15 The Embedded Product Development Life Cycle (EDLC)

LEARNING OBJECTIVES

LO 1 Understand the life-cycle stages in embedded product development and the modelling of the life cycle stages

LO 2 Explain the project management and its objectives in embedded product development

LO 3 Discuss the objectives for modelling the life cycle
- Learn about the techniques for productivity improvement in embedded product development

LO 4 Know the different phases (Need, Conceptualisation, Analysis, Design, Development, Testing, Deployment, Support, Upgrades and Disposal) of the product development life cycle and the activities happening in each stage of the life cycle

LO 5 Identify the different modelling techniques for modelling the stages involved in the embedded product development life cycle
- Learn about the application, advantages and limitations of Linear/Waterfall Model in embedded product development life cycle management
- Learn about the application, advantages and limitations of Iterative/Incremental or Fountain Model in embedded product development life cycle management
- Learn about the application, advantages and limitations of prototyping/ evolutionary model in embedded product development life cycle management
- Learn about the application, advantages and limitations of Spiral Model in embedded product development life cycle management

Just imagine about your favourite chicken dish that your mom served during a Weekend lunch. Have you ever thought of the effort behind preparing the delicious chicken dish? Frankly speaking No, Isn’t it? Okay let’s go back and analyse how the yummy chicken in the chicken stall became the delicious chicken dish served on the dining table. One fine morning Mom thinks “Wohh!! It’s Sunday and why can’t we have a special lunch with our favourite chicken dish.” Mom tells this to Papa and he agrees on it and together they decide how much quantity of chicken is required for the four-member family (may be based on past experience) and lists out the various ingredients required for preparing the dish. Now the list is ready and papa checks his wallet to ensure whether the item list is at the reach of the money in his wallet. If not, both of them sit together and make some reductions in the list. Papa goes to the market to buy the items in the list. Finds out a chicken stall where quality chicken is selling at a nominal rate. Procures all the items and returns back home with all the items in the list and hands over the packet to Mom. Mom starts cleaning the chicken pieces and chopping the ingredient/vegetables and then puts them in a vessel and starts cooking them. Mom may go for preparing other side dishes or rice and comes back to the chicken dish preparation at regular intervals to add the necessary ingredients (like chicken masala, chilly powder, coriander powder, tamarind, coconut oil, salt, etc.) in time and checks whether all ingredients are in correct proportion, if not adjust them to the required proportion. Papa gives an overall supervision to the preparation and informs mom if she forgot to add any ingredients and also helps her in verifying the proportion of the ingredients. Finally the fragrance of spicy boiled chicken comes out of the vessel and mom realises that chicken dish is ready. She takes it out from the stove and adds the final taste building ingredients. Mom tastes a small piece from the dish and ensures that it meets the regular quality. Now the dish is ready to serve. Mom takes the responsibility of serving the same to you and what you need to do is taste it and tell her how delicious it is. If you closely observe these steps with an embedded product developer's mind, you can find out that this simple chicken dish preparation contains various complex activities like overall management, development, testing and releasing. Papa is the person who did the entire management activity starting from item procurement to giving overall supervision. Mom is responsible for developing the dish, testing the dish to certain extent and serving the dish (release management) and finally you are the one who did the actual field test. All embedded products will have a design and development life cycle similar to our chicken dish preparation example, with management, design, development, test and release activities.

15.1 WHAT IS EDLC?

LO 1 Understand the life-cycle stages in embedded product development and the modelling of the life cycle stages

Embedded Product Development Life Cycle (Let us call it as EDLC, though it is not a standard and universal term) is an 'Analysis-Design-Implementation' based standard problem solving approach for Embedded Product Development. In any product development application, the first and foremost step is to figure out what product needs to be developed (analysis), next you need to figure out a good approach for building it (design) and last but not least you need to develop it (implementation).

15.2 WHY EDLC

LO 2 Explain the project management and its objectives in embedded product development

EDLC is essential for understanding 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 and development (hardware, firmware and enclosure design and development), system testing, release management and quality assurance. The standards imposed by EDLC on a product development makes the product, developer independent in terms of standard documents and it also provides uniformity in development approaches.

15.3 OBJECTIVES OF EDLC

LO 3 Discuss the objectives for modelling the life cycle

The ultimate aim of any embedded product in a commercial production setup is to produce marginal benefit. Marginal benefit is usually expressed in terms of Return on Investment (ROI). The investment for a product development includes initial investment, manpower investment, infrastructure investment, supply chain and marketing expenses, etc. A product is said to be profitable only if the turnover from the selling of the product is more than that of the overall investment expenditure. For this, the product should be acceptable by the end user and it should meet the requirements of end user in terms of quality, reliability and functionality. So it is very essential to ensure that the product is meeting all these criteria, throughout the design, development, implementation and support phases. Embedded Product Development Life Cycle (EDLC) helps out in ensuring all these requirements. EDLC has three primary objectives, namely

1. Ensure that high quality products are delivered to end user.
2. Risk minimisation and defect prevention in product development through project management.
3. Maximise the productivity.

15.3.1 Ensuring High Quality for Products

The primary definition of quality in any embedded product development is the Return on Investment (ROI) achieved by the product. The expenses incurred for developing the product may fall in any of the categories; initial investment, developer recruiting, training, or any other infrastructure requirement related. There will be some budgetary and cost allocation given to the development of the product and it will be allocated by some higher officials based on the assumption that the product will produce a good return or a return justifying the investment. The budget allocation might have done after studying the market trends and requirements of the product, competition, etc. EDLC must ensure that the development of the product has taken account of all the qualitative attributes of the embedded system. The qualitative attributes are discussed separately in an earlier chapter of this book. Please refer back for the same.

15.3.2 Risk Minimisation and Defect Prevention through Management

You may be thinking what is the significance of project management, or why project management is essential in product development. Nevertheless it is an additional expenditure to the project! If we look back to the chicken dish example, we can find out that the management activity from dad is essential in the beginning phase but in the preparation phase it can be handled by mom itself. There are projects in embedded product development which requires 'loose' or 'tight' project management. If the product development project is a simple one, a senior developer itself can take charge of the management activity and no need for a skilled project manager to look after this with dedicated effort throughout the development process, but there should be an overall supervision from a skilled project management team for ensuring that the development process is going in the right direction. Projects which are complex and requires timeliness should have a dedicated and skilled project management part and hence they are said to be "tightly" bounded to project management. '*Project management is essential for predictability, co-ordination and risk minimisation*'. Whenever a product development request comes, an estimate on the duration of the development and deployment activity should be given to the end user/client. The timeframe may be expressed in number of person days PDS (The effort in terms of single person working for this much days) or 'X person for X week (e.g. 2 person 2 week) etc. This is one aspect of predictability. The management team might have reached on this estimate based on past experience in handling similar project or on the analysis of work summary or data available for a similar project, which was logged in using an activity tool. Resource (Developer) allocation is another aspect of predictability in management. Resource allocations like how many resources should work for this project for how many days, how many resources are critical with respect to the work handling by them and how many backups required for the resources to overcome a critical situation where a resource is not available (Risk minimisation). Resource allocation is critical and it has a direct impact on investment. The communication aspect of the project management deals with co-ordination and interaction among partners, suppliers and client from which the request for the product development aroused. 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. Without management, the development work may go beyond the stipulated time frame (schedule slipping) and may end up in a product which is not meeting the requirements from the client side, as a result re-works should be done to rectify the possible deviations occurred and it will again put extra cost on the development. Project management makes the proverb "A stitch in time saves nine" meaningful in an embedded product development. Apart from resource allocation, project management also covers activities like task allocation, scheduling, monitoring and project tracking. Computer Assisted Software Engineering (CASE) Tools and Gantt charts help the manager in achieving this. Microsoft® Project Tool is a typical example of CASE tool for project management.

