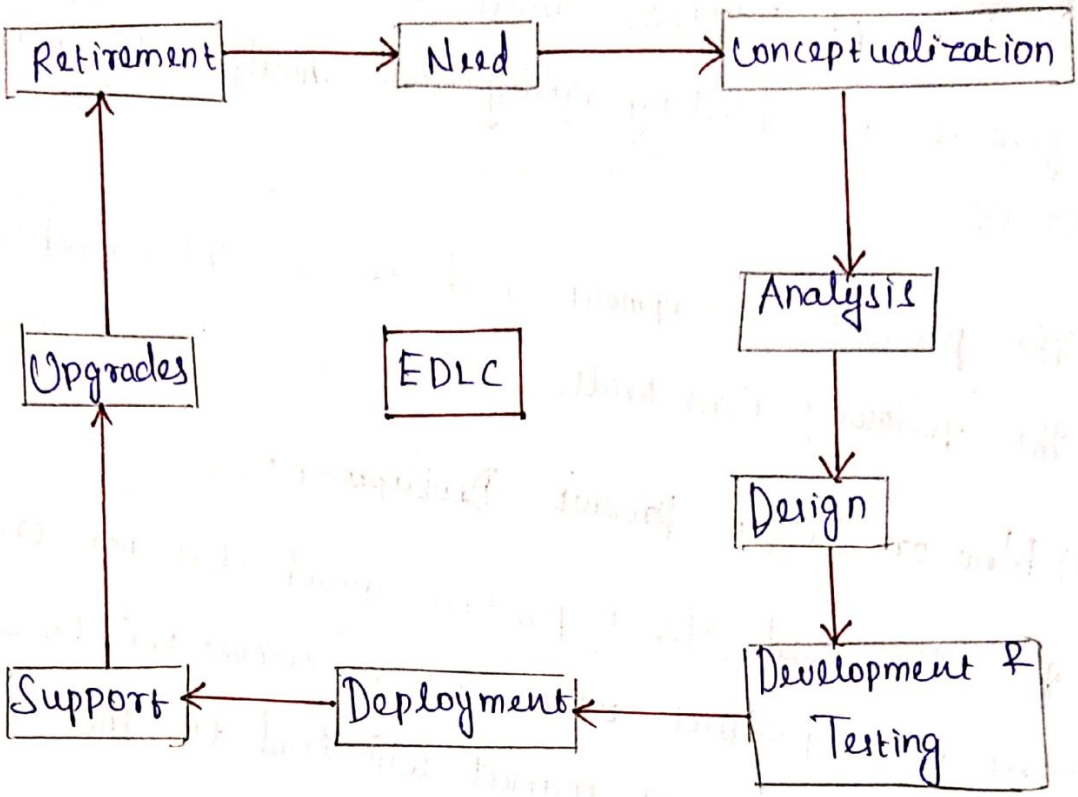
UNIT-3: EMBEDDED FIRMWARE DEVELOPMENT ENVIRONIMENT:

III EEE - Embedded Systems

1. Different phases of EDLC:



* The life cycle of a product development is commonly referred to as Models.
* Models defines the Various phases involved in the life cycle
* The number of phases involved in a model may vary according to the complexity of the product under consideration.

Need:

* Any Embedded Product evolves as an output of a Need.
* The need may come from an individual or from the public from a company.

* Need should be articulated to initiate the product dev. life cycle and based on the need for the product, a 'Statement of Need' or 'Concept proposal' is prepared.

* The 'Concept proposal' must be reviewed by the senior management and funding agency and should get necessary approval.

* The product development need can be visualized in any one of the following three needs.

(i) New or Custom Product Development:

* The need for a product which does not exist in the market or a product which act as competitor to an existing product in the current market will lead to the development of a completely new product.

(ii) Product Re-engineering:

* Re-engineering a product is the process of making changes in an existing product design and launching it as a new version.

* It is generally termed as product upgrade.

* Re-engineering an existing product comes as a result of the following needs

(1) Change in Business Requirements

(2) User Interface Enhancements

(3) Technology upgrades.

(iii) Product Maintenance:

* Product Maintanance 'need' deals with providing technical Support to the end user for an existing product in the Market.

* Product maintenance is generally classified into two Categories.

(i) Corrective Maintanance

(ii) Preventive Maintenance.

Corrective Maintenance:

* It's deals with making Corrective actions following a failure or non-functioning.

Preventive Maintenance:

