Hardware Software Co-design and Embedded Product development lifecyle management:

Hardware Software Co-design

- Design of ES involves two essential parts of design i.e. Hardware and software, both are equally important

- Because in complex system, hardware and software design influence each other and therefore should be in done in structured and systematic way.

- Co-design implies that hardware and software design be considered concurrent and be done together

Hardware Software Co-design

- Different from thinking that hardware should be designed first and software should be taken care of later and then ported into hardware.

- Because of growing complexity of ES, techniques for supporting hardware software co-design have been developed in order to permit the joint specification design and synthesis of hardware software systems.

Factors in current design scene which support co-design

- The availability of hardware components like processors which are programmable-MCUs, DSP processors etc. –fall in this category

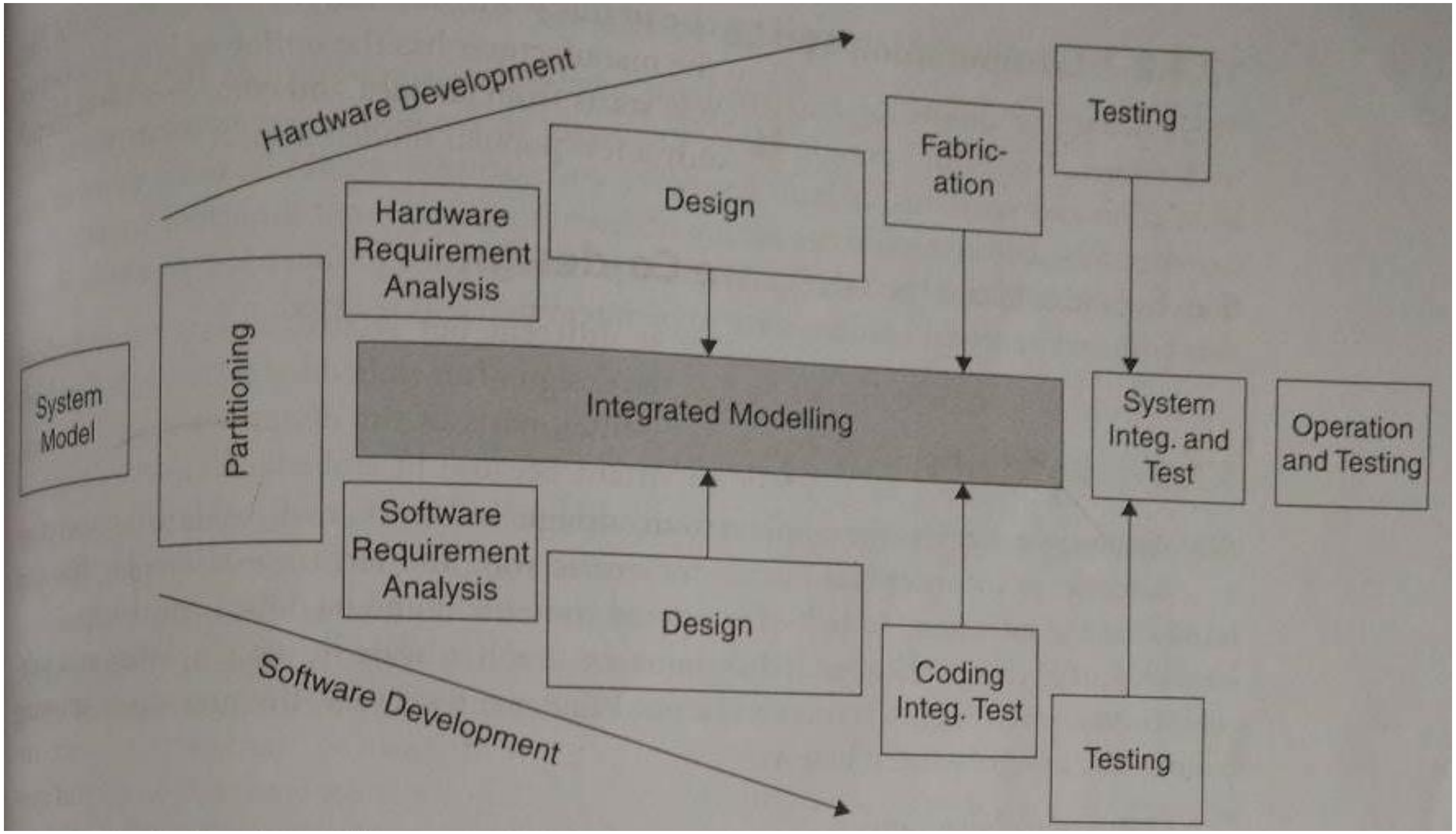
- The availability of re-configurable hardware like FPGAs and CPLDs.

- Design is made easy because of efficient C compilers and hardware description languages that automate design using synthesis and simulation tools.

Steps in Codesign

1. Functional Design
2. Mapping/Partitioning
3. Design of hardware and Software
4. Co-simulation and co-synthesis

Co-design illustrated with steps in development process



Functional Design

- A system should be functionally correct.
- For this to be ensured the behaviour of system should be specified correctly.
- Behavioural model should be made first .
- A set of inputs, set of outputs and mapping between two constitutes the behaviour and is equivalent to defining the system.
- This description is independent of HW and SW, several representations may be utilized.

Mapping/ Partitioning

- The next step is to divide the functionality into two parts for implementation into hardware and software called partitioning.

- Decision on which parts are best implemented in hardware and which in software , makes an impact on factors like speed, power, performance etc.

- Partitioning also means that design is broken down into smaller units each of which has defined ‘functionality’. This idea applies to hardware as well as software.

Mapping/ Partitioning

- Mapping also implies that a particular function is mapped to that is assigned to be performed by specified block.

- There is continual interaction between two partitions as illustrated by integrated modeling.

Design of Hardware and Software

- Hardware and software design proceed in two separate directions. But there is continuous interaction between two design teams such that feedback, correction and re-design is done.

- Once hardware and software designs are completed, the system is integrated and tested and then put into operation

Co-simulation and Co-synthesis

- Simulation implies testing of functionality without actual implementation

- Synthesis means the final realization of designed product.

- Co-synthesis means that software and hardware are to be integrated and realized

Modeling of system

- A model is conceptual notion for expressing function of a system

- Embedded systems are reactive systems which respond to events.

