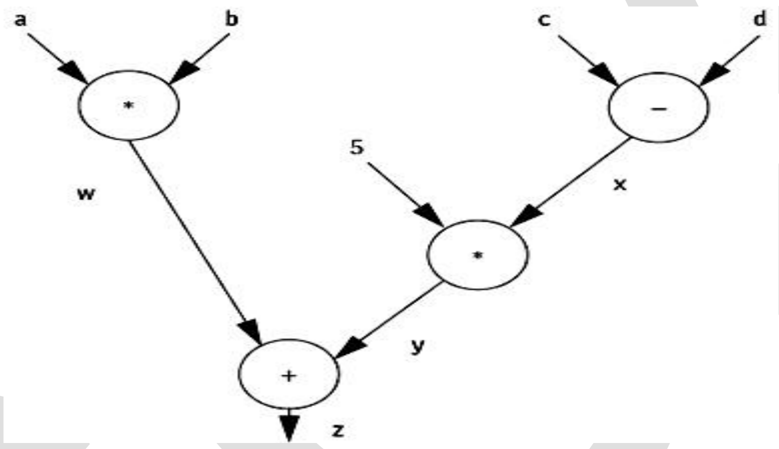
5. Discuss about the various computational models in embedded design.

<u>COMPUTATIONAL MODELS IN EMBEDDED DESIGN</u>

The commonly used computational models in embedded system design are

(i) Data flow graph model(DFG):

This model translates the data processing requirement into a data flow graph. It is a data model. This model emphasis on the data and operations on the data which transforms the Input data to output data. In DFG model, Data is represented by a circle data flow is represented arrows.An inward arrow to the process denotes the output data. Data driven Embedded are modeled using DFG.



A DFG model is said to be either acyclic DFG (does not contain multiple input values & multiple output values) or non-acyclic DFG (output is feedback to the input).

(ii) **State machine model:**

The state machine model explains the system behaviours with states, events, actions and transitions. The representation of a state is current situation. An event is an input to the state. The event acts as stimulus for state transition. The activity evolved in state machine is action. This model is used for event driven embedded systems. Eg: control and industrial applications.

A Finite State Machine (FSM) model is one in which the number of states are finite. Consider the FSM model for automatic seat belt warning system as an example.

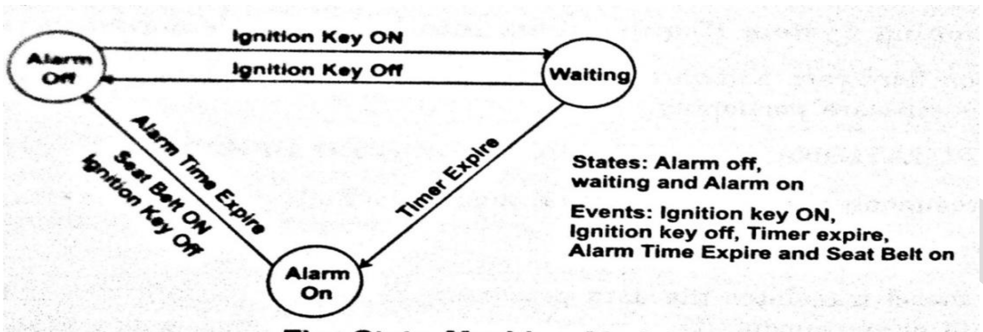

Ignition Key ON
Ignition Key Off
Alarm Off
Waiting
Alarm Time Expire
Seat Belt ON
Ignition Key Off
Timer Expire
Alarm On
States: Alarm off, waiting and Alarm on
Events: Ignition key ON, Ignition key off, Timer expire, Alarm Time Expire and Seat Belt on

Fig. State Machine Model

Motor control software design is another example for state machine model.

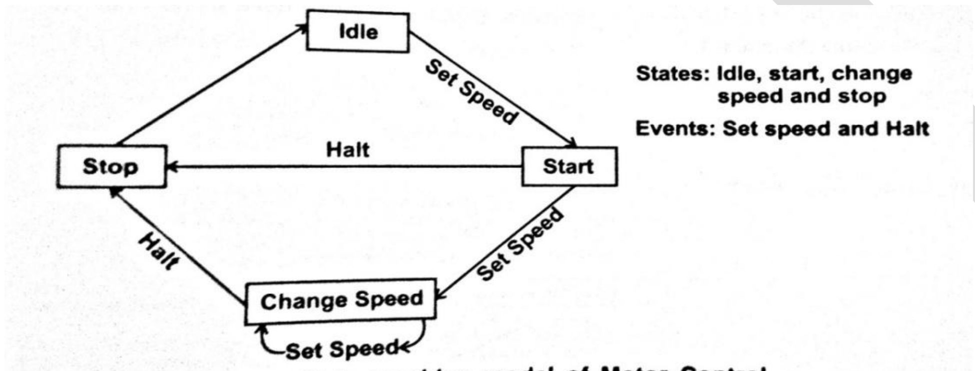

Idle
Set Speed
Stop
Halt
Start
Halt
Set Speed
Change Speed
Set Speed
States: Idle, start, change speed and stop
Events: Set speed and Halt