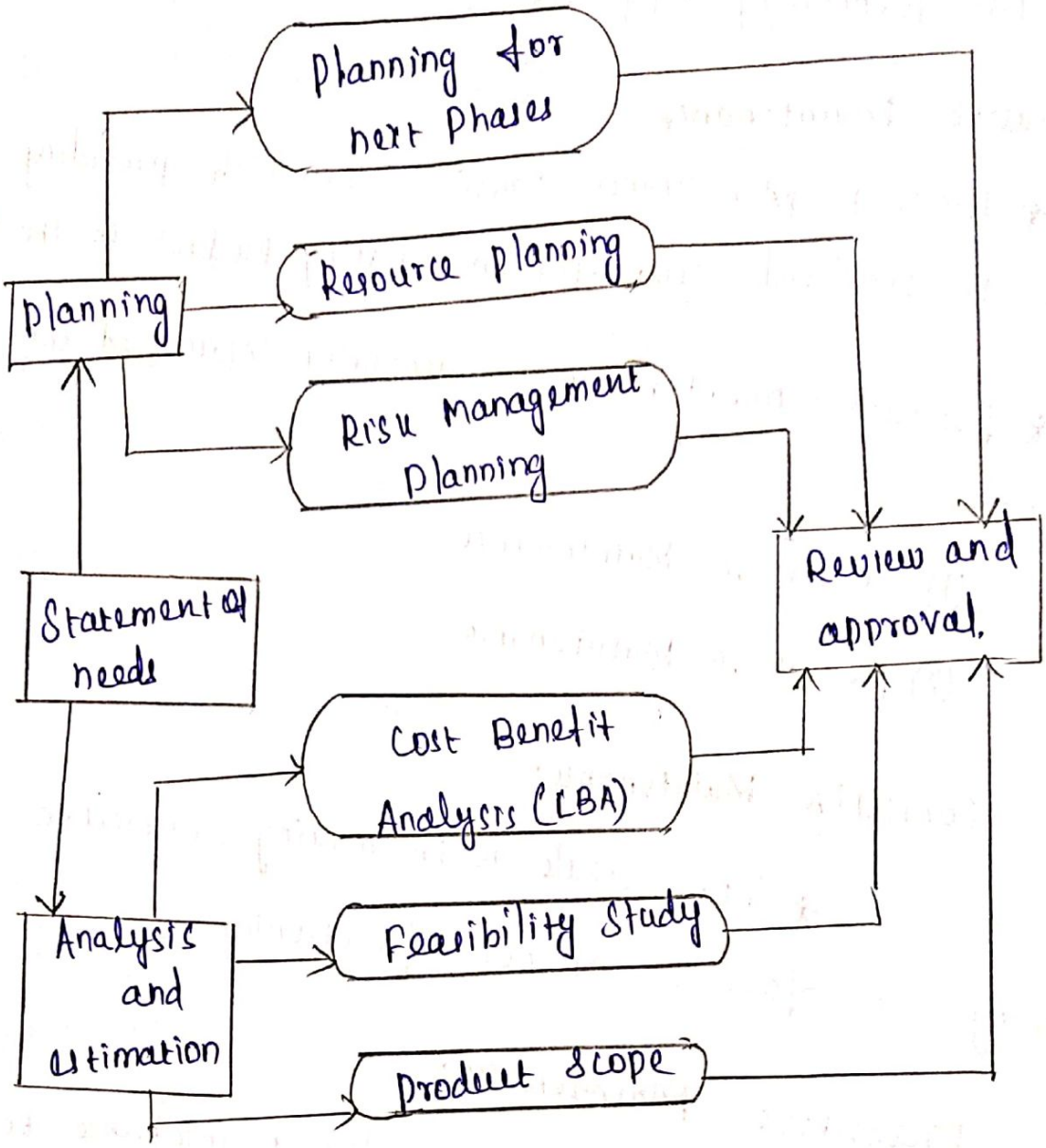
* It is the scheduled maintanance to avoid the failure or non-functioning of the product.

Conceptualization:

* Conceptualization is the 'product concept Development Phase' and it begins immediately after a concept proposal is formally approved.

* The conceptualization phase involves two types of activities namely 'planning activity' and 'Analysis and study

Activity



* The Analysis and Study are performed to understand the Opportunity of the product in the market.

(i) Feasibility Study:

* Feasibility Study examines the need for the product carefully and suggested one or more possible solutions to build the need as a product along with alternatives.

Cost Benefits Analysis (CBA)

* The cost benefit analysis (CBA) is a means of identifying, revealing and assessing the total development cost and the profit expected from the product.

(iii) Product Scope:

* Product Scope deals with what is in scope for the product and what is not in scope the product.

(iv) Planning Activities:

* The planning activity covers various plans required for the product development, like the plan and schedule for executing the next phases following conceptualization, Resource Planning, Risk Management Plans, etc.

1. The product is feasible; proceed to the next phase of the product life cycle.

2. The product is not feasible; scrap the project.

Analysis:

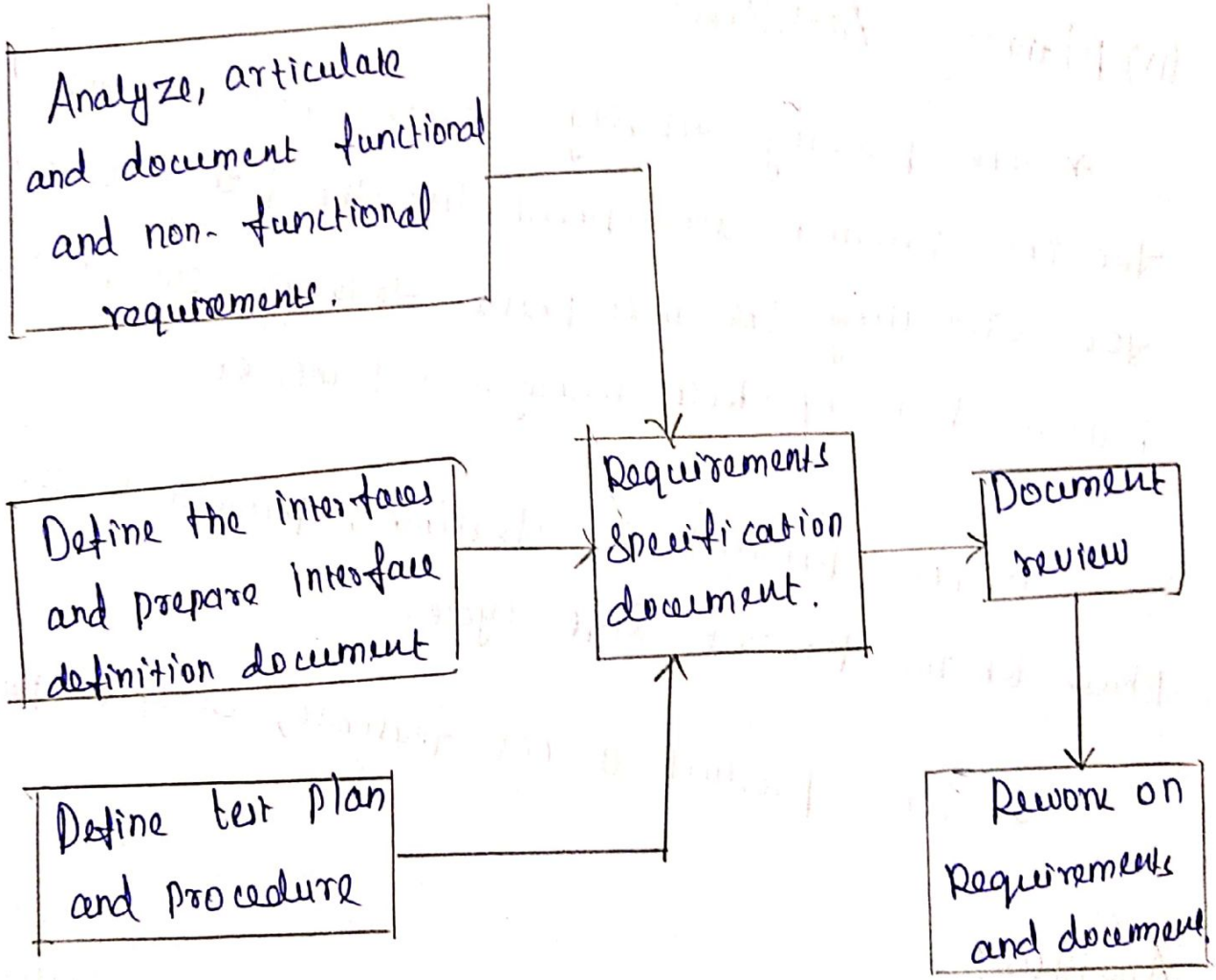
* Requirement Analysis phase starts immediately after the documents submitted during the 'conceptualization' phases is approved by the client or sponsor of the project.