- Systems can be designed using many techniques
  i. Data/ control Diagrams
  ii. Concurrent Models
  iii. Finite States Machine

i. Data Flow Diagram/DFD

- In such a model, it is flow of data alone that is taken into consideration.
- It is a data driven model

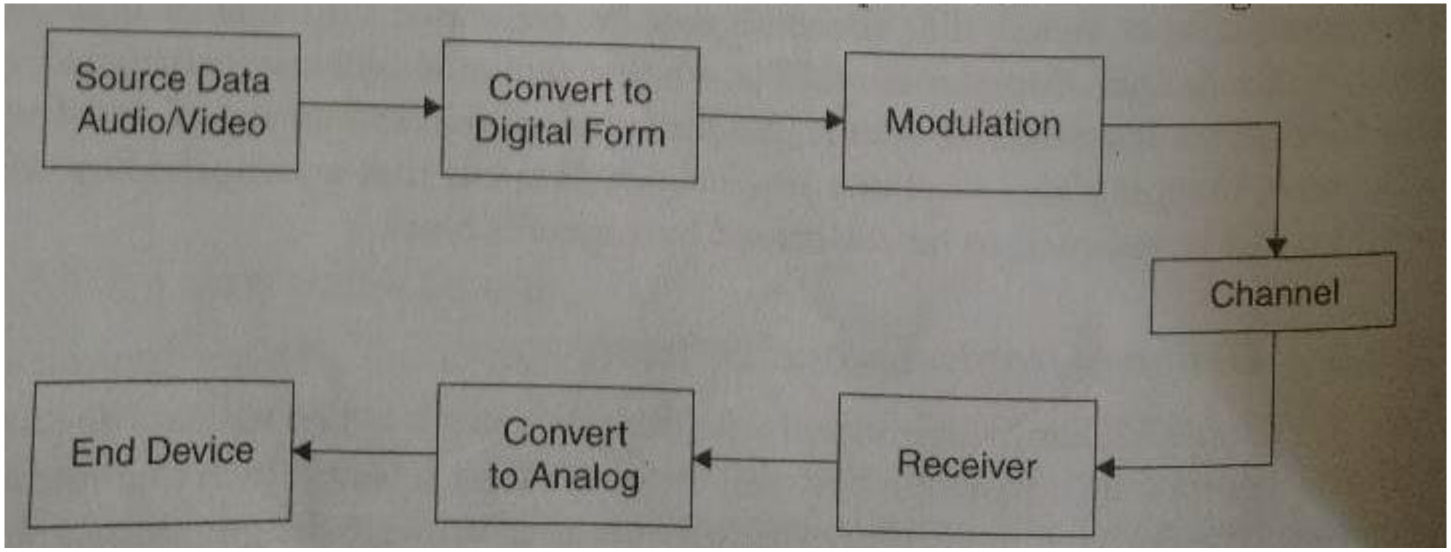
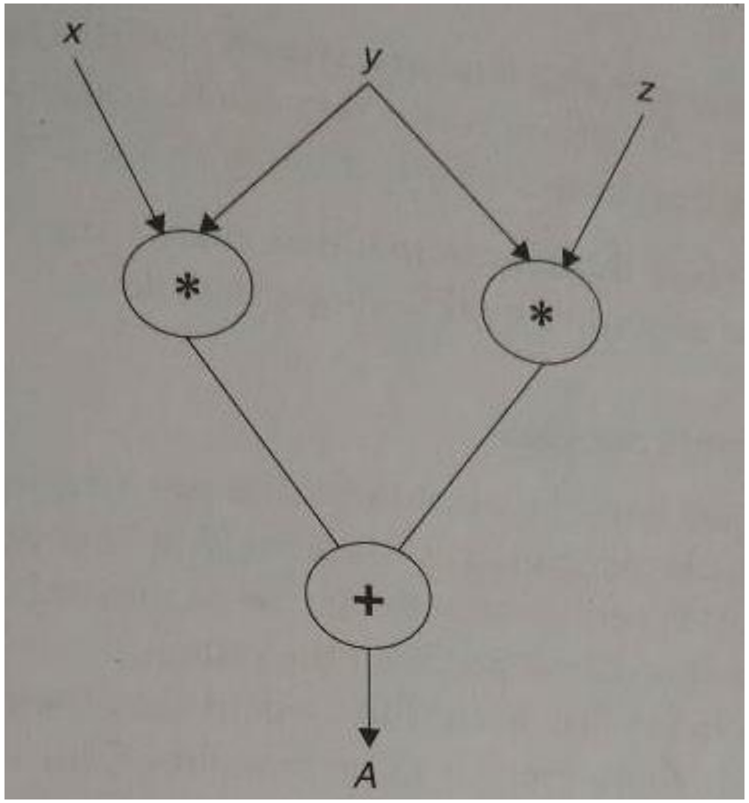


Fig: Flow of data in communication system

DFD

- Each box is subsystem which is data driven.
- A particular box obtains data as input from previous block
- Each functional block may need to wait till it gets processed data from previous block