15.3.3 Increased Productivity

Productivity is a measure of efficiency as well as Return on Investment (ROI). One aspect of productivity covers how many resources are utilised to build the product, how much investment required, how much time is taken for developing the product, etc. For example, the productivity of a system is said to be doubled if a product developed by a team of 'X' members in a period of 'X' days is developed by another team of 'X/2' members in a period of 'X' days or by a team of 'X' members in a period of 'X/2' days. This productivity measurement is based on total manpower efficiency. Productivity in terms of Returns is said to be increased, if the product is capable of yielding maximum returns with reduced investment. Saving manpower effort will definitely result in increased productivity. Usage of automated tools, wherever possible, is recommended for this. The initial investment on tools may be an additional burden in terms of money, but it will definitely save efforts in the next project also. It is a one-time investment. "Pay once use many time". Another important factor which can help in increased productivity is "re-usable effort". Some of the works required for the current product development may have some common features which you built for some of the other product development in the projects you executed before. Identify those efforts and design the new product in such a way that it can directly be plugged into the new product without any additional effort. (For example, the current product you are developing requires an RS-232C USB serial interface and one of the product you already developed have the same feature. Adapt this part directly from the existing product in terms of the hardware and firmware required for the same.) This will definitely increase the productivity by reducing the development effort. Another advised method for increasing the productivity is by using resources with specific skill sets which matches the exact requirement of the entire or part of the product (e.g. Resource with expertise in Bluetooth technology for developing a Bluetooth interface for the product). This reduces the learning time taken by a resource, who does not have prior expertise in the particular feature or domain. This is not possible in all product development environments, since some of the resources with the desired skill sets may be engaged with some other work and releasing them from the current work is not possible. Recruiting people with desired skill sets for the current product development is another option; this is worth only if you expect to have more work on the near future on the same technology or skill sets. Use of Commercial Off-the-Shelf Components (COTS) wherever possible in a product is a very good way of reducing the development effort and thereby increasing the productivity (Refer back to the topic 'Commercial Off-the-shelf components' given in Chapter 2 for more details on COTS). COTS component is a ready to use component and you can use the same as plug-in modules in your product. For example, if the product under development requires a 10 base T Ethernet connectivity, you can either implement the same in your product by using the TCP/IP chip and related components or can use a readily available TCP/IP full functional plug-in module. The second approach will save effort and time. EDLC should take all these aspects into consideration to provide maximum productivity.

15.4 DIFFERENT PHASES OF EDLC

LO 4 Know the different phases (Need, Conceptualisation, Analysis, Design, Development, Testing, Deployment, Support, Upgrades and Disposal) of the product development life cycle and the activities happening in each stage of the life cycle

The life cycle of a product development is commonly referred to as models. Model 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. A typical simple product contains five minimal phases namely: 'Requirement Analysis', 'Design', 'Development and Test', 'Deployment' and 'Maintenance'. The classic Embedded Product Life Cycle Model contains the phases: 'Need', 'Conceptualisation', 'Analysis', 'Design', 'Development and Testing', 'Deployment', 'Support', 'Upgrades' and 'Retirement/Disposal' (Fig. 15.1). In a real product development environment, the phases given in this model may vary and can have submodels or the models contain only certain important phases of the classic model. As mentioned earlier, the number of phases involved in the EDLC model depends on the complexity of the product. The following section describes each phases of the classic EDLC model in detail.

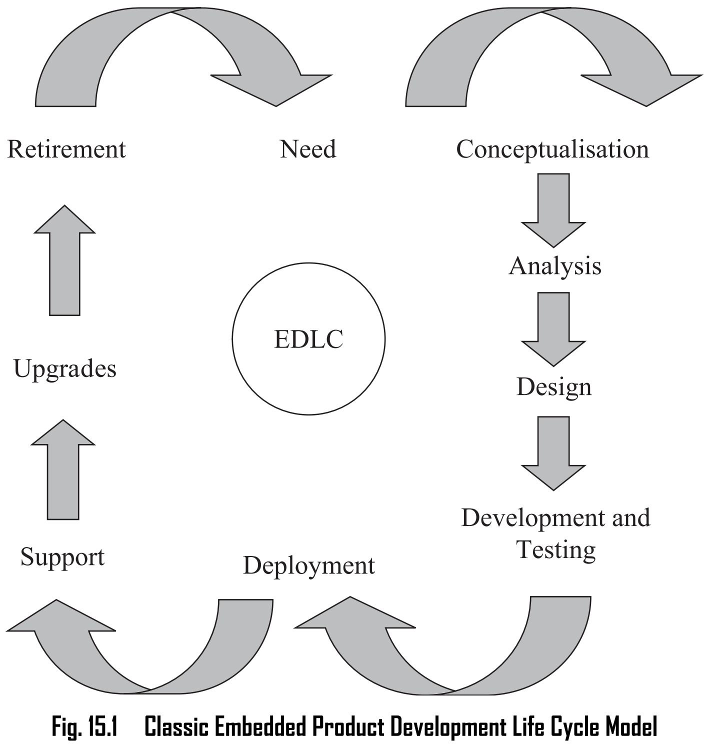

Fig. 15.1 Classic Embedded Product Development Life Cycle Model

15.4.1 Need

Any embedded product evolves as an output of a '*Need*'. The need may come from an individual or from the public or from a company (Generally speaking from an end user/client). 'Need' should be articulated to initiate the Product Development 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. Once the proposal gets approval, it goes to a product development team, which can either be an organisation from which the need arise or a third party product development/service company (If a product development company approaches a client/sponsor with the idea of a product, the business needs for the product will be prepared by the company itself). The product development need can be visualised in any one of the following three needs.

15.4.1.1 New or Custom Product Development

The need for a product which does not exist in the market or a product which acts as competitor to an existing product in the current market will lead to the development of a completely new product. The product can be a commercial requirement or an individual requirement or a specific organisation's requirement. Example for an Embedded Product which acts a competitor in the current market is 'Mobile handset' which is going to be developed by a new company apart from the existing manufacturers. A PDA tailored for any specific need is an example for a custom product.

15.4.1.2 Product Re-engineering

The embedded product market is dynamic and competitive. In order to sustain in the market, the product developer should be aware of the general market trends, technology changes and the taste of the end user and should constantly improve an existing product by incorporating new technologies and design changes for performance, functionalities and aesthetics. 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

15.4.1.3 Product Maintenance

Product maintenance 'need' deals with providing technical support to the end user for an existing product in the market. The maintenance request may come as a result of product non-functioning or failure. Product maintenance is generally classified into two categories; Corrective maintenance and Preventive maintenance. Corrective maintenance deals with making corrective actions following a failure or non-functioning. The failure can occur due to damages or non-functioning of components/parts of the product and the corrective maintenance replaces them to make the product functional again. Preventive maintenance is the scheduled maintenance to avoid the failure or non-functioning of the product.

15.4.2 Conceptualisation

Conceptualisation is the 'Product Concept Development Phase' and it begins immediately after a Concept Proposal is formally approved. Conceptualisation phase defines the scope of the concept, performs cost benefit analysis and feasibility study and prepares project management and risk management plans. Conceptualisation can be viewed as a phase shaping the "Need" of an end-user or convincing the end user, whether it is a feasible product and how this product can be realised (Fig. 15.2).

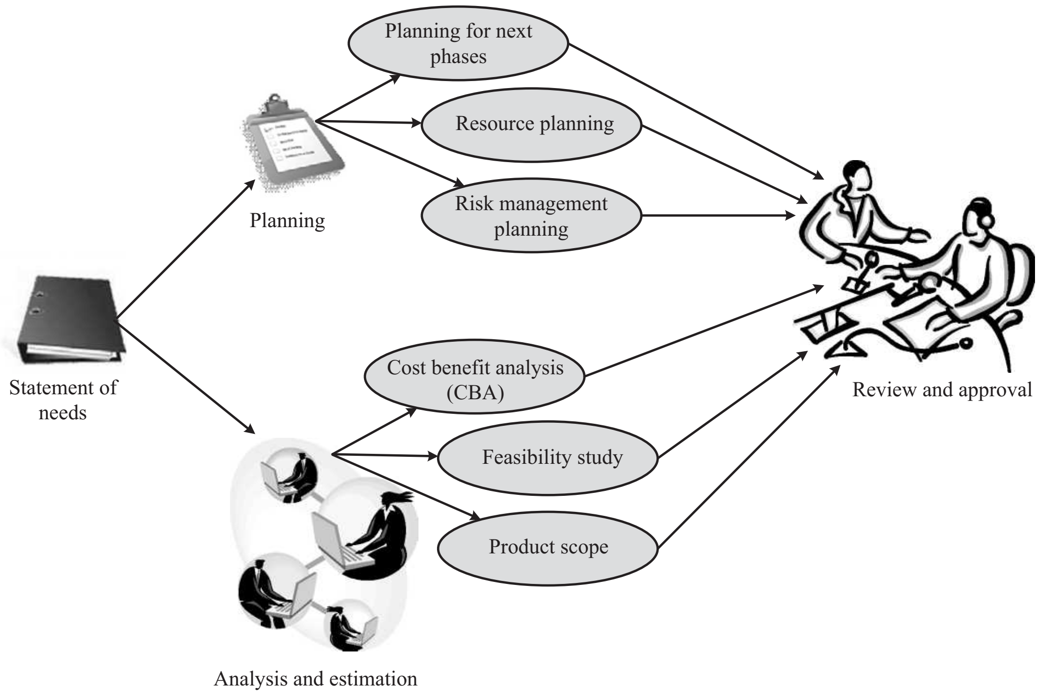

Fig. 15.2 Various activities involved in Conceptualisation Phase

The 'Conceptualisation' phase involves two types of activities, namely; 'Planning Activity' and 'Analysis and Study Activity'. The Analysis and Study Activities are performed to understand the opportunity of the product in the market (for a commercial product). The following are the important 'Analysis and Study activities' performed during 'Conceptualisation Phase'.