* The product is defined in deal detail with respect to the inputs, processes, outputs and interfaces at a functional

level.

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

* The various activities performed during 'Requirement Analysis' phase is shown in fig:



(i) Analysis and documentation:

* The analysis and documentation activity consolidates the business needs of the product under development and analyses the purpose of the product.

The product under considerations are listed below,

* Functional capabilities like performance, operational characteristics.

1. Operational and non-operational quality attri
2. Product external Interface requirements
3. Data requirements
4. User manuals
5. Operational requirements
6. Maintenance requirements
7. General assumptions.

(ii) Interface Definition and Documentation:

* The Interface definition and documentation activity should clearly analyze on the physical interface as well as data exchange through these Interfaces and should document it.

(iii) Defining Test plan and procedures:

* Identifies the performance of the tests and the coverage of testing the product.

* It is essential for ensuring the total quality of the product.

Types of Testing:

1. Unit testing:

* Testing each unit or module of the product independently for required functionality and quality aspects.

2. Integration testing:

* Integrating each modules and testing the integrated unit for required functionality.

3. System testing:

* Testing the functional aspects or product requirements of the product after integration.

4. User acceptance testing:

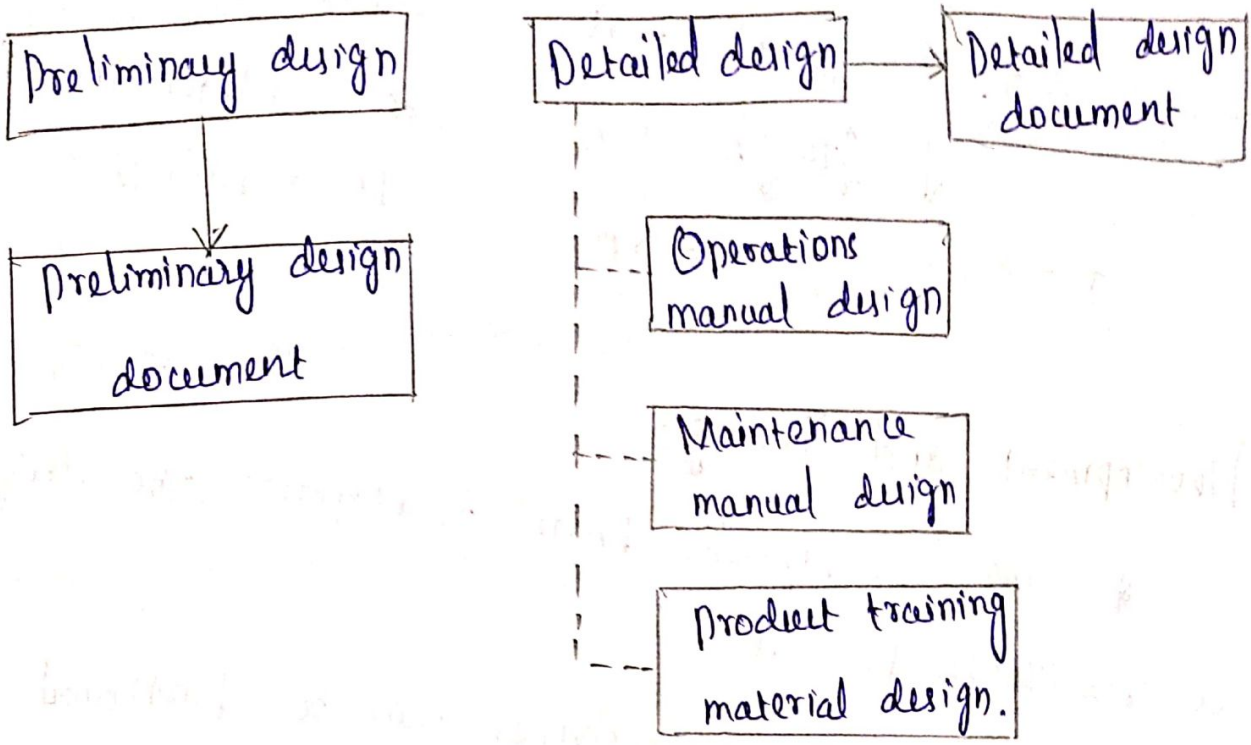
* Testing the product to ensure it is meeting all requirements of the end user.

Design:

* Product Design Phase deals with the entire design of the product taking the requirements into consideration and it focuses on how the required functionalities can be delivered to the product.

