Mumbai University

2013 - 2014

B.Sc.IT: Semester – VI

Practical Manual Question Paper

[CBSGS – 75:25 Pattern]

Project Management

TYBSc(IT) Semester VI

Credit Based Semester and Grading System

Paper II – Project Management

Reference Manual for Instructors for Conducting Practicals

Developed & Compiled By

Mr. Hiren Dand, MCC

&

Mrs. R. Srivaramangai, UDIT

2013 – 2014

Requirements :

1. Ms Project
2. StarUML
3. WBSPro(Optional)

Practical No. 1 : SYSTEM REQUIREMENT STUDY (SRS) FOR A PROJECT

Case :

ABC company wishes to create a small software program for keeping the stock of their goods especially the raw materials. This they named as ABC_Inventory and would like to deploy their in-house development cell for the development.

Aim:

TO Describe the System Requirement Study of your Project (in this case the ABC Inventory)with necessary illustrations and following the standards

Objective : ABC Inventory's main objective is to keep track of all the parts and finished goods in the warehouses and all the transactions involving these parts and goods.

Introduction

Currently the Inventory is carried out in the traditional way and the information of the products, supplier, orders etc are kept in an Excel workbook. The project is aimed in such a way that it should be possible for the new system to interact with the already existing systems or may with the newer systems that may be created. Other existing systems are :

1. Purchase
2. Production
3. Customer Order Entry/Shipping
4. Subsystems

The Inventory Control System (ICS) known as ABC Inventory will work in the existing Enterprise Planning System (ERP) system.

Overall Description

The newly developed system will be invoked mainly when:

- purchased goods arrive from vendors,
- goods are returned to the vendor for various reasons,

  - stock is issued to/from manufacturing,
  - finished goods are entered in the Finished Goods Warehouse
  - finished goods are shipped to the customer
  - sales persons wish to check the availability of products.

- **Interface with Purchasing:**

  When an order from a vendor arrives, our Purchase Order Number on the vendor's invoice will be used to locate the the PO and other related documents, such as vendor proposals, price lists, product specifications.

  These will then be forwarded to Quality Control to be used in the inspection of the goods.

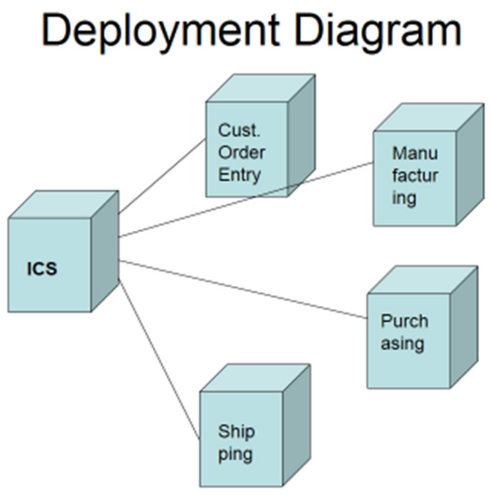
  An interface will be worked out with Purchasing subsystem currently running on Server PS1 so that ICS can find all documents related to the current purchase.

- **Interface with Manufacturing:**

  Mfg requests materials
  Mfg wants to store finished goods

- Interface with Customer Order Entry/Shipping:

  COES wants to ship goods
  COES wants to store returned goods



An automatic purchase order generation should be provisioned when the stock levels of parts falls below the minimum has been contemplated but postponed to a future date.

The main functions of system are:

- Entering new parts into the inventory
- Issuing parts to Manufacturing
- Entering finished goods to the inventory
- It also should:
  - Provide a user interface consistent with the other subsystems
  - Verify user identities and access rights
  - Do error checking on all inputs
  - Permit users to add customized reports

Project Constraints

- **Compatibility**
  - The software should run compatibly (i.e. under the same operating system, database and networking capabilities) with the other subsystems software it works together with.
  - It should allow an Administrator to enroll new users and give them access rights required by their duties.
- **Reliability and Availability**
  - Since it interfaces with Purchasing, Customer Order Entry and Manufacturing, it should be available as a minimum when any one of these subsystems are available.
  - It should permit 1 hour per day for maintenance and backup activities with minimal disruption to users.
  - Any failure should cause no more than 10-minute downtime, with the average not exceeding 2 minutes.
  - Backup should spot-tested to ensure they are reliable.
- **Performance**

- It should allow up to 10 users to logon simultaneously and receive an average response time not exceeding 3 seconds.
- Parts received should be in the recorded in the system in no more than 2 hours, with the average under 30 minutes.