15.4.2.1 Feasibility Study

Feasibility study examines the need for the product carefully and suggests one or more possible solutions to build the need as a product along with alternatives. Feasibility study analyses the Technical (Technical Challenges, limitations, etc.) as well as financial feasibility (Financial requirements, funding, etc.) of the product under consideration.

15.4.2.2 Cost Benefit 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. It is somewhat similar to the loss-gain analysis in business term except that CBA is an assumption oriented approach based on some rule of thumb or based on the analysis of data available for similar products developed in the past. In summary, CBA estimates and totals up the equivalent money value of the benefits and costs of the product under consideration and give an idea about the worthwhile of the product. Some common underlying principles of Cost Benefit Analysis are illustrated below.

Common Unit of Measurement In order to make the analysis sensible and meaningful, all aspects of the product, either plus points or minus points should be expressed in terms of a common unit. Since the ultimate aim of a product is “Marginal Profit” and the marginal profit is expressed in terms of money in most cases, ‘money’ is taken as the common unit of measurement. Hence all benefits and costs of the product should be expressed in terms of money. Money in turn may be represented by a common currency. It can be Indian Rupee (INR) or US Dollar (USD), etc.

Market Choice based Benefit Measurement Ensure that the product cost is justifying the money (value for money), the end user is spending on the product to purchase, for a commercial product.

Targeted End Users For a commercial product it is very essential to understand the targeted end-users for the product and their tastes to give them the best in the industry.

15.4.2.3 Product Scope

Product scope deals with what is in scope (what all functionalities or tasks are considered) for the product and what is not in scope (what all functionalities or tasks are not considered) for the product. Product scope should be documented properly for future reference.

15.4.2.4 Planning Activities

The planning activity covers various plans required for the product development, like the plan and schedule for executing the next phases following conceptualisation, **Resource Planning** – How many resources should work on the project, **Risk Management Plans** – The technical and other kinds of risks involved in the work and what are the mitigation plans, etc. At the end of the conceptualisation phase, the reports on ‘Analysis and Study Activities’ and ‘Planning Activities’ are submitted to the client/project sponsor for review and approval, along with any one of the following recommendation.

1. The product is feasible; proceed to the next phase of the product life cycle.
2. The product is not feasible; scrap the project

The executive committee, to which the various reports are submitted, will review the same and will communicate the review comments to the officials submitted the conceptualisation reports and will give the green signal to proceed, if they are satisfied with the analysis and estimates.

15.4.3 Analysis

Requirements Analysis Phase starts immediately after the documents submitted during the ‘Conceptualisation’ phase is approved by the client/sponsor of the project. Documentation related to user requirements from the Conceptualisation phase is used as the basis for further user need analysis and the development of detailed user requirements. Requirement analysis is performed to develop a detailed functional model of the product under consideration. During the Requirements Analysis Phase, the product is defined in 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 illustrated in Fig. 15.3.

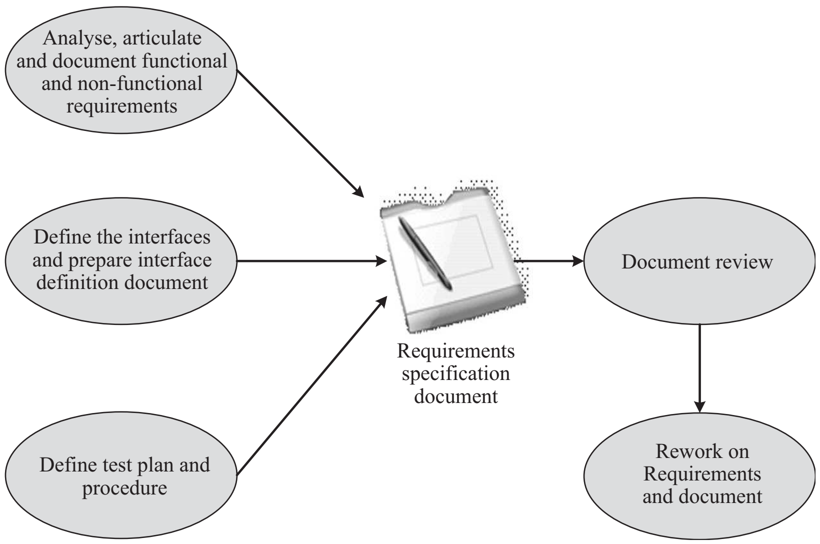
Analyse, articulate and document functional and non-functional requirements
Define the interfaces and prepare interface definition document
Define test plan and procedure
Requirements specification document
Document review
Rework on Requirements and document

Fig. 15.3 Various activities involved in Requirements Analysis Phase

15.4.3.1 Analysis and Documentation

The analysis and documentation activity consolidates the business needs of the product under development and analyses the purpose of the product. It addresses various functional aspects (product features and functions) and quality attributes (non-functional requirements) of the product, defines the functional and data requirements and connects the functional requirements with the data requirements. In summary, the analysis activities address all business functionalities of the product and develop a logical model describing the fundamental processes and the data required to support the functionalities. The various requirements that need to be addressed during the requirement analysis phase for a product under consideration are listed below.

1. Functional capabilities like performance, operational characteristics (Refer to Chapter 3 for details on operational characteristics), etc.
2. Operational and non-operational quality attributes (Refer to Chapter 3 for Quality attributes of embedded products)
3. Product external interface requirements
4. Data requirements
5. User manuals
6. Operational requirements
7. Maintenance requirements
8. General assumptions

15.4.3.2 Interface Definition and Documentation

The embedded product under consideration may be a standalone product or it can be a part of a large distributed system. If it is a part of another system there should be some interface between the product and the other parts of the distributed system. Even in the case of standalone products there will be some standard interfaces like serial, parallel etc as a provision to connect to some standard interfaces. The interface definition and documentation activity should clearly analyse on the physical interface (type of interface) as well as data exchange through these interfaces and should document it. It is essential for the proper information flow throughout the life cycle.

15.4.3.3 Defining Test Plan and Procedures

Identifies what kind of tests are to be performed and what all should be covered in testing the product. Define the test procedures, test setup and test environment. Create a master test plan to cover all functional as well as other aspects of the product and document the scope, methodology, sequence and management of all kinds of tests. It is essential for ensuring the total quality of the product. The various types of testing performed in a product development are listed below.

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/product requirements of the product after integration. System testing refers to a set of different tests and a few among them are listed below.
   - **Usability testing**—Tests the usability of the product
   - **Load testing**—Tests the behaviour of the product under different loading conditions.
   - **Security testing**—Testing the security aspects of the product
   - **Scalability testing**—Testing the scalability aspect of the product
   - **Sanity testing**—Superficial testing performed to ensure that the product is functioning properly
   - **Smoke testing**—Non exhaustive test to ensure that the crucial requirements for the product are functioning properly. This test is not giving importance to the finer details of different functionalities
   - **Performance testing**—Testing the performance aspects of the product after integration
   - **Endurance testing**—Durability test of the product
4. **User acceptance testing**—Testing the product to ensure it is meeting all requirements of the end user.