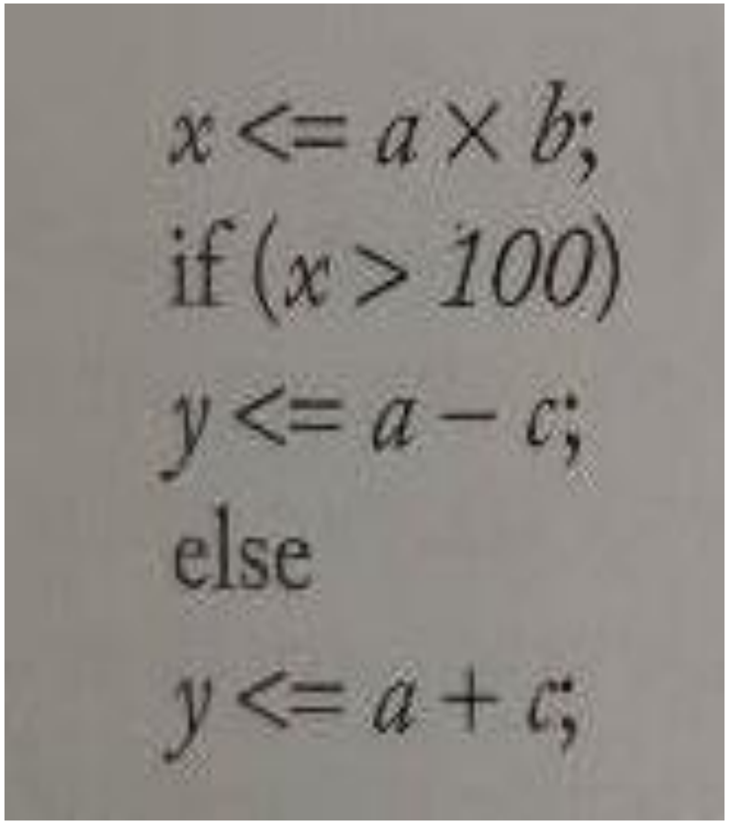


Data flow Diagram for A=xy+yz

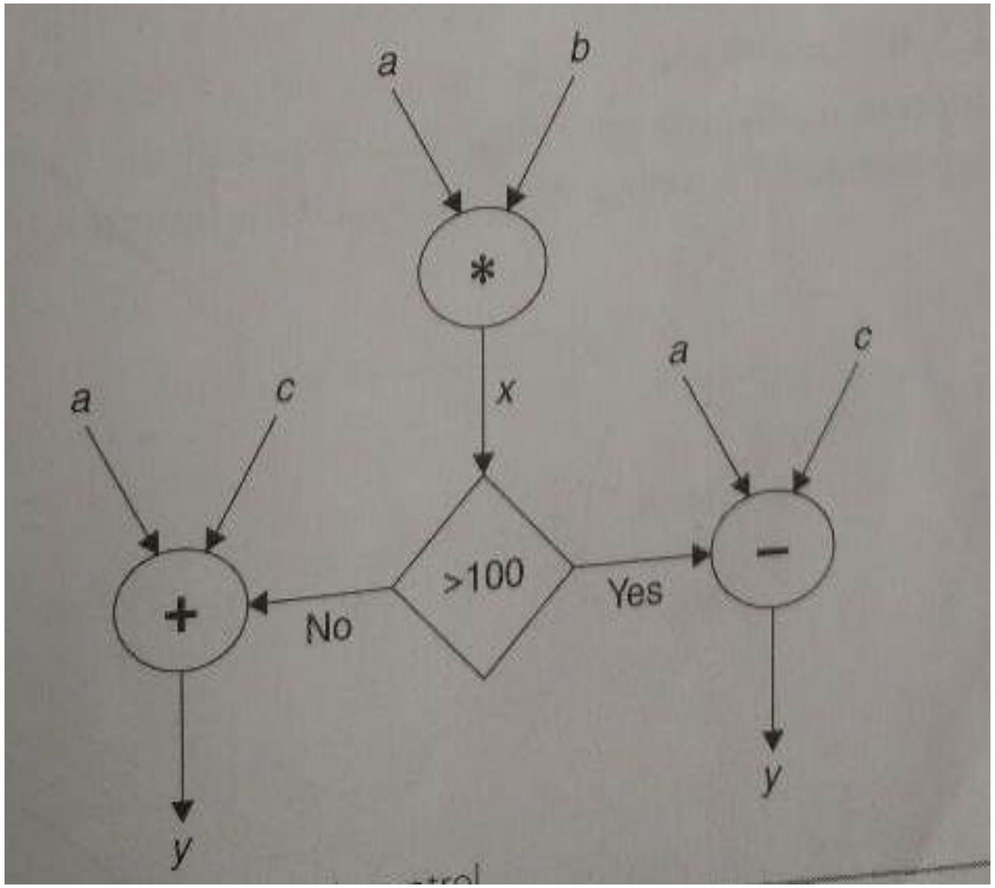
Control/Data Flow graph

•This is data flow graph with control added to it.

•Conditional processing activities fit into this type of graph



```
x <= a × b;
if (x > 100)
y <= a − c;
else
y <= a + c;
```



Do we realize DFDs into Hardware or Software

- They can be realized either ways and that would have been decided at time of partitioning in first level

How flow chart different from DFG

- Flowchart represents ‘control flow’ only, while a DFG shows ‘data flow’

ii. Concurrent Processes

- These are the processes in which activities are scheduled to happen concurrently.

- Activities are performed actually in parallel or it is ‘pseudo parallelism’ that occurs, depends on the system

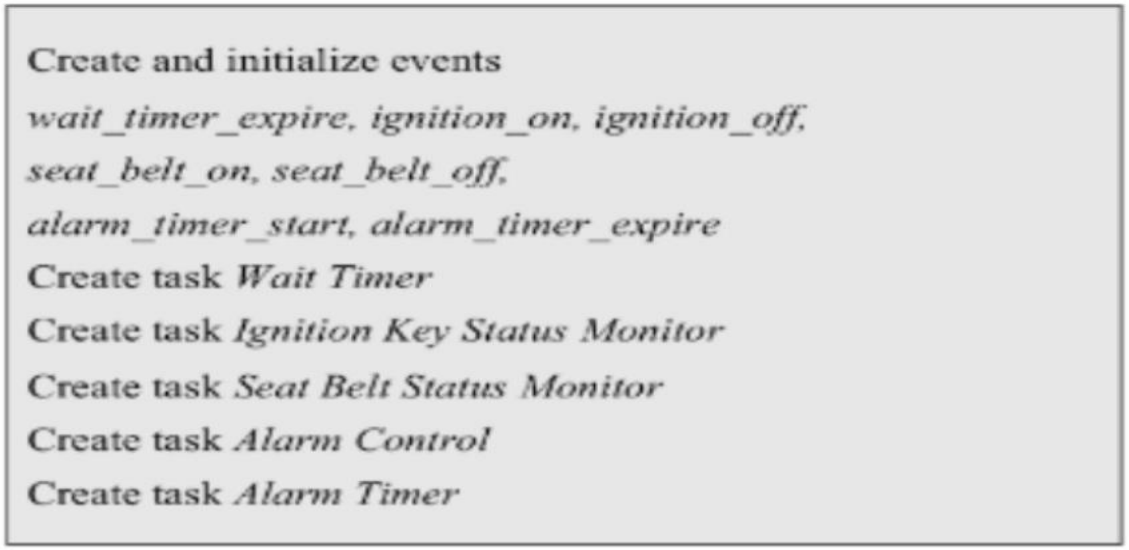
- Concurrency is to be thought of software, then tools for software design should ensure the correct implementation for inter-process communications with synchronization

ii. Concurrent Processes

- In hardware design, the idea of concurrency appears.
- All the parts of circuit are to be work in parallel, the design of hardware finally boils down to writing codes with concurrent statements.
- There is communication between these processes/ statements and hardware is finally realized from such concurrent code which leads to hardware structure