* Product design starts with 'Preliminary Design or High level Design'.

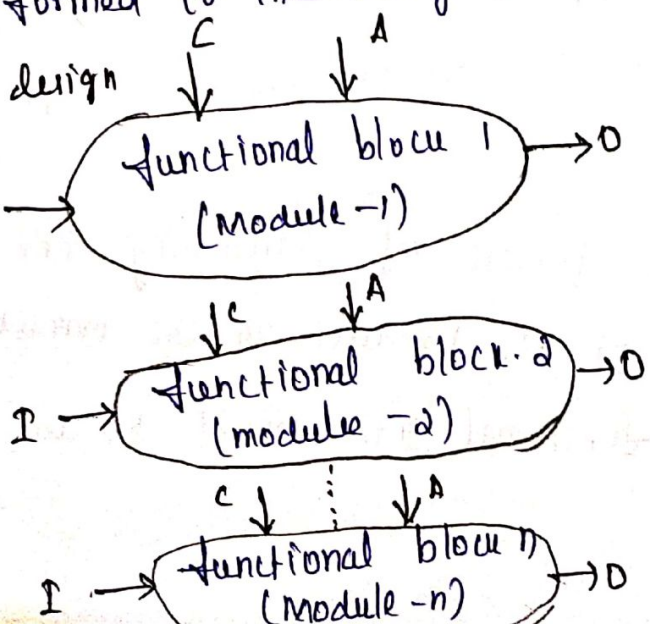
The functional block will look like a black box at this design point, with only inputs and outputs of the block being defined.



* On completion, the Preliminary Design Document (PDD) is sent for review to the end-user or client who come up with a need for the product

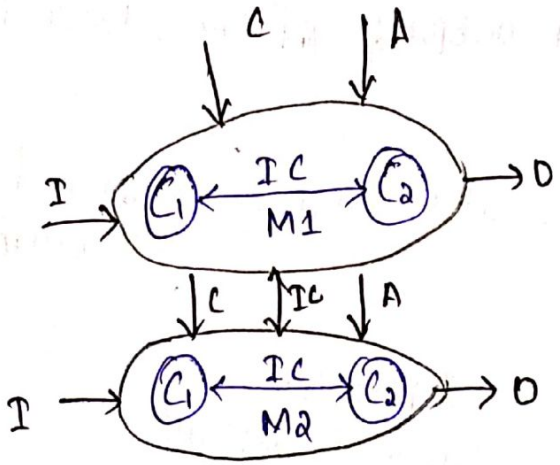
* If The end user wants modification on the design, it can be informed to the design team through review comments.

Preliminary design Illustration:



I – Inputs
O – Outputs
C – Constraints
A – Assumptions.

Detailed design Illustration:



C_1, C_2 → Component.

IC → Inter-connection

M → Module.

Development and Testing:

* The 'Development Phase' transforms the design into a realizable product.

* The development activities can be partioned into embedded hardware development, embedded firmware development and product enclosure development.

* Embedded hardware development refers to the development of the component placement platform - PCB using CAD tools and its fabrication using CAM tools.

Deployment:

* It is the process of launching the 1st fully functional model of the product in the market or handling over the fully functional intial model to an end user or client.

is also known as First customer shipping (FCS)

* The Deployment Phase is Initiated after the System is tested and accepted by the end user.

* The Important tasks performed during the deployment Phase are listed below.

1. Notification of product Deployment:

* The notification can be sent out through e-mail, media, etc. mentioning the following in a few words.

⇒ Deployment schedule (Data, Time and venue)

⇒ Brief description about the product

⇒ Targeted end users

⇒ Extra features Supported with respect to an existing product.

⇒ Product Support Information, including the Suppo Person, name, contact number, Contact mail ID, etc.

2. Execution of training Plan:

* Proper training should be given to the end user to get them acquainted with the new product.

3. product Installation:

Install the Product as per the Installation

document to ensure that it is fully functional.

4. product post-Implementation review:

* Once the product is launched in the market, conta a post-Implementation review to determine the success of the product.

Support:

* The Support phase deals with the Operations and Maintanance of the product in a production environment.

* Bugs in the product may be Observed and reported during the Operations phase.

* Support should be provided to the end user or client to fix the bugs in the product.

Upgrades:

* The upgrade phase of product development deals with the development of upgrade for the product which is already present in the Market.

* Product upgrade result as an output of major bug fixes or feature enhancement requirements from the end users.

* Version numbering is essential for backward trana-bility

Releasing of upgrades is managed through release Management.

Retirement / Disposal:

* Due to increased user needs and revolutionary technological changes, a product cannot sustain in the market for a long time.

* The disposal / retirement of a product is a gradual process.

* The product is declared as obsolete and discontinued from the market.

* The disposition of a product is essential due to the following reasons

1. Rapid technology advancement
2. Increased User needs.

② MODELLING OF EDLC :

* Interconnection of various phases involved in the development of an embedded product.

(i) LINEAR OR WATERFALL MODEL

* Adopted in older systems and this approach is executed in sequences.
* Unidirectional flow with output of one phase serving as the input to the next phase.
* Gives an insight into what should be done in the next phase and how it can be done.
* Feedback of each phase is available locally and only after they are executed
* Significant feature of this model is that even if bugs are identified in current design, corrective actions are not implemented immediately and the development process proceeds with current design

ADVANTAGE :

* Rich in documentation
* Easy project management
* Good control over cost and schedule
* Suited for well defined requirements and within the scope, no change requests are expected till the completion of life cycle

DISADVANTAGE :

* All analysis can be done and everything will be in right place without doing any design or implementation

Need (Statement of need)
↓
Conceptualization (System boundary definition, CBA report)
↓
Analysis (Requirements gathering)
↓
Design (Preliminary & Detailed design)
↓
Development & Testing (Unit / Integration System testing)
↓
Deployment
↓
Support
↓
Upgrades
↓
Requirements