At the end, all requirements should be put in a template suggested by the standard (e.g. ISO-9001) Process Model (e.g. CMM Level 5) followed for the Quality Assurance. This document is referred as 'Requirements Specification Document'. It is sent out for review by the client and the client, after reviewing the requirements specification document, communicates back with the review comments. According to the review comments more requirement analysis may have to be performed and re-work required is performed on the initial 'Requirement Specification Document'.

15.4.4 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. The design phase identifies the application environment and creates an overall architecture for the product. Product design starts with '*Preliminary Design/High Level Design*'. Preliminary design establishes the top-level architecture for the product, lists out the various functional blocks required for the product, and defines the inputs and outputs for each functional block. 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/client who come up with a need for the product. If the end-user agrees on the preliminary design, the product design team can take the work to the next level – '*Detailed Design*'. Detailed design generates a detailed architecture, identifies and lists out the various components for each functional block, the inter connection among various functional blocks, the control algorithm requirements, etc. The 'Detailed Design' also needs to be reviewed and get approved by the end user/customer. If the end user wants modifications on the design, it can be informed to the design team through review comments.

An embedded product is a real mix of hardware, embedded firmware and enclosure. Hence both *Preliminary Design* and *Detailed Design* should take the design of these into consideration. For traditional embedded system design approach, the functional requirements are classified into either hardware or firmware at an early stage of the design and the hardware team works on the design of the required hardware, whereas the software team works on the design of the embedded firmware. Today the scenario is totally changed and a hardware software co-design approach is used. In this approach, the functional requirements are not broken down into hardware and software, instead they are treated as system requirements and the partitioning of system requirements into either hardware or software is performed at a later stage of the design based on hardware-software trade-offs for implementing the required functionalities. The hardware and software requirements are modelled using different modelling techniques. Please refer to Chapter 7 for more details on hardware-software co-design and program models used in hardware-software co-design. The other activities performed during the design phase are 'Design of Operations and maintenance manual' and 'Design of Training materials' (Fig. 15.4). The operations manual is necessary for educating the end user on 'how to operate the product' and the maintenance manual is essential for providing information on how to handle the product in situations like non-functioning, failure, etc.

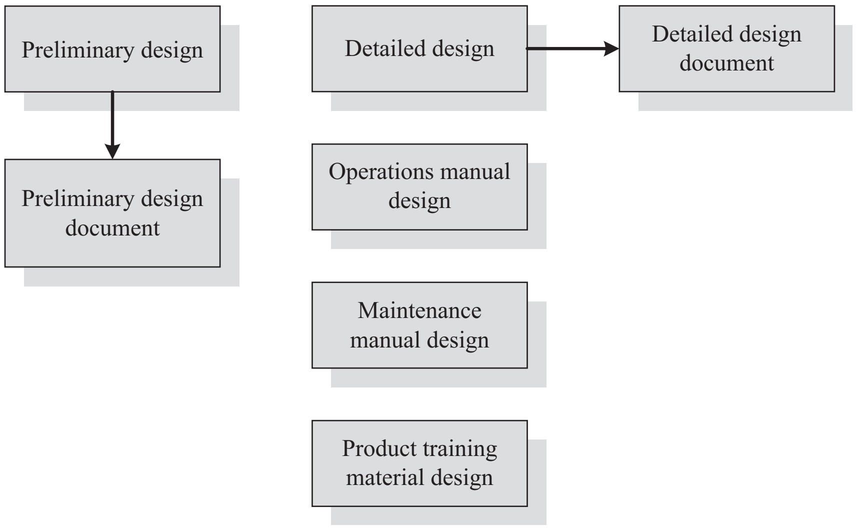
Preliminary design
Detailed design
Detailed design document
Preliminary design document
Operations manual design
Maintenance manual design
Product training material design

Fig. 15.4 Various Activities involved in Design Phase

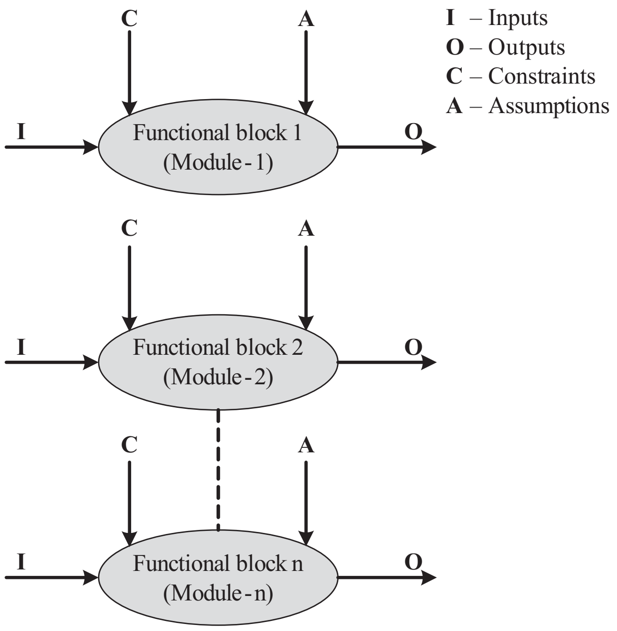
C
A
I – Inputs
O – Outputs
C – Constraints
A – Assumptions
I
Functional block 1 (Module-1)
O
C
A
I
Functional block 2 (Module-2)
O
C
A
I
Functional block n (Module-n)
O

Fig. 15.5 Preliminary Design Illustration

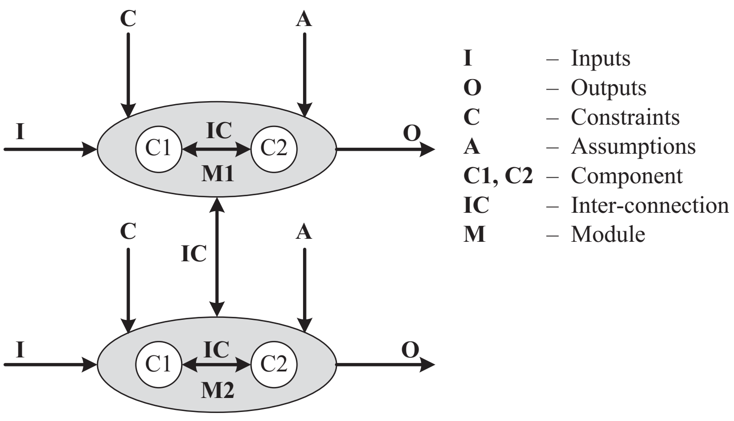

Fig. 15.6 Detailed Design Illustration

15.4.5 Development and Testing

The 'Development Phase' transforms the design into a realisable product. For this the detailed specifications generated during the design phase are translated into hardware and firmware. During development phase, the installation and setting up of various development tools is performed and the product hardware and firmware is developed using different tools and associated production setup. The development activities can be partitioned 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. Embedded firmware development is performed using the embedded firmware development tools. The mechanical enclosure design is performed with CAD Tools like Solid Works, AutoCAD, etc. "Look and Feel" of a commercial product plays an important role on its market demand. So the entire product design and development should concentrate on the product aesthetics (Size, Weight, Appearance, etc.) and special requirements like protection shielding, etc.

Component procurement also carried out during this phase. As mentioned in one of the earlier chapters, some of the components may have "*lead time* – The time required for the component to deliver after placing the purchase order for the component" and the component procurement should be planned well in advance to avoid the possible delay in the product development. The other important activities carried out during this phase are preparation of test case procedures, preparation of test files and development of detailed user, deployment, operations and maintenance manual. The embedded firmware development may be divided into various modules and the firmware (code) review for each module as well as the schematic diagram and layout file review for hardware is carried out during the development phase and the changes required should be incorporated in the development from time to time.