External Information from other subsystems to be used by Inventory System

1. Vendor Order Master File - Purchasing
2. Vendor Order Detail File - Purchasing

3. Customer Order Master File – Customer Order Entry
4. Customer Order Detail File – Customer Order Entry

5. Manufacturing Requests – Manufacturing
6. Manufacturing Schedule– Manufacturing

System's Own information storage

1. Part Transaction File
2. Part Master File
3. Part Detail File
4. Part Inventory File

Major Transactions to be provided by the system are:

- Create New Part or Product Master Record Edit/Delete Part or Product Master Record
- Edit/Delete Part or Product Record
- Enter New Part or Product
- Enter Returned Part or Product
- Issue Part or Product
- Re-issue Part or Product
- View Part or Product

Detailed description of the Transactions

For Creation of Products/Parts

Narrative: Creates a new part or product type. After the type has been created, part or product instances can be entered. Permits user to enter a new type code and its description. The type code should be selected according to the “Part/product Type Code Guidelines” document. Uses PartMaster Class and PartDetail Class.

Constraints:
Requires “Product Administrator” rights for use.

Performance constraints: Since it locks PartMaster, the transaction should time-out after 15 seconds and backout all changes.

Design Constraints: Must check type code for compliance with Guidelines

Supporting Diagrams: See Class diagram for PartMaster and Part Detail classes.

For Modification and Deletion of Parts

Permits user to edit the description of a part or product or delete it. If there are any instances of the part or product, it can not be deleted. Uses PartMaster Class, PartDetail Class and PartInventory Class.

Constraints:
Requires “Product Administrator” rights for use.

Performance constraints: Since it locks PartMaster, PartDetail and PartInventory Classes, the transaction should time-out after 15 seconds and backout all changes.

Design Constraints: Must check type code for compliance with Guidelines. Any instances of the part or product are in existence, only certain fields may be edited.

Methodology adopted :

Object oriented approach since the components created can be reused later on in other susystems. Scalability and Reliability also makes it towards object orientation.

So the requirement specifications are to be done with UML diagrams for the front – end and processing which consists of

Use case diagrams

Acitvity diagrams

Sequence diagrams

Class diagrams

An E-R model will help to imcorporate the information storage in the form of relational tables in the chosen ORDBMS environment

Before placing the SRS, feasibility study needs to be done. The overall methodology and description constitutes the SRS of the system

Practical No. 2 : Waterfall Model as the conventional process model to prepare the flow and Gantt Chart

To be done as per Software Engineering approach for any project preferably the student's project using waterfall model as the process model and to prepare the scheduling chart using Gantt chart

Practical No. 3 : Cost Estimation of the project Using Function Point Analysis (FPA)

What is Function Point Analysis (FPA)?

- **It is designed to estimate and measure the time, and thereby the cost, of developing new software applications and maintaining existing software applications.**
- **It is also useful in comparing and highlighting opportunities for productivity improvements in software development.**
- **It was developed by A.J. Albrecht of the IBM Corporation in the early 1980s.**
- **The main other approach used for measuring the size, and therefore the time required, of software project is lines of code (LOC) – which has a number of inherent problems.**

How is Function Point Analysis done?

Working from the project design specifications, the following system functions are measured (counted):

- **Inputs**
- **Outputs**
- **Files**
- **Inquires**
- **Interfaces**

These function-point counts are then weighed (multiplied) by their degree of complexity:

	Simple	**Average**	**Complex**
Inputs	**2**	**4**	**6**
Outputs	**3**	**5**	**7**
Files	**5**	**10**	**15**
Inquires	**2**	**4**	**6**
Interfaces	**4**	**7**	**10**

A simple example:

inputs

3 simple X 2 = 6

4 average X 4 = 16

1 complex X 6 = 6

outputs

6 average X 5 = 30

2 complex X 7 = 14

files

5 complex X 15 = 75

inquiries

8 average X 4 = 32

interfaces

3 average X 7 = 21

4 complex X 10 = 40

Unadjusted function points 240

- **There are 14 more "Degree of Influences" (0 to 5 scale) :**
    - data communications
    - distributed data processing

- performance criteria
- heavy hardware utilization
- high transaction rate
- online data entry
- end user efficiency
- on-line update
- complex computation
- reusability
- ease of installation
- ease of operation
- portability
- maintainability

- **Define Technical Complexity Factor (TCF):**
  - TCF = .65 + [(.01) x (14 DIs)]
  - where DI = ∑ (influence factor value)

Function Point = Initial FP X TCF.