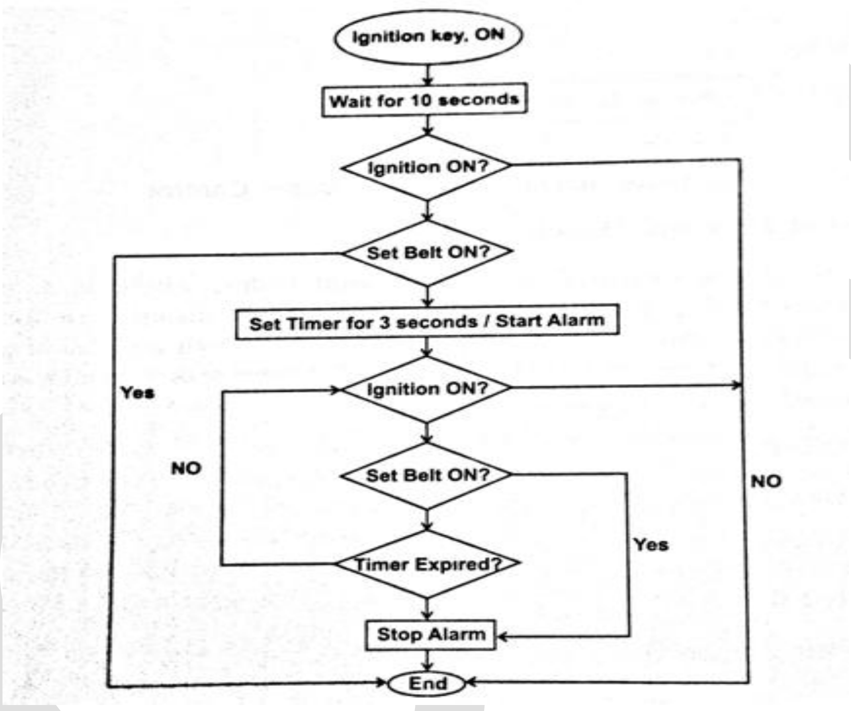
Fig. State machine model of Motor Control

(iii) Sequential Program Model:

The functions are executed in a sequential order, which is same as the conventional procedural programming. The program instructions are iterated and executed with condition and the data gets transformed through a series of operations. The tools for sequential program model are FSM & flowcharts. Consider an example 'Seat Belt warning system' programming.

```
# define ON 1
# define OFF 0
# define YES 1
# define NO 0
Void seatbelt_caution ()
{
Waiting ( );
If (checkIgnition ( ) = = ON)
{
If (checkseat belt ( ) = = OFF)
        {
                SetTimes (3);
                Start Alarm ( );
While ((check seat belt ( ) = = OFF) && (check Ignition ( ) = = OFF) &&(Timer Expire ( )
= = NO));
Stop Alarm ( );
}
}
}
```

Flow chart Approach:



(iv) Concurrent Process model:

This model executes the processes concurrently. It is easier to implement than the conventional sequential execution. Sequential execution leads to poor processor utilization. Concurrent processing model requires additional overheads in task scheduling, task synchronization and communication. Now consider the example of seat belt warning system in concurrent processing model. We can split the tasks into:

1. Timer task for waiting 10 seconds
2. Task for checking the ignition key status
3. Task for checking the seat belt status
4. Task for starting & stopping the alarm
5. Alarm timer task for waiting 3 seconds

These tasks are needed to synchronize.