Example

Concurrent/Communicating Process Model



Create and initialize events

wait_timer_expire, ignition_on, ignition_off,

seat_belt_on, seat_belt_off,

alarm_timer_start, alarm_timer_expire

Create task *Wait Timer*

Create task *Ignition Key Status Monitor*

Create task *Seat Belt Status Monitor*

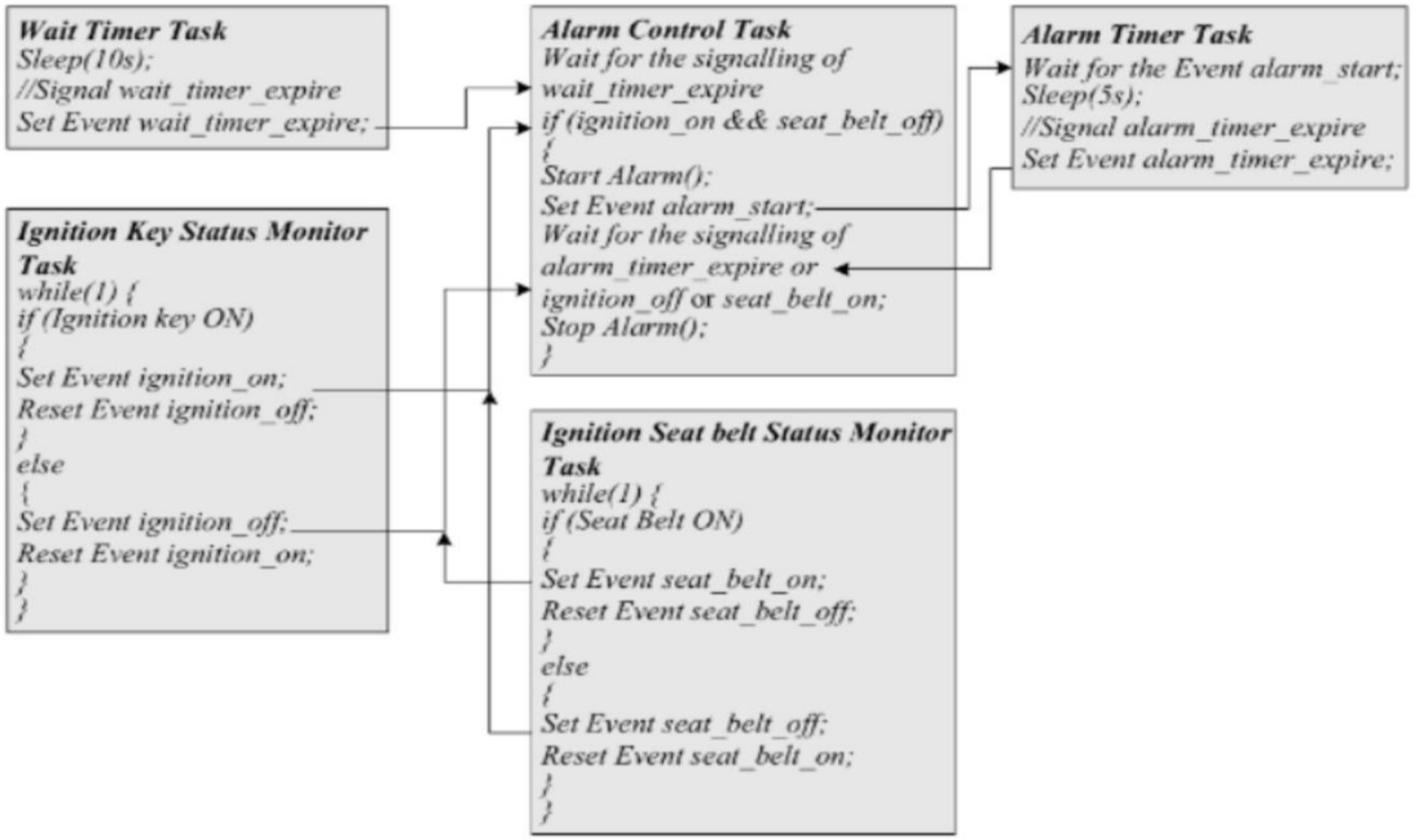
Create task *Alarm Control*

Create task *Alarm Timer*

Tasks for 'Seat Belt Warning System'

Example
Concurrent Process

Concurrent/Communicating Process Model



```
Wait Timer Task
Sleep(10s);
//Signal wait_timer_expire
Set Event wait_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;
}
}
```

```
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 the signalling of
alarm_timer_expire or
ignition_off or seat_belt_on;
Stop Alarm();
}
```

```
Alarm Timer Task
Wait for the Event alarm_start;
Sleep(5s);
//Signal alarm_timer_expire
Set Event alarm_timer_expire;
```

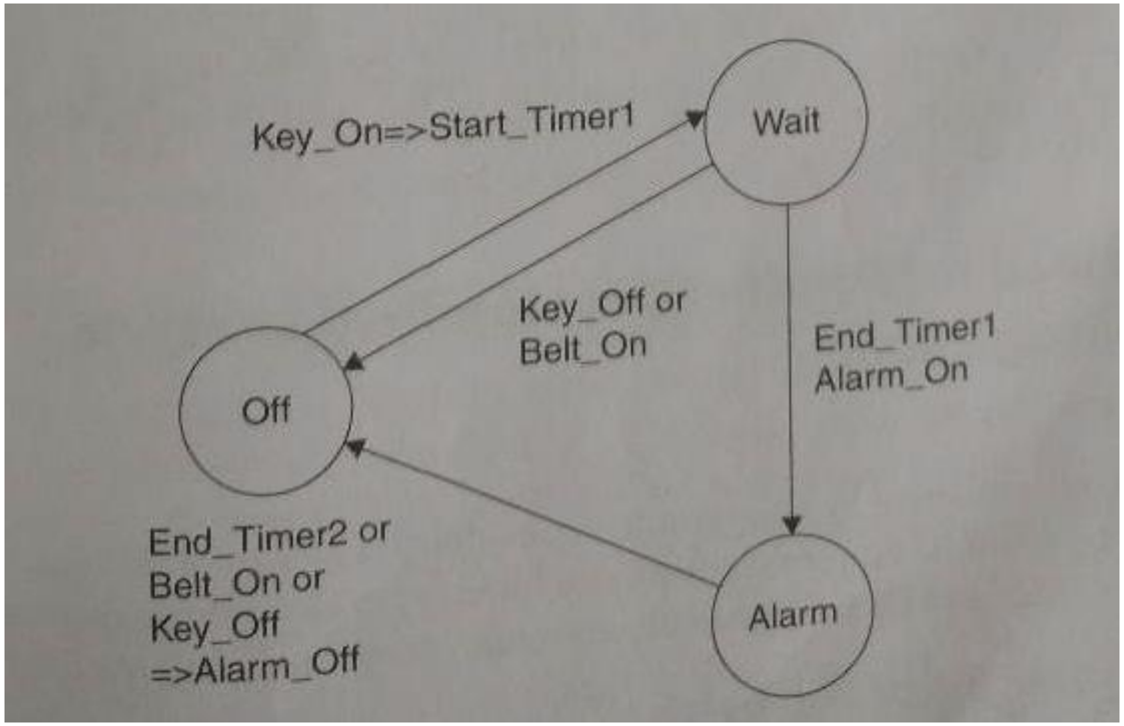
```
Ignition Seat belt Status Monitor
Task
while(1) {
if (Seat Belt ON)
{
Set Event seat_belt_on;
Reset Event seat_belt_off;
}
else
{
Set Event seat_belt_off;
Reset Event seat_belt_on;
}
}
```