The testing phase can be divided into independent testing of firmware and hardware (Unit Testing), testing of the product after integrating the firmware and hardware (Integration Testing), testing of the whole system on a functionality and non-functionality basis (System Testing) and testing of the product against all acceptance criteria mentioned by the client/end user for each functionality (User Acceptance Testing). The Unit testing deals with testing the embedded hardware and its associated modules, the different modules of the firmware for their functionality independently. A test plan is prepared for the unit testing (Unit Test plan) and test cases (Unit Test cases) are identified for testing the functionality of the product as per the design. Unit test is normally performed by the hardware developer or firmware developer as part of the development phase. Depending on the product complexity, rest of the tests can be considered as the activities performed in the 'Testing phase' of the product. Once the hardware and firmware are unit tested, they are integrated using various firmware integration techniques and the integrated product is tested against the required functionalities. An integration test plan is prepared for this and the test cases for integration testing (Integration test cases) is developed. The Integration testing ensures that the functionality which is tested working properly in an independent mode remains working properly in an integrated product (Hardware + Firmware). The integration test results are logged in and the firmware/hardware is reworked against any flaws detected during the Integration testing. The purpose of integration testing is to detect any inconsistencies between the firmware modules units that are integrated together or between any of the firmware modules and the embedded hardware. Integration testing of various modules, hardware and firmware is done at the end of development phase depending on the readiness of hardware and firmware.

The system testing follows the Integration testing and it is a set of various tests for functional and non-functional requirements verification. System testing is a kind of black box testing, which doesn't require any knowledge on the inner design logic or code for the product. System testing evaluates the product's compliance with the requirements. User acceptance test or simply Acceptance testing is the product evaluation performed by the customer as a condition of the product purchase. During user Acceptance testing, the product is tested against the acceptance value set by customer/end user for each requirement (e.g. The loading time for the application running on the PDA device is less than 1 millisecond ☺). The deliverables from the 'Design and Testing' phase are firmware source code, firmware binaries, finished hardware, various test plans (Hardware and Firmware Unit Test plans, Integration Test plan, System Test plan and Acceptance Test plan), test cases (Hardware and Firmware Unit Test cases, Integration Test cases, System Test cases and Acceptance Test cases) and test reports (Hardware and Firmware Unit Test Report, Integration Test Report, System Test Report and Acceptance Test Report).

15.4.6 Deployment

Deployment is the process of launching the first fully functional model of the product in the market (for a commercial embedded product) or handing over the fully functional initial model to an end user/client. It is also known as *First Customer Shipping* (*FCS*). During this phase, the product modifications as per the various integration tests are implemented and the product is made operational in a production environment. The 'Deployment Phase' is initiated after the system is tested and accepted (User Acceptance Test) by the end user. The important tasks performed during the Deployment Phase are listed below.

15.4.6.1 Notification of Product Deployment

Whenever the product is ready to launch in the market, the launching ceremony details should be communicated to the stake holders (all those who are related to the product in a direct or indirect way) and to the public if it is a commercial product. The notifications can be sent out through e-mail, media, etc. mentioning the following in a few words.

1. Deployment Schedule (Date Time and Venue)
2. Brief description about the product
3. Targeted end users
4. Extra features supported with respect to an existing product (for upgrades of existing model and new products having other competitive products in the market)
5. Product support information including the support person name, contact number, contact mail ID, etc.

15.4.6.2 Execution of Training Plan

Proper training should be given to the end user to get them acquainted with the new product. Before launching the product, conduct proper training as per the training plan developed during the earlier phases. Proper training will help in reducing the possible damages to the product as well as the operating person, including personal injuries and product mal-functioning due to inappropriate usage. User manual will help in understanding the product, its usage and accessing its functionalities to certain extend.

15.4.6.3 Product Installation

Instal the product as per the installation document to ensure that it is fully functional. As a typical example take the case of a newly purchased mobile handset. The user manual accompanying it will tell you clearly how to instal the battery, how to charge the battery, how many hours the battery needs to be charged, how the handset can be switched on/off, etc.

15.4.6.4 Product Post-Implementation Review

Once the product is launched in the market, conduct a post-implementation review to determine the success of the product. The ultimate aim behind post-implementation review is to document the problems faced during installation and the solutions adopted to overcome them which will act as a reference document for future product development. It also helps in understanding the customer needs and the expectations of the customer on the next version of the product.

15.4.7 Support

The support phase deals with the operations and maintenance of the product in a production environment. During this phase all aspects of operations and maintenance of the product are covered and the product is scrutinised to ensure that it meets the requirements put forward by the end user/client. 'Bugs (product mal-functioning or unexpected behaviour or any operational error)' in the products may be observed and reported during the operations phase. Support should be provided to the end user/client to fix the bugs in the product. The support phase ensures that the product meets the user needs and it continues functioning in the production environment. The various activities involved in the 'Support' phase are listed below.

15.4.7.1 Set up a Dedicated Support Wing

The availability of certain embedded products in terms of product functioning in production environment is crucial and they may require 24x7 support in case of product failure or malfunctioning. For example the patient monitoring system used in hospitals is a very critical product in terms of functioning and any malfunctioning of the product requires immediate technical attention/support from the supplier. Set up a dedicated support wing (customer care unit) and ensure high quality service is delivered to the end user. 'After service' plays significant role in product movement in a commercial market. If the manufacturer fails to provide timely and quality service to the end user, it will naturally create the talk '*the product is good but the after service is very poor*' among the end users and people will refrain from buying such products and will opt for competitive products which provides more or less same functionalities and high quality service. The support wing should be set up in such a way that they are easily reachable through e-mail, phone, fax, etc.

15.4.7.2 Identify Bugs and Areas of Improvement

None of the products are bug-free. Even if you take utmost care in design, development and implementation of the product, there can be bugs in it and the bugs reveal their real identity when the conditions are favouring them, similar to the harmless microbes living in human body getting turned into harmful microbes when the body resistance is weak. You may miss out certain operating conditions while design or development phase and if the product faces such an operating condition, the product behaviour may become unexpected and it is addressed with the familiar nick name '*Bug*' by a software/product engineer. Since the embedded product market is highly competitive, it is very essential to stay connected with the end users and know their needs for the survival of the product. Give the end user a chance to express their views on the product and suggestions, if any, in terms of modifications required or feature enhancements (areas of improvement), through user feedbacks. Conduct product specific surveys and collect as much data as possible from the end user.

15.4.8 Upgrades

The upgrade phase of product development deals with the development of upgrades (new versions) for the product which is already present in the market. Product upgrade results as an output of major bug fixes or feature enhancement requirements from the end user. During the upgrade phase the system is subject to design modification to fix the major bugs reported or to incorporate the new feature addition requirements aroused during the support phase. For embedded products, the upgrades may be for the product resident firmware or for the hardware embedding the firmware. Some bugs may be easily fixed by modifying the firmware and it is known as firmware up-gradation. Some feature enhancements can also be performed easily by mere firmware modification. The product resident firmware will have a version number which starts with version say 1.0 and after each firmware modification or bug fix, the firmware version is changed accordingly (e.g. version 1.1). Version numbering is essential for backward traceability. Releasing of upgrades is managed through release management. (Embedded Product Engineers might have used the term "Today I have a release" at least once in their life). Certain feature enhancements and bug fixes require hardware modification and they are generally termed as hardware upgrades. Firmware also needs to be modified according to the hardware enhancements. The upgraded product appears in the market with the same name as that of the initial product with a different version number or with a different name.

15.4.9 Retirement/Disposal

We are living in a dynamic world where everything is subject to rapid changes. The technology you feel as the most advanced and best today may not be the same tomorrow. 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. When the product manufacture realises that there is another powerful technology or component available in the market which is most suitable for the production of the current product, they will announce the current product as obsolete and the newer version/upgrade of the same is going to be released soon. The retirement/disposition phase is the final phase in a product development life cycle where the product is declared as obsolete and discontinued from the market. There is no meaning and monetary benefit in continuing the production of a device or equipment which is obsolete in terms of technology and aesthetics. The disposition of a product is essential due to the following reasons.

1. Rapid technology advancement
2. Increased user needs

Some products may get a very long ‘Life Time’ in the market, beginning from the launch phase to the retirement phase and during this time the product will get reasonably good amount of maturity and stability and it may be able to create sensational waves in the market (e.g. Nokia 3310 Mobile Handset; Launched in September 2000 and discontinued in early 2005, more than 4 years of continued presence in the market with 126 million pieces sold). Some products get only small ‘Life Time’ in the market and some of them even fails to register their appearance in the market.

By now we covered all the phases of embedded product development life cycle. We can correlate the various phases of the product development we discussed so far to the various activities involved in our day-today life. Whenever you are a child you definitely have the ambition to become an earning person like your mom/dad (There may be exceptions too ☺). The need for a job arises here. To get a good job you must have a good academic record and hence you should complete the schooling with good marks and should perform very well in interviews. Finally you are managed to get a good job and your professional career begins here. You may get promotions to next senior level depending on your performance. At last that day comes—Your retirement day from your professional life.