Hardware & Firmware development

Product launching

Maintainence and Bug fixes

Product upgrades

Product disposal

LINEAR MODEL OF EDLC

(ii) ITERATIVE / INCREMENTAL OR FOUNTAIN MODEL :

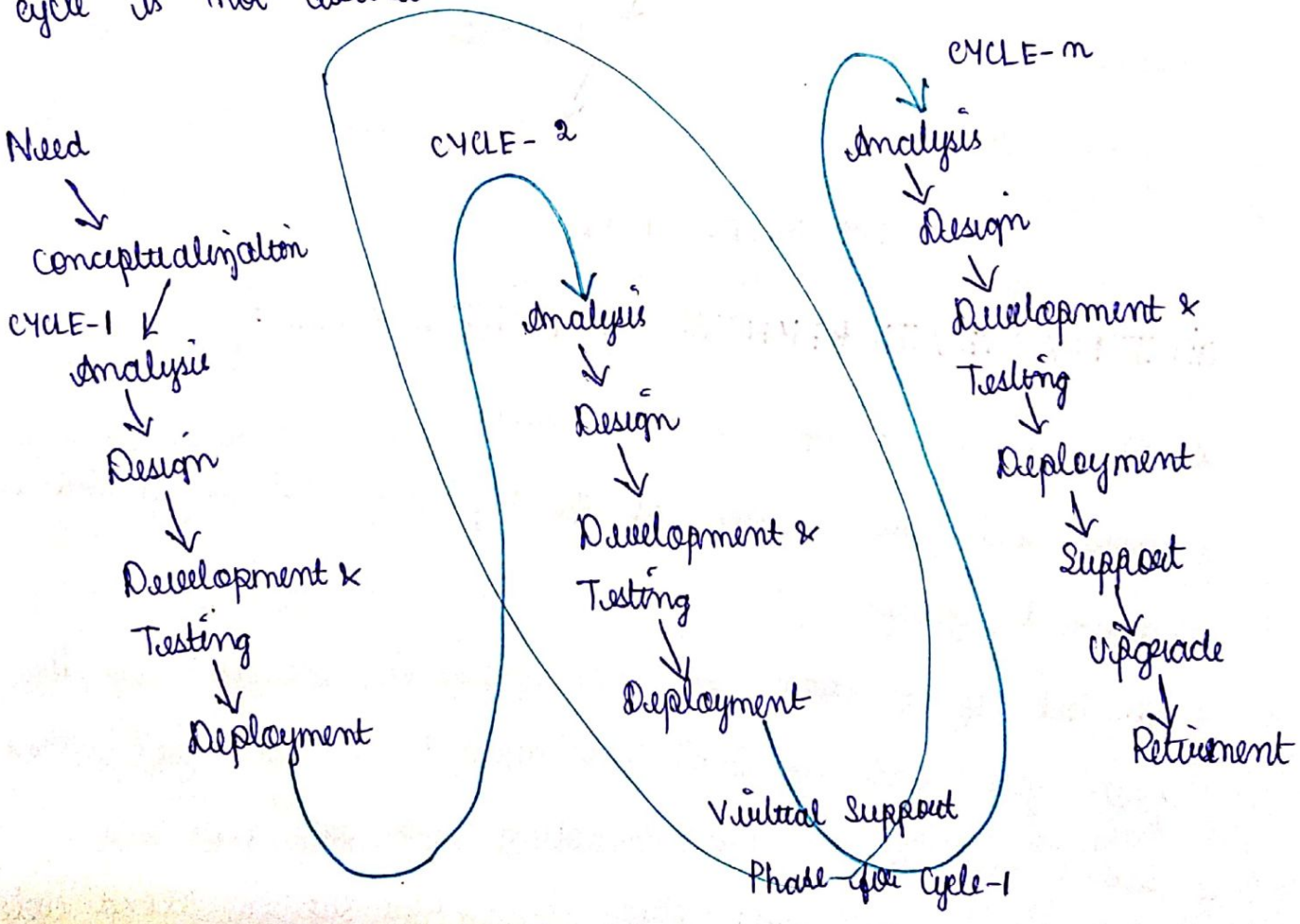
* Cascaded series of linear models
* Requirements are known at the beginning and are divided into different groups.
* Core set of functions for each group is identified in the first cycle and is built and deployed as the first release.
* Second set of requirements along with bug fixes and modification for first release is carried out in second cycle

and the process is repeated until all functionalities are implemented and they are meeting the requirements.

* Another approach is the overlapped model where development cycles overlap, that means subsequent iterative cycle may begin before the completion of previous cycle.

ADVANTAGE :

* Very good development cycle feedback at each function or feature implementation.
* Product development can be stopped at any stage with a base minimum working product
* Suited for developments where the continued funding for each cycle is not assured.