Concurrent processing Program model for 'Seat Belt Warning System'

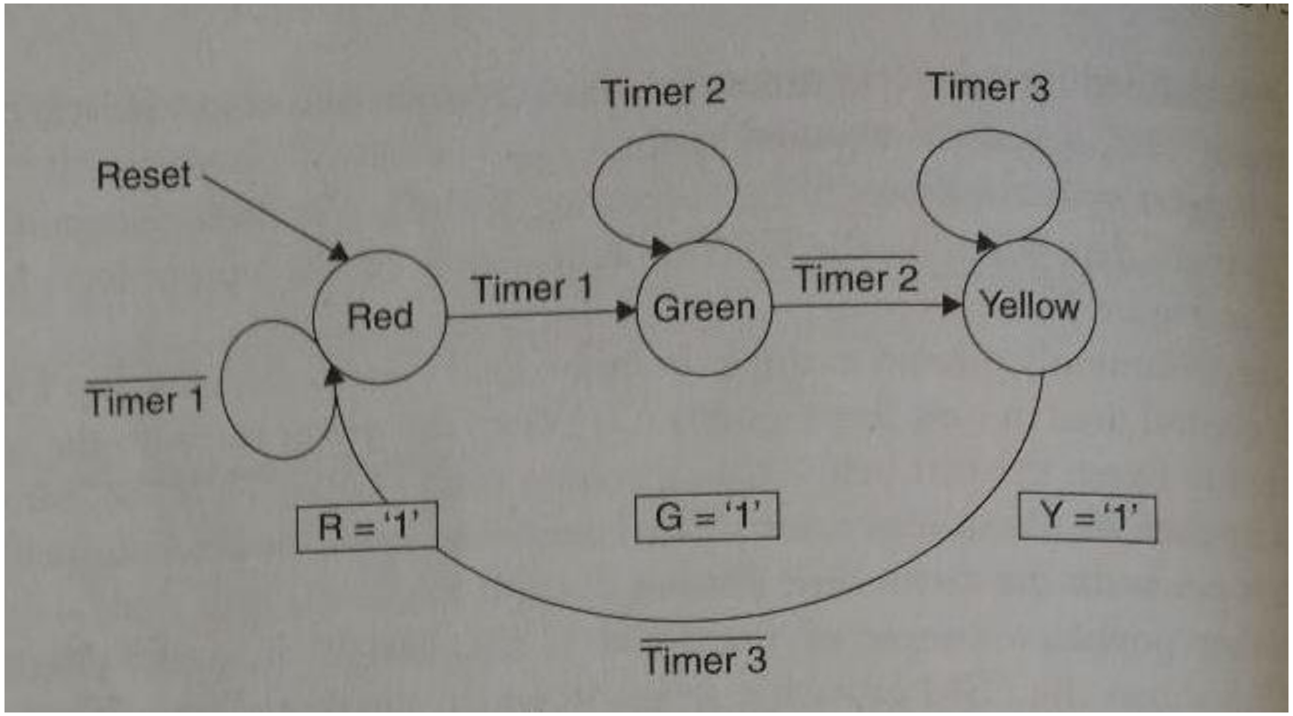
iii. Finite state Machine

- The idea of state machine is that on occurrence of event, the state of machine changes.
- FSM is defined by three quantities input, state and output. It is change in input which function as required event .
- For bigger systems, there is output logic block for each change of state the output changes depending on function represented by output logic block.

Example 1(Seat belt control used in Cars)



Example2(Traffic Light Control)


Timer 2
Timer 3
Reset
Red
Timer 1
Green
$\overline{\text{Timer 2}}$
Yellow
$\overline{\text{Timer 1}}$
R = '1'
G = '1'
Y = '1'
$\overline{\text{Timer 3}}$

Embedded Product Development Lifecycle Management

- Consider an example of building an house.
- Flow of phases will be as shown in fig.