15.5 EDLC APPROACHES (MODELING THE EDLC)

LO 5 Identify the different modelling techniques for modelling the stages involved in the embedded product development life cycle

The term ‘Modelling’ in Embedded Product Development Life Cycle refers to the interconnection of various phases involved in the development of an embedded product. The various approaches adopted or models used in Modelling EDLC are described below.

15.5.1 Linear or Waterfall Model

Linear or waterfall approach is the one adopted in most of the olden systems and in this approach each phase of EDLC is executed in sequence. The linear model establishes a formal analysis and design methodology with highly structured development phases. In linear model, the various phases of EDLC are executed in sequence and the flow is unidirectional with output of one phase serving as the input to the next phase. In the linear model all activities involved in each phase are well documented, giving an insight into what should be done in the next phase and how it can be done. The feedback of each phase is available locally and only after they are executed. The linear model implements extensive review mechanisms to ensure the process flow is going in the right direction and validates the effort during a phase. One significant feature of linear model is that even if you identify bugs in the current design, the corrective actions are not implemented immediately and the development process proceeds with the current design. The fixes for the bugs are postponed till the support phase, which accompanies the deployment phase. The major advantage of 'Linear Model' is that the product development is rich in terms of documentation, easy project management and good control over cost and schedule. The major drawback of this approach is that it assumes all the analysis can be done and everything will be in right place without doing any design or implementation. Also the risk analysis is performed only once throughout the development and risks involved in any changes are not accommodated in subsequent phases, the working product is available only at the end of the development phase and bug fixes and corrections are performed only at the maintenance/support phase of the life cycle. 'Linear Model' (Fig. 15.7) is best suited for product developments, where the requirements are well defined and within the scope, and no change requests are expected till the completion of the life cycle.

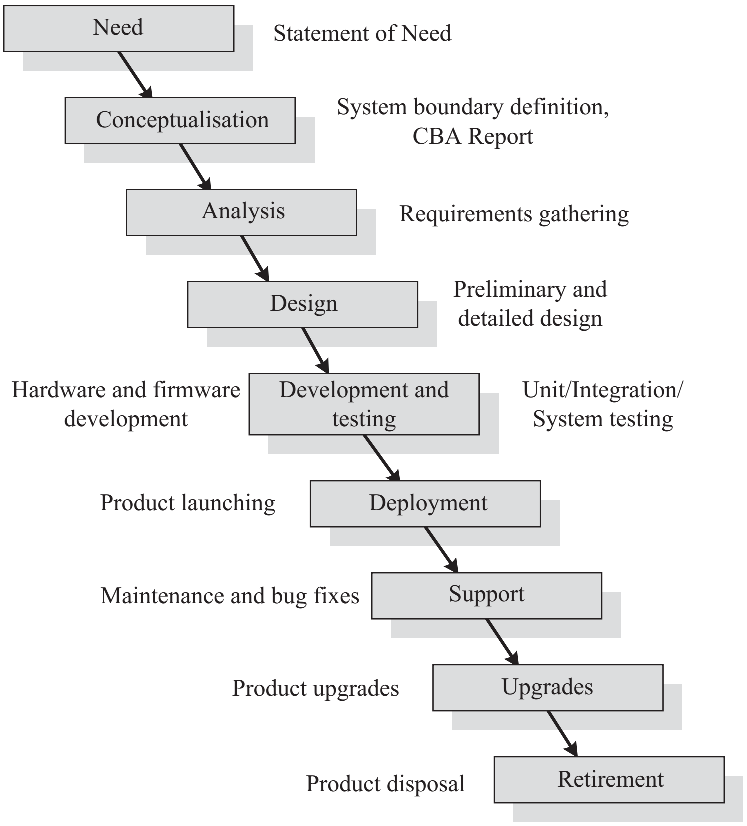

Fig. 15.7 Linear (Waterfall) EDLC Model

15.5.2 Iterative/Incremental or Fountain Model

Iterative or fountain model follows the sequence—Do some analysis, follow some design, then some implementation. Evaluate it and based on the shortcomings, cycle back through and conduct more analysis, opt for new design and implementation and repeat the cycle till the requirements are met completely. The iterative/fountain model can be viewed as a cascaded series of linear (waterfall) models. The incremental model is a superset of iterative model where the requirements are known at the beginning and they are divided into different groups. The core set of functions for each group is identified in the first cycle and is built and deployed as the first release. The second set of requirements along with the bug fixes and modification for first release is carried out in the second cycle and the process is repeated until all functionalities are implemented and they are meeting the requirements. It is obvious that in an iterative/incremental model (Fig. 15.8), each development cycle act as the maintenance phase for the previous cycle (release). Another approach for the iterative/interactive model is the 'Overlapped' model where development cycles overlap; meaning subsequent iterative/incremental cycle may begin before the completion of previous cycle.

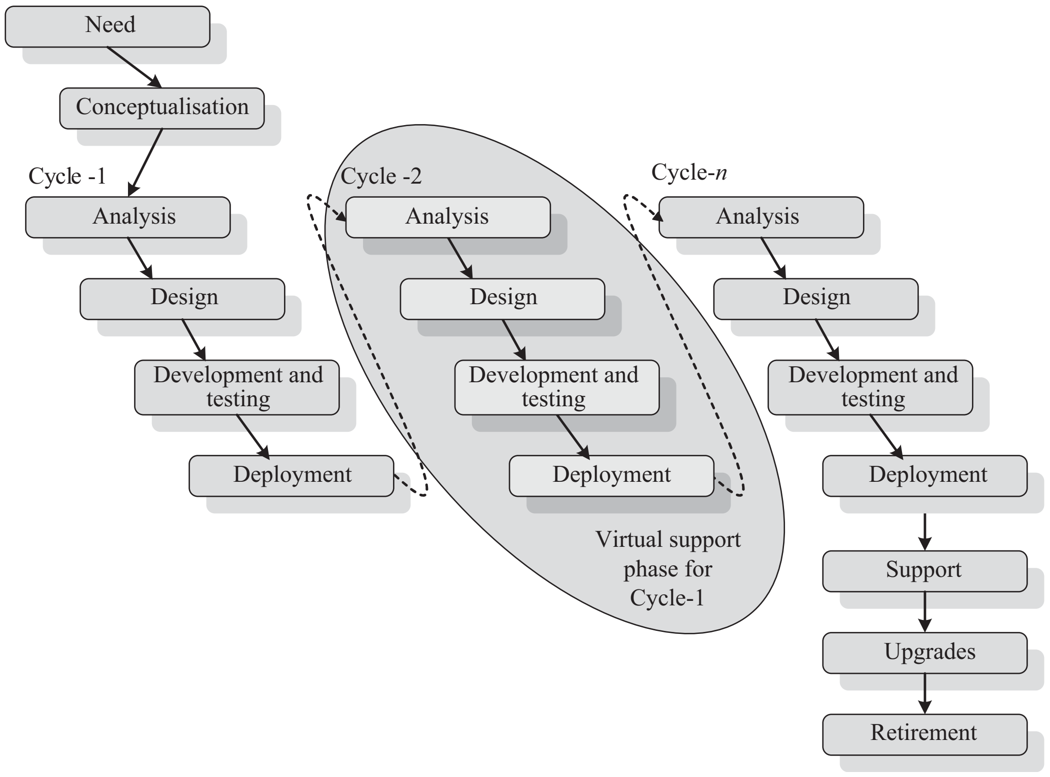

Fig. 15.8 Iterative/Incremental/Fountain EDLC Model

If you closely observe this model you can see that each cycle is interconnected in a similar fashion of a fountain, where water first moves up and then comes down, again moves up and comes down. The major advantage of iterative/fountain model is that it provides very good development cycle feedback at each function/feature implementation and hence the data can be used as a reference for similar product development in future. Since each cycle can act as a maintenance phase for previous cycle, changes in feature and functionalities can be easily incorporated during the development and hence more responsive to changing user needs. The iterative model provides a working product model with at least minimum features at the first cycle itself. Risk is spread across each individual cycle and can be minimised easily. Project management as well as testing is much simpler compared to the linear model. Another major advantage is that the product development can be stopped at any stage with a bare minimum working product. Though iterative model is a good solution for product development, it possess lots of drawbacks like extensive review requirement at each cycle, impact on operations due to new releases, training requirement for each new deployment at the end of each development cycle, structured and well documented interface definition across modules to accommodate changes. The iterative/incremental model is deployed in product developments where the risk is very high when the development is carried out by linear model. By choosing an iterative model, the risk is spread across multiple cycles. Since each cycle produces a working model, this model is best suited for product developments where the continued funding for each cycle is not assured.