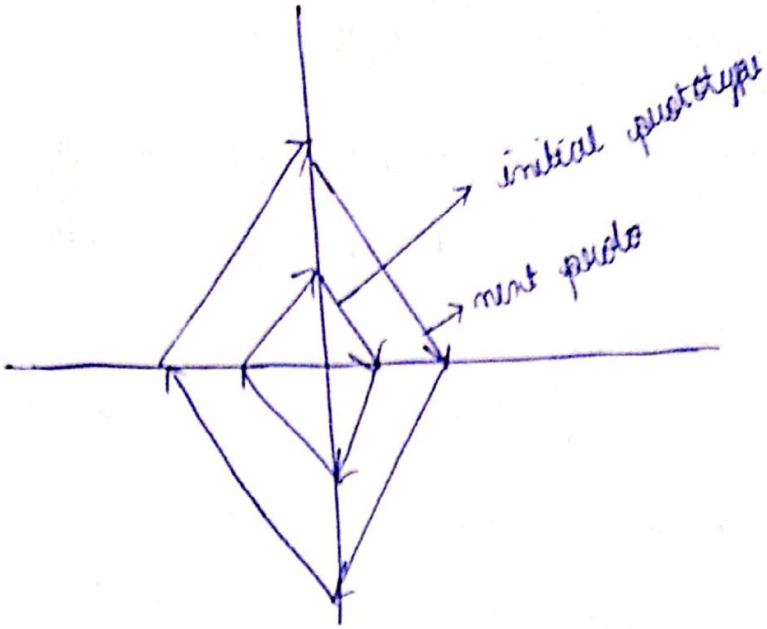


PROTOTYPING / EVOLUTIONARY MODEL

* Product development in multiple cycles.
* Model produces a more refined prototype of the product at the end of each cycle instead of functionality or feature addition in each cycle
* Shortcomings of this model after each cycle are evaluated and it is fixed in the next cycle.
* After initial requirement analysis, design for first prototype is made, development process is started
* On finishing prototype, it is sent to the customer for evaluation.
* After finite number of iterations, final product is delivered to the customer.

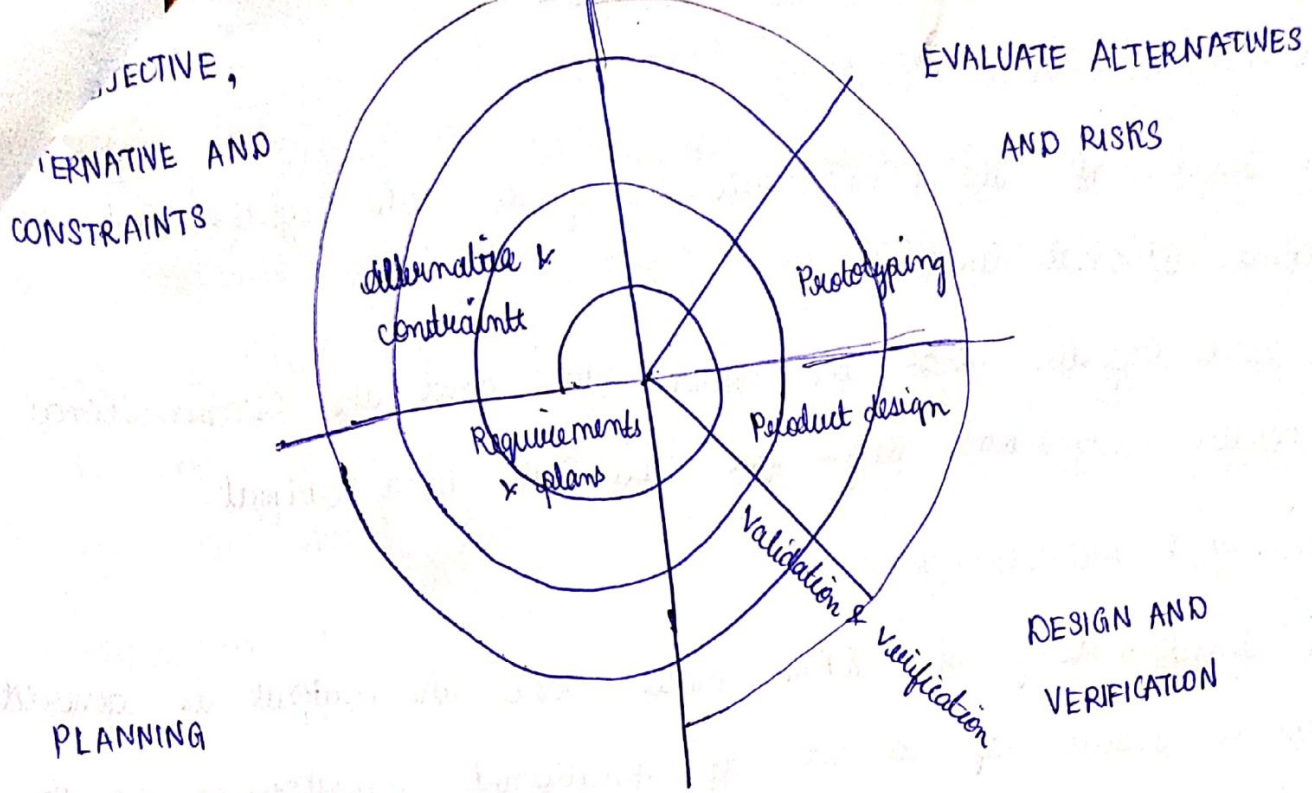
DRAWBACKS :

* Deviations from expected cost and schedule due to requirements refinement
* Increased project management.
* Increased configuration management activities
* Suited for products whose requirements are not fully available and are subject to change
* Not suited for projects involving the upgradation of existing product.



(iv) SPIRAL MODEL :

* Combines linear and prototyping models
* Starts with project definition and traverses through all phases of EDLC.
* Activities involved,
  → Determine objectives, alternatives & constraints
  → Evaluate alternatives, identify and resolve risks
  → Develop & Test
  → Plan
* Suited for complex embedded products and situations where requirements are changing from customer side.
* Risk evaluation in each stage helps in risk planning and mitigation



3. FUNDAMENTAL ISSUES IN HARDWARE SOFTWARE CO-DESIGN

SELECTING THE MODEL :

* Used for capturing and describing the system characteristics
* When the design moves to the implementation aspect, the information about the system component is revealed & the designer has to switch to a model capable of capturing the system's structure.

SELECTING THE ARCHITECTURE :

* Specifies how a system is going to implement in terms of the number and different types of components and interconnection among them.
* Commonly used architectures are Controller architecture, Datapath architecture, CISC, RISC, VLIW, SIMD, MIMD, etc.,

CONTROLLER ARCHITECTURE :

* Implements the FSM model using a state register and two combinational circuits.
* State Register holds the present state and the combinational circuits implement logic for next state and output.

DATAPATH ARCHITECTURE :

* Implementing the DFG model where the output is generated as a result of a set of predefined computations on the input data
* Represents a channel between the input and output and in datapath architecture, the datapath may contain registers, counters, register files, memories and ports.