Different phases involved in building a house

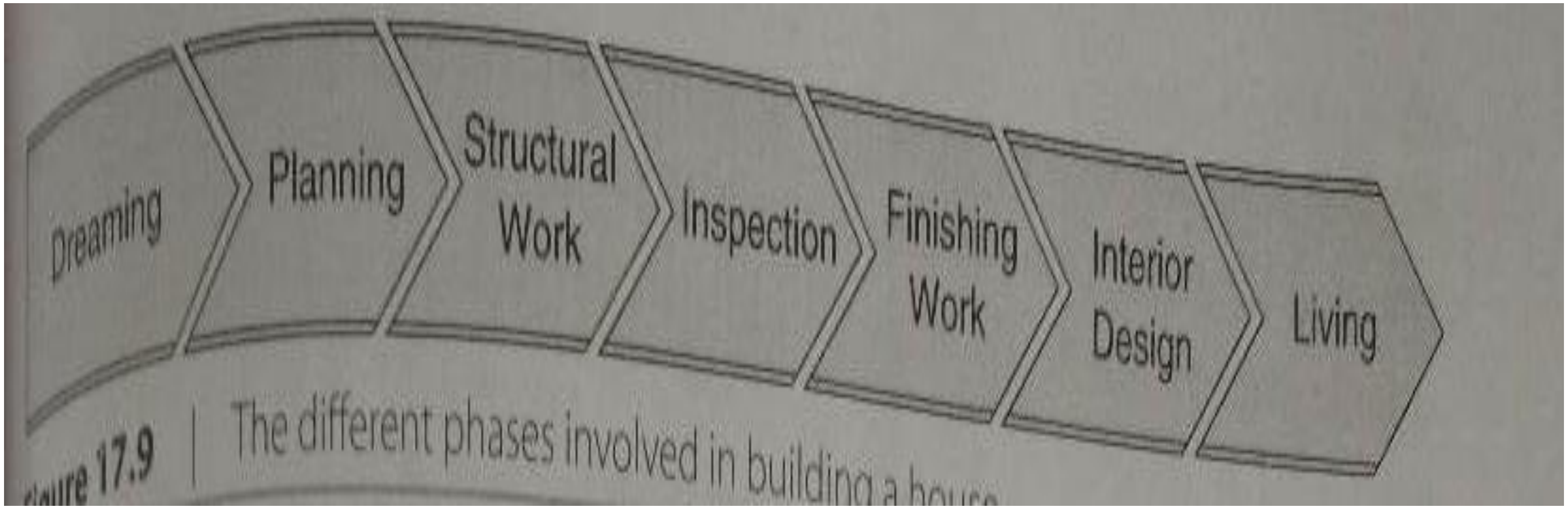


Figure 17.9 | The different phases involved in building a house

Embedded Product Development

- Compare the phases of house building with with phases of building an embedded product.
- Steps are not same but mechanism of planning and development work here also

Big Bang Model

- Developer receives list of requirements
- Developer works in isolation for some time
- Developer delivers the result to customer
- Developer hopes the client is satisfied

But developing a sophisticated embedded product is not done by just one person.

There are hardware and software developers, managers etc.

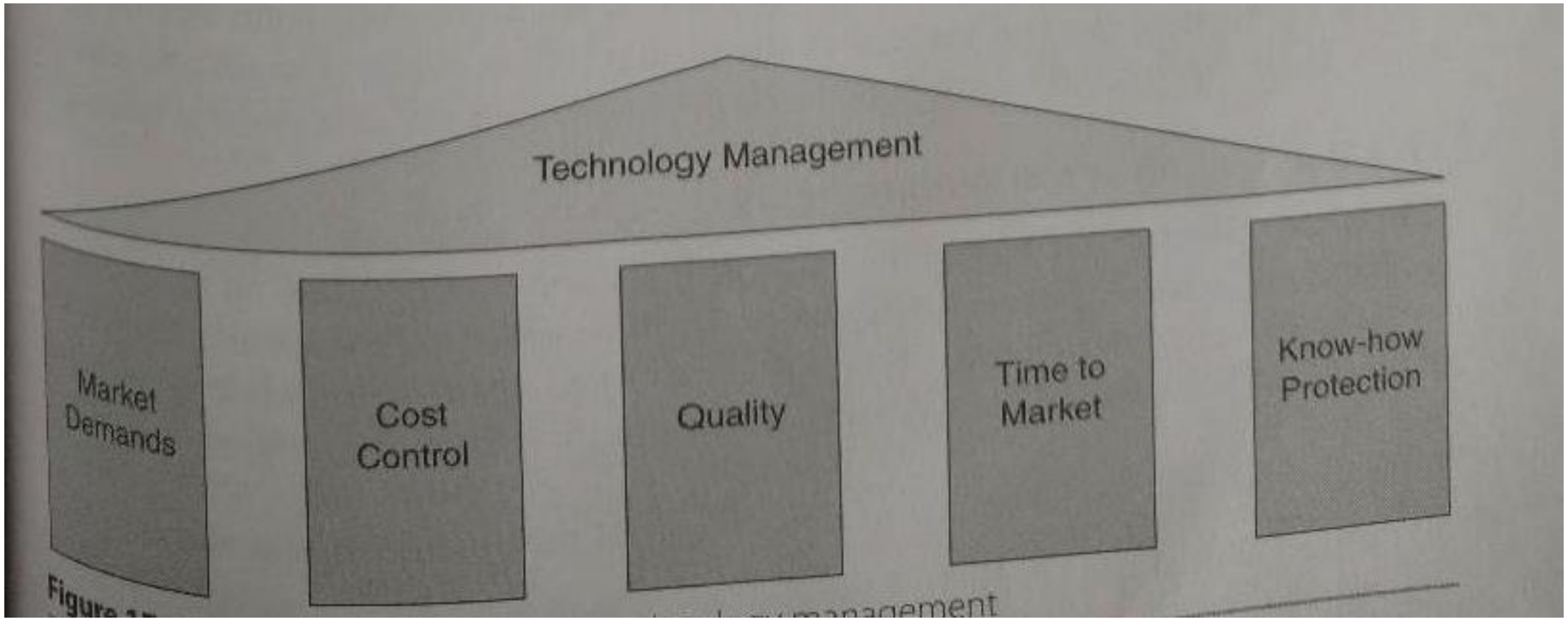
Why is EDLC Management considered to be important?

- In early phases of embedded revolution the embedded product was not really a ‘big deal’.
- The product was usually small and did not involve high-end technology.
- Now these products are developed by many Well-established companies usually have a definite action plan and activities are done from beginning to end.
- How well the action plan and activities are done, reflect on final product

Why is EDLC Management considered to be important?

- Better product from better manufacturers- they follow best practices for product development.
- There are bench marks using which Companies are rating.
- Companies having better ratings and deliver better product because they follow model for product development

Important factors in technology Management



Phases of EDLC

1. Innovate
2. Analyse
3. Design
4. Implement
5. Verify
6. Deploy
7. Support
8. Retire

Innovate/Conceptualize

- When **idea of product takes shape**
- Phase of dreaming when a company wonders whether it can introduce **an entirely new product or**
- **‘me too’ product-product** already available in market, but company is introducing its new version of product

Analyse

- Phase of analysing the requirements and specifications of intended product

Formally First stage . In this steps are

i. Make a formal and thorough feasibility study including investment and return expected.
ii. Make a list of requirements of product including its specifications (electrical, mechanical, packaging, appearance and use interface)
iii. Analyse requirements and make sure they can be implemented practically.