But how long will the project take and how much will it cost?

- Suppose programmers in our organization average 18 function points per month and suppose the Function Point is 197

  197 FP divided by 18 = 11 man-months

- If the average programmer is paid $5,200 per month (including benefits), then the [labor] cost of the project will be . . .

  11 man-months X $5,200 = $57,200

The total cost for far estimated is $57,200

Practical No. 4 : Cost Estimation of the project Using COCOMO Model I

Software cost estimation is important for making good management decisions. It is also connected to determining how much effort and time a software project requires. The

COnstructive COst MOdel (COCOMO) is an example of regression models used for estimating software cost and effort. These methods use a basic formula with parameters that are determined via a historical database and current project characteristics.

The Basic COCOMO Model:

The Basic Model makes its estimates of required effort (measured in Staff-Months **SM**) based primarily on your estimate of the software project's size (as measured in thousands of Delivered Source Instructions **KDSI**):

$SM = a * (KDSI)^b$

The Basic model also presents an equation for estimating the development schedule (Time of Develop **TDEV**) of the project in months:

$TDEV= c * (SM)^d$

Following questions are to be answered before proceeding with the calculation.

- *Which Instructions count as delivered source instructions?*
- *Which Staff-Months are included in the estimate?*
- *Which phases are included in "development" ?*
- *What classes of projects are covered by the estimating equations?*

Definitions and Assumptions while estimating COCOMO model I

1. The primary cost driver is the number of delivered source instructions (**DSI**) developed by the project. DSI is defined such that:
   - Only source lines that are *DELIVERED* as part of the product are included. Therefore test drivers and other support software are excluded.
   - *SOURCE* codes, created by project staff and processed into machine code by some combination of pre-processors, compilers, and assemblers. Code created by applications generators is excluded.
   - *INSTRUCTIONS* are defined as lines of code or card images. Declarations are counted as instructions but Comments are excluded.
2. The COCOMO cost estimates only cover the period between beginning of the design phase and end of the integration and test phase. Costs and schedules of other phases (like requirement phase) are estimated separately.
3. A COCOMO Staff-Month (**SM**) consists of 152 hours of working time. This has to be found to be consistent with practical experience with the average monthly time due to holidays, vacation and sick leaves. COCOMO avoids estimating labour costs in dollars because of the large variations between organizations in what is included in labour costs and because **SM** is a more stable unit than dollars. After estimating the effort in **SM** we convert it into dollar estimate by applying different average dollar per Staff-Month figure for each major phase, to account for inflation and the difference in salary level of the people required for each phase.For example, the senior staff (with high salary)are heavily

involved in the requirement and early design phases while junior staff (less salary) are more involved in later phases (detailed design, code and test phases) Therefore the average **SM** is higher for early phases.

4. COCOMO assumes that the requirement specification is not substantially changed after the plans and requirement phase.Any significant modification should be covered by a revised cost estimate.
5. The detailed COCOMO model assumes that the influence of the software cost drivers is phase dependent. Basic COCOMO and Intermediate COCOMO do not, except for distinguishing between development and maintenance.

The Development Mode:

There are several modes of software development .These different software development modes have cost-estimating relationships which are similar in form, but which yield significantly different cost estimates for software products of the same size.In the COCOMO Model, one of the most important factors contributing to a project's duration and cost is the Development mode. Every project is considered to be developed in one of three modes:

- **Organic Mode.**
- **Semidetached Mode**
- **Embedded Mode**

To estimate the effort and development time, COCOMO use the same equations but with different coefficients (*a, b, c, d* in the effort and schedule equations *)* for each development mode. Therefore before using the COCOMO model we must be able to recognise the development mode of our project.

1. **Organic Mode:**

   In the organic mode the project is developed in a familiar, stable environment and the product is similar to previously developed products. The product is relatively small, and require little innovation. Most people connected with the project have extensive experience in working with related systems within the organization and therefore can usefully contribute to the project in its early stages, without generating a great deal of project communication overhead. An organic mode project is relatively relaxed about the way the software meets its requirements and interface specifications. If a situation arises where an exact correspondence of the software product to the original requirements would cause an extensive rework, the project team can generally negotiate a modification of the specifications that can be developed more easily.