15.5.3 Prototyping/Evolutionary Model

Prototyping/evolutionary model is similar to the iterative model and the product is developed in multiple cycles. The only difference is that this model produces a more refined prototype of the product at the end of each cycle instead of functionality/feature addition in each cycle as performed by the iterative model. There won't be any commercial deployment of the prototype of the product at each cycle's end. The shortcomings of the proto-model after each cycle are evaluated and it is fixed in the next cycle.

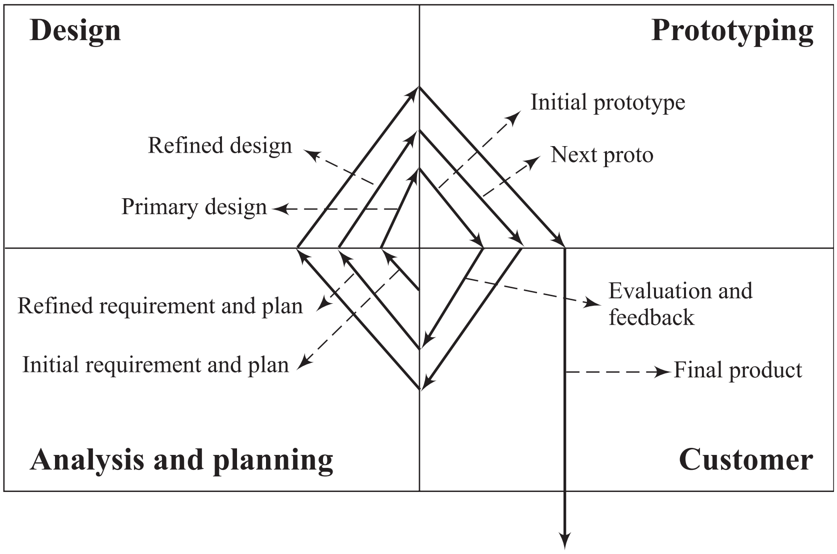

Fig. 15.9 Proto-typing/Evolutionary EDLC Model

After the initial requirement analysis, the design for the first prototype is made, the development process is started. On finishing the prototype, it is sent to the customer for evaluation. The customer evaluates the product for the set of requirements and gives his/her feedback to the developer in terms of shortcomings and improvements needed. The developer refines the product according to the customer's exact expectation and repeats the proto development process. After a finite number of iterations, the final product is delivered to the customer and launches in the market/operational environment. In this approach, the product undergoes significant evolution as a result of periodic shuttling of product information between the customer and developer. The prototyping model follows the approach – 'Requirements definition, proto-type development, proto-type evaluation and requirements refining'. Since the requirements undergo refinement after each proto model, it is easy to incorporate new requirements and technology changes at any stage and thereby the product development process can start with a bare minimum set of requirements. The evolutionary model relies heavily on user feedback after each implementation and hence finetuning of final requirements is possible. Another major advantage of proto-typing model is that the risk is spread across each proto development cycle and it is well under control. The major drawbacks of proto-typing model are

1. Deviations from expected cost and schedule due to requirements refinement
2. Increased project management
3. Minimal documentation on each prototype may create problems in backward prototype traceability
4. Increased Configuration Management activities

Prototyping model is the most popular product development model adopted in embedded product industry. This approach can be considered as the best approach for products, whose requirements are not fully available and are subject to change. This model is not recommended for projects involving the upgradation of an existing product. There can be slight variations in the base prototyping model depending on project management.

15.5.4 Spiral Model

Spiral model (Fig. 15.10) combines the elements of linear and prototyping models to give the best possible risk minimised EDLC Model. Spiral model is developed by Barry Boehm in 1988. The product development starts with project definition and traverse through all phases of EDLC through multiple phases. The activities involved in the Spiral model can be associated with the four quadrants of a spiral and are listed below.

1. Determine objectives, alternatives, constraints.
2. Evaluate alternatives. Identify and resolve risks.
3. Develop and test.
4. Plan.

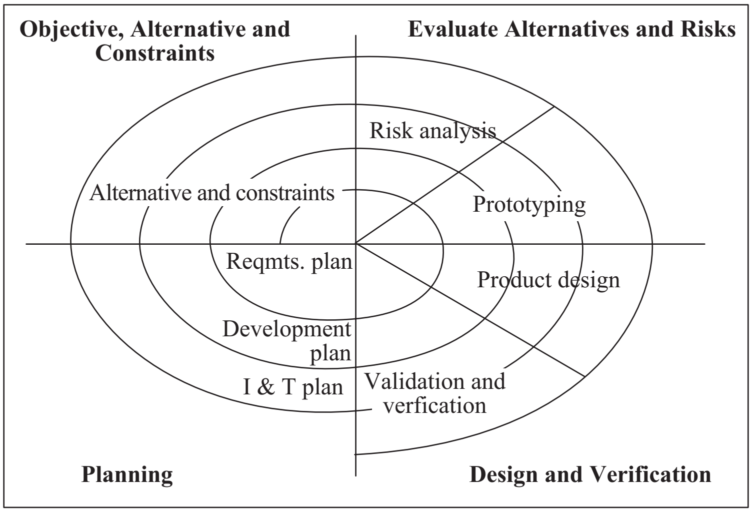
Objective, Alternative and Constraints
Evaluate Alternatives and Risks
Risk analysis
Alternative and constraints
Prototyping
Reqmts. plan
Product design
Development plan
I & T plan
Validation and verfication
Planning
Design and Verification

Fig. 15.10 Spiral Model

Spiral model is best suited for the development of complex embedded products and situations where requirements are changing from customer side. Customer evaluation of prototype at each stage allows addition of requirements and technology changes. Risk evaluation in each stage helps in risk planning and mitigation. The proto model developed at each stage is evaluated by the customer against various parameters like strength, weakness, risk, etc. and the final product is built based on the final prototype on agreement with the client.

Summary

- ✓ Embedded Product Development Life Cycle (EDLC) is an 'Analysis-Design-Implementation' based standard problem solving approach for Embedded Product Development **LO1**
- ✓ EDLC defines the interaction and activities among various groups of a product development sector including project management, system design and development (hardware, firmware and enclosure design and development), system testing, release management and quality assurance **LO1**
- ✓ Project management, productivity improvement and ensuring quality of the product are the primary objectives of EDLC **LO2**
- ✓ *Project Management* is essential for predictability, co-ordination and risk minimisation in product development **LO2**
- ✓ 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 **LO2**
- ✓ The classic Embedded Product Life Cycle Model contains the phases: *Need*, *Conceptualisation*, *Analysis*, *Design*, *Development and Testing*, *Deployment*, *Support*, *Upgrades* and *Retirement/ Disposal* **LO3**
- ✓ The product development *need* falls into three categories namely, New or custom product development, Product Re-engineering and product maintenance **LO4**
- ✓ *Conceptualisation phase* is the phase dealing with product concept development. Conceptualisation phase includes activities like product feasibility analysis, cost–benefit analysis, product scoping and planning for next phases **LO4**
- ✓ The *Requirements analysis* phase defines the inputs, processes, outputs, and interfaces of the product at a functional level. Analysis and documentation, interface definition and documentation, high level test plan and procedure definition, etc. are the activities carried out during requirement analysis phase **LO4**
- ✓ Product design phase deals with the implementation aspects of the required functionalities for the product. **LO4**
- ✓ The *Preliminary design* establishes the top-level architecture for the product, lists out the various functional blocks required for the product, and defines the inputs and outputs for each functional block **LO4**
- ✓ *Detailed Design* generates a detailed architecture, identifies and lists out the various components for each functional block, the inter-connection among various functional blocks, the control algorithm requirements, etc. **LO4**
- ✓ The *Development phase* transforms the design into a realisable product. The detailed specifications generated during the design phase are translated into hardware and firmware during the development phase **LO4**
- ✓ The *Testing phase* deals with the execution of various tests like *Integration testing*, *System testing*, *User acceptance testing*, etc. **LO4**
- ✓ 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* **LO4**
- ✓ The *Support phase* deals with the operations and maintenance of the product in a production environment **LO4**