Design the intended product

i. Propose a prototype
ii. Partition the prototype design into hardware and software
iii. Do functional modelling for the hardware and software

Implement the Design

- Choose hardware components
- Design the hardware
- Generate the code and test it
- Integrate the hardware and software and test it

Perform Verification

System level verification

The integrated system is verified against the requirements made in phase ii to confirm that they match

Formalize and Deploy

- In this phase, field trials, reliability and quality tests are conducted.
- after these are completed the product is deployed in the market
- If product was built for specific customer, it is supplied to customer after training suitable no. of persons
- Most critical Phase—for companies with proven track records have an easier time in this phase

Sustain and support

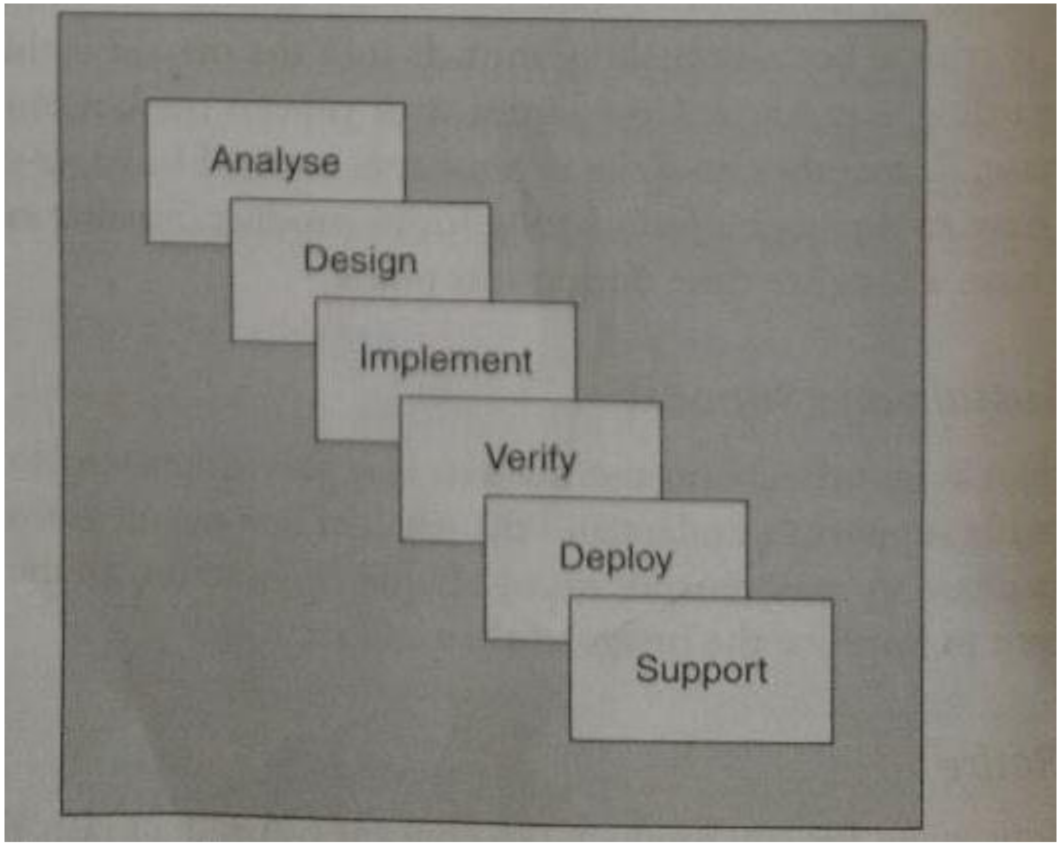
- Once the product is launched in the market, it is important to collect feedback from users, provide support to understand the product better and give solution for bugs detected in this phase
- A maintenance group should be in action all time along with marketing team to improve the image of the product

Retire

- No product or concept lasts for ever
- In today’s dynamic world, rapid obsolescence of electronic products is rule rather than the expectation
- Company should be ready to allow retirement for the product before it dies a natural death
- In either case there is possibility of rebirth
- Manufacturer can give a rebirth to product in a new, fresh and advanced form with new ideas, concept and features