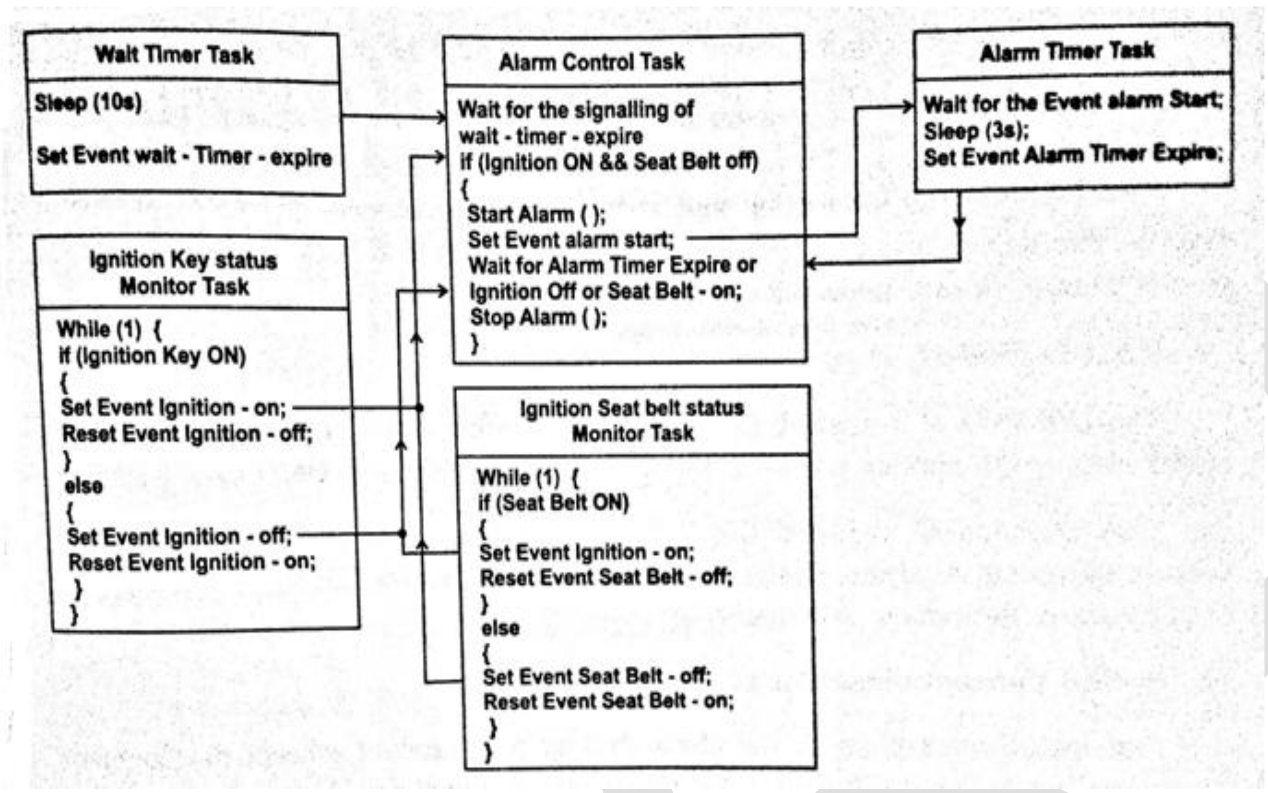

Wait Timer Task
Sleep (10s)
Set Event wait - Timer - expire
Alarm Control Task
Wait for the signalling of
wait - timer - expire
if (Ignition ON && Seat Belt off)
{
Start Alarm ( );
Set Event alarm start;
Wait for Alarm Timer Expire or
Ignition Off or Seat Belt - on;
Stop Alarm ( );
}
Alarm Timer Task
Wait for the Event alarm Start;
Sleep (3s);
Set Event Alarm Timer Expire;
Ignition Key status
Monitor Task
While (1) {
if (Ignition Key ON)
{
Set Event Ignition - on;
Reset Event Ignition - off;
}
else
{
Set Event Ignition - off;
Reset Event Ignition - on;
}
}
Ignition Seat belt status
Monitor Task
While (1) {
if (Seat Belt ON)
{
Set Event Ignition - on;
Reset Event Seat Belt - off;
}
else
{
Set Event Seat Belt - off;
Reset Event Seat Belt - on;
}
}

Figure: Concurrent Processing program model for seat belt warning system.

The concurrent processing model is commonly used for the modeling of real time system.

(v) Object – oriented model:

- It is an object based model
- It splits the complex software requirement into well defined pieces called objects
- It brings re-usability, maintainability & productivity in system design
- Each object is characterized by a set of unique behavior & state. A class is an abstract of a set of objects.
- A class represents the state of an object through member variables and object behavior through member functions.
- The member variables can be public, private or protected
- Private member access within the class and public member access within the class as well as outside the class.
- The concept of object & class brings abstraction, holding and protection.