FINITE STATE MACHINE DATAPATH (FSMD)

* Combines controller architecture with datapath architecture
* Implements controller with datapath
* Consists of two I/O ports out of which one acts as the control port for receiving or sending the control signals from or to controller unit and the second I/O port interfaces the datapath with external world for data input and data output.

PLEX INSTRUCTION SET COMPUTING (CISC)

* Uses instruction set representing complex operations
* CISC instruction set to perform a large complex operation with a single instruction.
* Reduces the program memory access and program memory size requirement.
* Datapath is complex.

VLIW :

* Implements multiple functional units.
* Packages one standard instruction per functional unit of the datapath.

PARALLEL PROCESSING ARCHITECTURE :

* Implements multiple concurrent processing elements & each PE may associate a datapath containing register and local memory
* Examples :- SIMD, MIMD.
* Scheduling of instruction execution and controlling of each PE is performed through single controller.

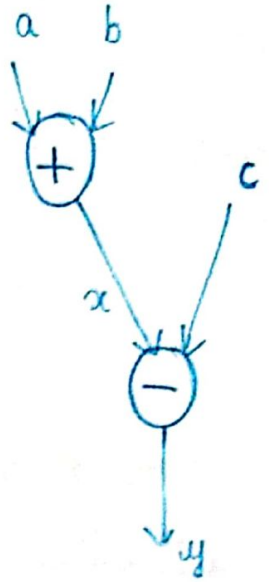
SELECTING THE LANGUAGE :

* Model can be captured using multiple programming languages like C, C++, C#, JAVA, etc., for software implementation and languages like VHDL, system C, verilog, etc., for hardware implementation.

MPUTATIONAL MODELS IN EMBEDDED DESIGN

(i) Dataflow Graph Model (DFG) :

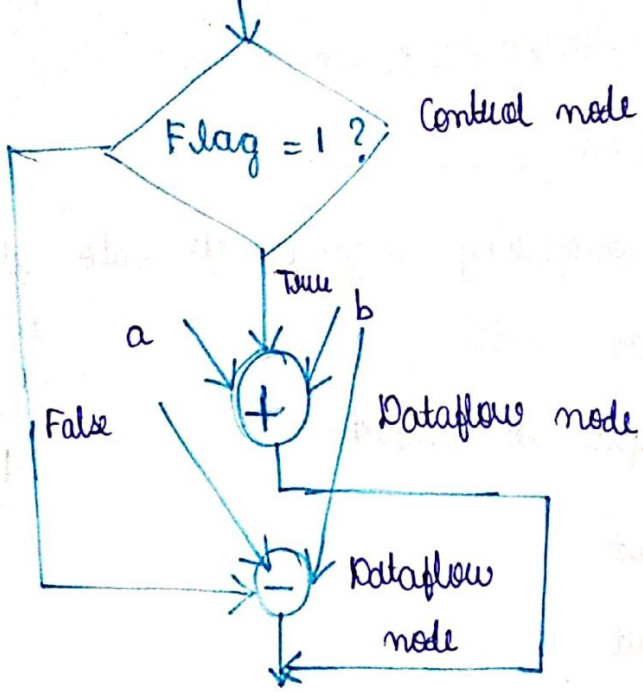
* DFG translates data processing requirements into dataflow graph
* Also called as Diagram model
* Data flows from input to output.
* Inward arrow → input
  Outward arrow → output.
* Acyclic DFG ⇒ does not contain multiple values for the input variable and multiple output values for a given set of inputs.



* Feedback inputs, events are examples of non-acyclic inputs.

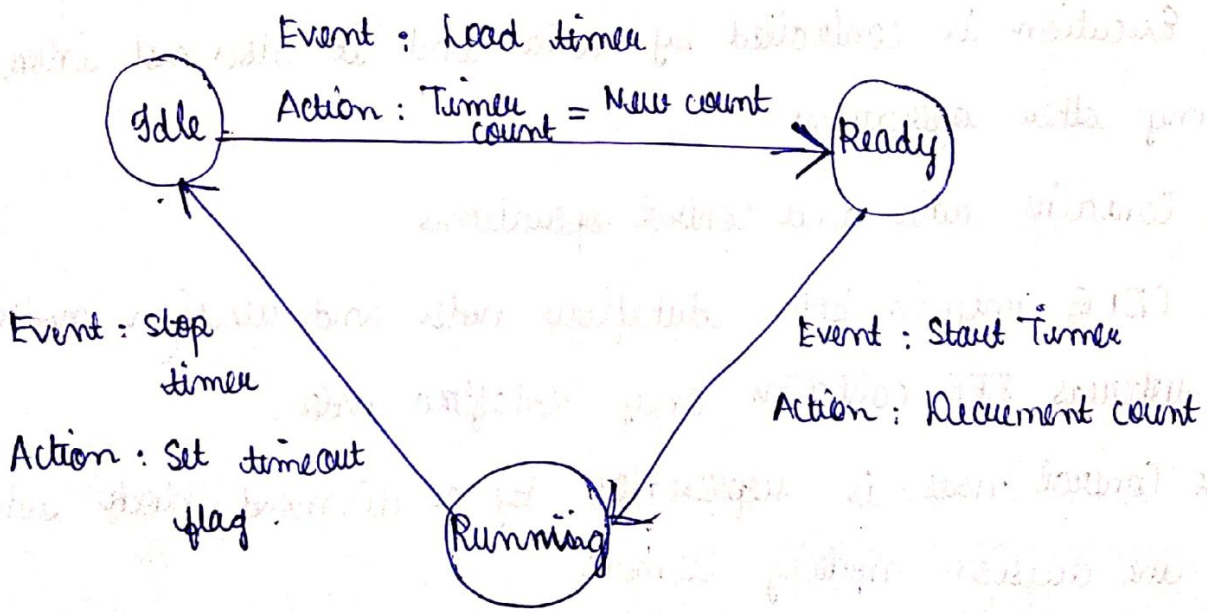
(ii) Control Dataflow Graph (CDFG) :

* Execution is controlled by data and it does not invoke any other operations
* Contains data and control operations.
* CDFG contains both dataflow nodes and decision nodes, whereas DFG contains only dataflow nodes.
* Control node is represented by a diamond block which is the decision making element



(iii) STATE MACHINE MODEL :

* Describes the system behaviour with States, Events, Actions and Transitions.
* State → Represents current situation
* Event → Input to the state.
* Transition is the movement from one state to another.