Lifecycle Models

- Water Fall Model



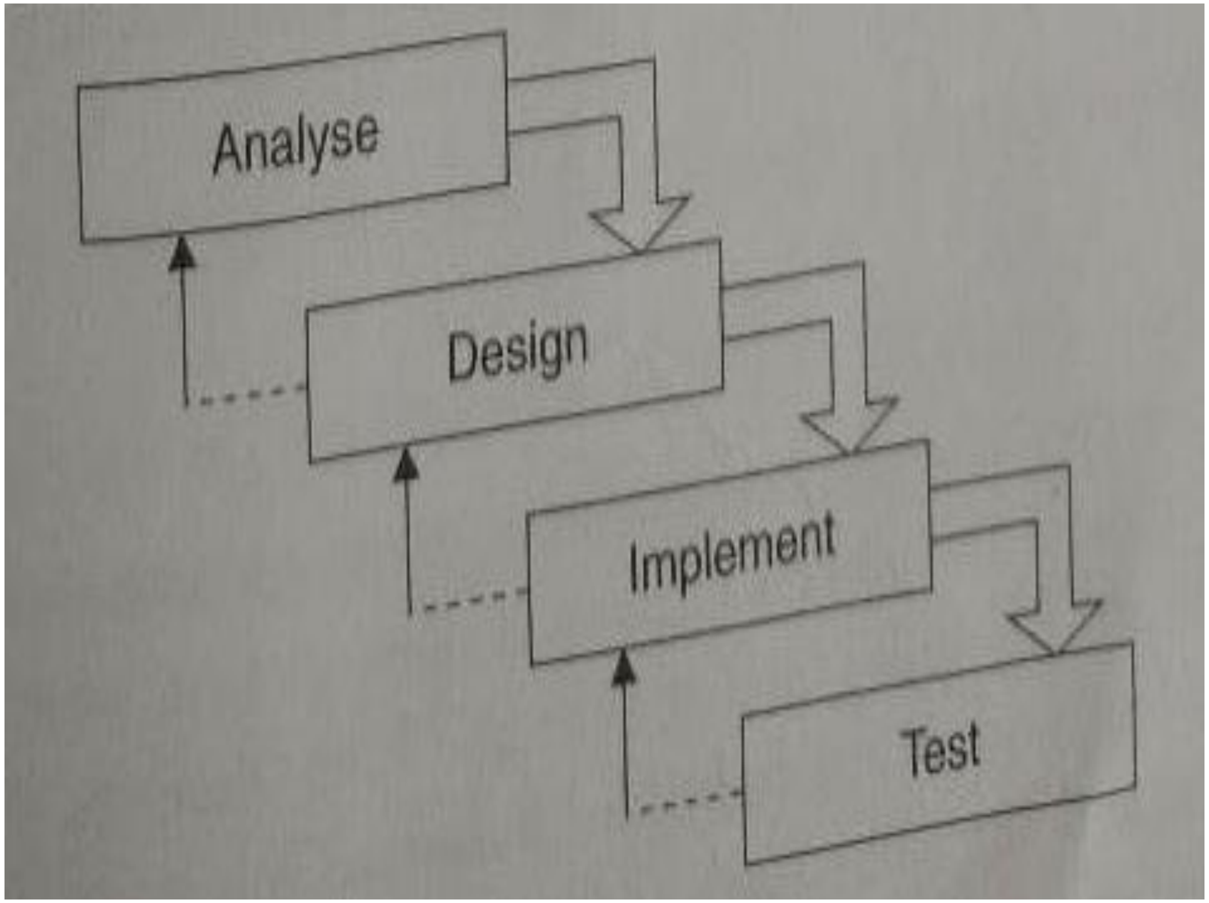
Water Fall Model

- Based on Serialization of development phases
- One phase flows into next

Demerits

- All requirements must be known before hand, as there is no phase which goes back and makes a correction
- There is no opportunity for customer to preview and correct the system

Waterfall Model With backflow



Waterfall Model With backflow

- Each phase has a backflow
- Correction to immediate previous stage is possible based on the feedback at the end of each phase

V Model

- Each phase and corresponding test or validation
- Tests are conducted at end of each phase before proceeding to next phase

Advantages

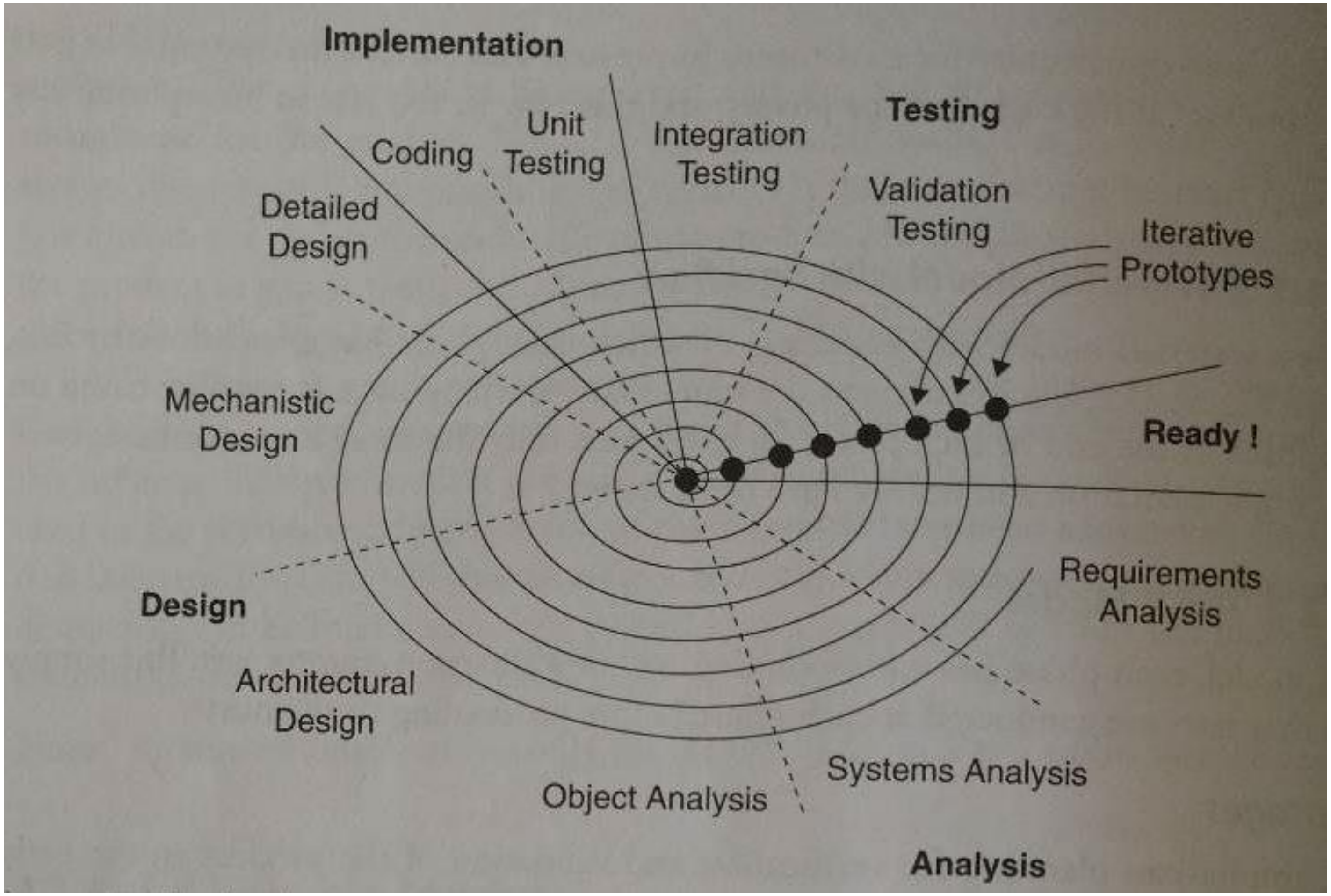
i. Verification and validation of product in early stages of development
ii. Deliverable at each stage is tested one
iii. Once each phase is tested and verified, it is marked as milestone in development process

V Model

Disadvantages

- Dies not easily handle dynamic changes in requirements
- Does not contain risk analysis activities

The Iterative Model



The Iterative Model

- Waterfall model executes many times and each iteration turn of wheel result in a prototype.
- Such in complete models are subject to testing, analysis and re-design.
- The design evolves iteratively.
- Such a model is flexible and risk management is automatically taken care of

Risk in this context

- The customer is not clear about what he want and keeps changing his requirements as project progresses.
- New trends and technology appears, and there is the need to incorporate these also in the product
- Designers and developers may be inexperienced in this domain