- ✓ The *Upgrade phase* of product development deals with the development of upgrades (new versions) for the product which is already present in the market. Product upgrade results as an output of major bug fixes or feature enhancement requirements from the end user **LO4**
- ✓ *Retirement/Disposal phase* of a product deals with the gradual disposal of a product from the market **LO4**
- ✓ *Linear or Waterfall*, *Iterative or incremental or fountain*, *Prototyping or evolutionary* and *Spiral models* are the commonly used EDLC models in embedded product development **LO5**
- ✓ The *Linear* or *Waterfall* model executes all phases of the EDLC in sequence, one after another. It is the best suited method for product development where the requirements are fixed **LO5**
- ✓ *Iterative* or *Fountain* model follows the sequence—Do some analysis, follow some design, then some implementation. Evaluate it and based on the short comings, cycle back through and conduct more analysis, opt for new design and implementation and repeat the cycle till the requirements are met completely. **LO5**
- ✓ *Prototyping model* is a variation to the *Iterative model*, in which a more refined prototype is produced at the end of each iteration. It is the best suited model for developing embedded products whose requirements are not fully available at the time of starting the project, and are subject to change **LO5**
- ✓ *Spiral model* is the EDLC model combining linear and prototyping model to give the best possible risk minimisation in product development **LO5**

Keywords

Embedded Product Development Life Cycle (EDLC): An 'Analysis-Design-Implementation' based standard problem solving approach for Embedded Product Development **[LO 1]**

Conceptualisation: Product Life Cycle stage which deals with the concept development for a product **[LO 2]**

Model: The various phases involved in a product development life cycle **[LO 2]**

Need: Demand for a custom product or re-engineering of an existing product or maintenance of an existing product **[LO 2]**

Deployment Phase: Product Life Cycle stage which deals with the launching of the product **[LO 4]**

Design Phase: Product Life Cycle stage which deals with the implementation aspects of the required functionalities for the product **[LO 4]**

Development Phase: Product Life Cycle stage which transforms the design into a realisable product **[LO 4]**

First Customer Shipping (FCS): The process of handing over the fully functional initial model of the product to an end user/client **[LO 4]**

Integration Testing: Tests carried out to verify the functioning of a product for the required functionalities after the integration of embedded firmware and hardware **[LO 4]**

Requirement Analysis: Product Life Cycle stage which deals with the activities for developing a detailed functional model of the product under consideration **[LO 4]**

Retirement/Disposal: Product Life Cycle stage which deals with the gradual disposal of a product from the market **[LO 4]**

Support Phase: Product Life Cycle stage which deals with the operations and maintenance of the product in a production environment **[LO 4]**

System Testing: A set of various tests for functional and non-functional requirements verification of the product **[LO 4]**

Testing Phase: Phase which deals with the execution of various tests like Integration testing, System testing, User acceptance testing. etc. **[LO 4]**
Unit Testing: Tests carried out to verify the functioning of the individual modules of the firmware and hardware **[LO 4]**
Upgrade Phase: Product Life Cycle stage which deals with the development of upgrades (new versions) for the product which is already present in the market **[LO 4]**
User Acceptance Testing (UAT): Tests performed by the user (customer) against the acceptance values for each requirement **[LO 4]**
Iterative or Fountain Model: EDLC Model which follows the sequence—Do some analysis, follow some design, then some implementation **[LO 5]**
Linear or Waterfall Model: EDLC Model which executes all phases of the EDLC in sequence, one after another **[LO 5]**
Prototyping Model: EDLC Model which is the variation of the iterative model in which a more refined prototype is produced at the end of each iteration **[LO 5]**
Spiral Model: EDLC model combining linear and prototyping model to give the best possible risk minimisation in product development **[LO 5]**

Objective Questions

1. Which of the following is not in the scope of EDLC
(a) Maximise Return on Investment (RoI)
(b) Minimise the risks involved in the product development
(c) Advertise the product
(d) Defect prevention in Product development

2. The term ‘model’ in EDLC represents
(a) The various phases in the life cycle of the product
(b) The various analysis of the product
(c) The various designs of the product
(d) The various architecture of the product

3. Which of the following is(are) a driving factor(s) for ‘Product Re-engineering’?
(a) Change in business requirement (b) User interface enhancements
(c) Technology upgrades (d) All of these
(e) (a) and (b) (f) (a) and (c)

4. An embedded product requires interfacing with another subsystem of an enterprise for data logging and information exchange. The interface for this is defined during?
(a) Conceptualisation phase (b) Requirement analysis phase
(c) Design phase (d) Development phase
(e) Deployment phase

5. An embedded product requires a TCP/IP interface and it is decided that the TCP/IP interface can be implemented using Commercial of the Shelf (COTS) component. The COTS module for the TCP/IP Module is finalised and selected during?
(a) Conceptualisation phase (b) Requirement analysis phase
(c) Design phase (d) Development phase
(e) Deployment phase

6. An embedded product contains LCD interfacing and a developer is assigned to work on this module. The developer coded the module and tested its functioning using a simulator. The test performed by the developer falls under?
   (a) Integration Testing (b) Unit Testing
   (c) System Testing (d) Acceptance Testing
7. For an Embedded product, the requirements are well defined and are within the scope and no change requests are expected till the completion of the life cycle. Which is the best-suited life cycle model for developing this product?
   (a) Linear (b) Iterative (c) Prototyping (d) Spiral
8. Which of the following is(are) not a feature of prototyping model
   (a) Requirements refinement (b) Documentation intensive
   (c) Intensive Project Management (d) Controlled risk
9. An embedded product under consideration is very complex in nature and there is a possibility for change in requirements of the product. Also the risk associated with the development of this product is very high. Which is the best-suited life cycle method to handle this product development?
   (a) Linear (b) Iterative (c) Prototyping (d) Spiral

Review Questions

1. What is EDLC? Why EDLC is essential in embedded product development? **[LO 1]**
2. What are the objectives of EDLC? **[LO 1, LO 2]**
3. Explain the significance of *Productivity* in embedded product development. Explain the techniques for productivity improvements. **[LO 3]**
4. Explain the different phases of Embedded Product Development Life Cycle (EDLC). **[LO 4]**
5. Explain the term '*model*' in relation to EDLC. **[LO 5]**
6. Explain the different types of *Product Development* needs. **[LO 4]**
7. Explain the *Product Re-engineering* need in detail. **[LO 4]**
8. Explain the various activities performed during the *Conceptualisation* phase of an Embedded product development. **[LO 4]**
9. Explain the various activities performed during the *Requirement Analysis* phase of an embedded product development. **[LO 4]**
10. Explain the various activities performed during the *Design* phase of an embedded product development. **[LO 4]**
11. Explain the various activities performed during the *Development* phase of an embedded product development. **[LO 4]**
12. Explain the various types of tests performed in the development of an embedded product. **[LO 4]**
13. Explain the various activities performed during the *Deployment* phase of an embedded product. **[LO 4]**
14. Explain the various activities performed during the *Support* phase of an embedded product. **[LO 4]**
15. Explain the need for *Product upgrades* in the Embedded Product development. Explain the different types of product upgrades. **[LO 4]**
16. Explain the various factors leading to the *Retirement/Disposal* of an embedded product. **[LO 4]**

17. Explain the different *Life Cycle Models* adopted in embedded product development. **[LO 5]**

18. Explain the merits and drawbacks of *Linear* model for embedded product development. **[LO 5]**

19. Explain the similarities and differences between *Iterative* and *Incremental* life cycle model. **[LO 5]**

20. Explain the merits and drawbacks of *Fountain* model for embedded product development. **[LO 5]**

21. Explain the similarities and differences between *Iterative* and *Evolutionary* life cycle model. **[LO 5]**

22. Explain the merits and drawbacks of *Prototyping* model for embedded product development. **[LO 5]**

23. Explain the need for *Spiral* life cycle model in embedded product development. **[LO 5]**