SEQUENTIAL PROGRAM MODEL :

* Functions or processing requirements are executed in sequence
* FSM and flowchart are good choice for sequential program modeling.
* Program instructions are iterated and executed conditionally

(V) CONCURRENT / COMMUNICATION PROCESS MODEL :

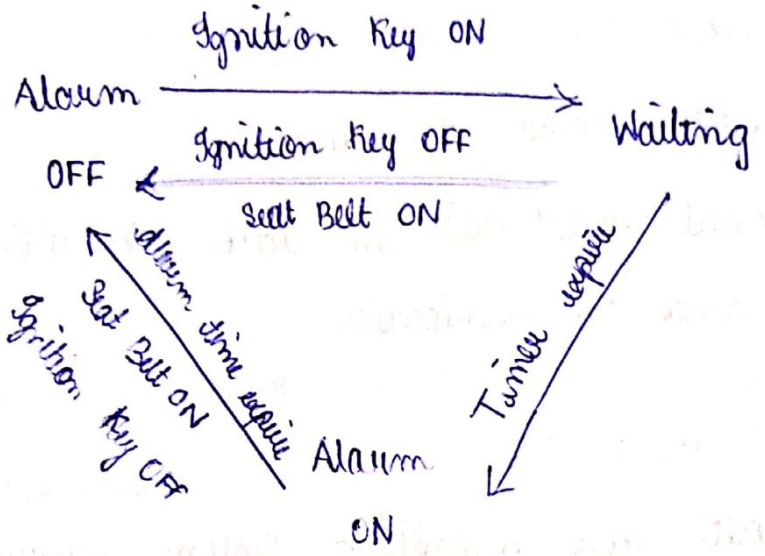
* Concurrently executes tasks or processes.
* Requires additional overheads in task scheduling, task synchronization and communication.

(vi) OBJECT ORIENTED MODEL

* Object based model for modelling system requirements
* Disseminates complex software requirement into simple well defined pieces
* Object → set of unique behaviour and state
* Class → description of a set of objects and considered as a "blueprint of an object.
* Brings together abstraction, hiding and protection

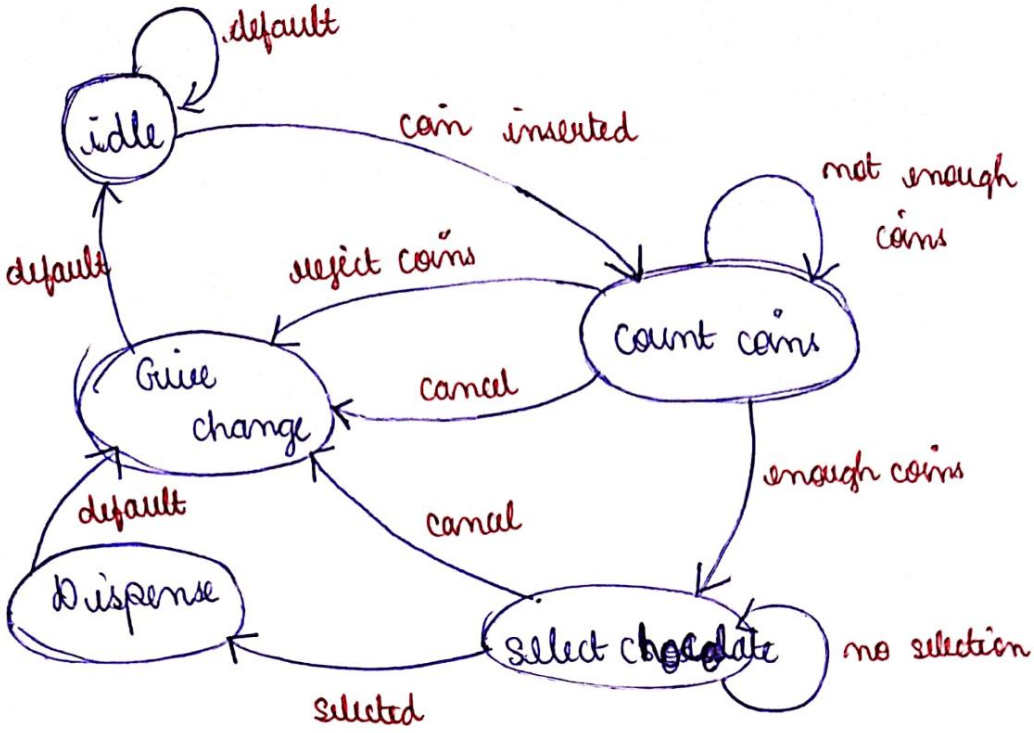
5. STATE MACHINE MODEL :

* Describes the system behaviour with states, Events, Actions and Transitions
* State → Represents current situation
* Event → Input to the state



- When the vehicle ignition is turned ON, Seat belt is not fastened within 10 seconds of ignition ON, system generates an alarm signal for 5 seconds
- Alarm is turned OFF when the alarm time expires or if the driver or passenger fastens the belt or if the ignition switch is turned OFF, whichever happens first.
- Here the states are Alarm OFF, waiting and Alarm ON and the events are Ignition Key ON, Ignition Key OFF, Timer expire, Alarm Time expire and seat belt ON.

AUTOMATIC CHOCOLATE VENDING MACHINE



States : Idle, count coins, select soda, dispense, give change

Events : Default, coin (insert, reject), select, cancel.