   The Basic COCOMO Effort and schedule equations for organic mode software projects are:

   $SM = 2.4 * (KDSI)^{1.05}$

   $TDEV= 2.50 * (SM)^{0.38}$

2. **Semidetached Mode:**

   In this mode project's characteristics are intermediate between Organic and Embedded. "Intermediate" may mean either of two things:

   1. An intermediate level of project characteristics.
   2. A mixture of the organic and embedded mode characteristics.

   Therefore in an Semidetached mode project, it is possible that:

   - The team members all have an intermediate level of experience with related systems.
   - The team has a wide mixture of experienced and inexperienced people.
   - The team members have experience related to some aspects of the system under development, but not others.

   The size of a Semidetached mode product generally extends up to 300 KDSI.

   The Basic COCOMO Effort and schedule equations for organic mode software projects are:

   $SM = 3.0 * (KDSI)^{1.12}$

   $TDEV= 2.50 * (SM)^{0.35}$

Embedded Mode:

In this development mode Project is characterized by tight , inflexible constraints and interface requirements. The product must operate within a strongly coupled complex of hardware, software, regulations, and operational procedures. The embedded-mode project does not generally have the option of negotiating easier software changes and fixes by modifying the requirements and interface specifications.The project therefore need more effort to accommodate changes and fixes. The embedded mode project is generally charting its way through unknown territory to a greater extent than the organic mode project. This lead the project to use a much smaller team of analyst in the early stages, as a large number of people would get swamped in communication overhead.

Once the embedded mode project has completed its product design, its best strategy is to bring on a very large team of programmers to perform detailed design, coding and unit testing in parallel. Otherwise the project would take much longer to complete. This strategy as we will see leads to the higher peaks in the personnel curves of embedded-mode projects, and to the greater amount of effort consumed compared to an organic mode project working to the same total development schedule.

The Basic COCOMO Effort and schedule equations for organic mode software projects are:

$SM = 3.6 * (KDSI)^{1.20}$

*TDEV= 2.50 * (SM)*$^{0.32}$

Problem : *An initial study has determined that the size of the program will be roughly 52,000 delivered source instructions for ABC Inventory.*

This project is a good example of an organic-mode software project. Using the Basic COCOMO equations for this development mode we have:

Effort :

SM *= 2.4*(52)*$^{1.05}$ *= 152* ***Staff-Months***

Productivity:

*52000****DSI*** */ 152* ***SM =*** *342* ***DSI / SM***

Duration and Staffing:

Once an estimate is obtained for effort (Staff-Month), The project manager must determine how many persons to put on the job. This will ultimately determine the calendar duration of the project. It is very important to note that more staff does not mean proportionately less calendar time. More staff complicate communications and this complexity translates into a project slowdown. The second equation of the COCOMO model use the estimated effort of the project (**SM**) to suggest the optimum calendar duration of the project. For the example above with estimated effort of 152 SM we have:

Schedule:

TDEM = *2.5 * (152)*$^{0.38}$ *= 16* ***months 8 days(17 months)***

After estimating the duration of the project the manager can easily determine how many persons in the average must be put on the project:

Average Staffing:

152 ***staff-months / 17 months = 9 FSP (F****ull Time Equivalent* ***S****oftware* ***P****ersonnel* ***)***

Phase Distribution of Effort and Schedule for Organic Mode:

After estimating the effort an schedule of a project we need to determine how to distribute them among different phases of the project.This distribution varies as a function of the size of the product.Larger software projects require relatively more time and effort to perform integration and test activities, and are able to compress the programming portion of the program by having larger number of peoples programming components in parallel. Smaller software projects have a more uniform, flat distribution of labour throughout the development cycle, and have relatively more resources devoted to the phases other than integration and test.Table 1 and Table 2 present the percentage distribution of the basic software effort and schedule within the development phases of an organic mode product:

Table 1 - Phase Distribution of Effort: Organic Mode

Phase	***Small (2 KDSI)***	***Intermediate(8 KDSI)***	***Medium(32 KDSI)***	***Large(128 KDSI)***
Plans & Requirements	*6%*	*6%*	*6%*	*6%*
Product Design	*16*	*16*	*16*	*16*
Detailed Design	*26*	*25*	*24*	*23*
Code & Unit Test	*42*	*40*	*38*	*36*
Integration & Test	*16*	*19*	*22*	*25*
Total:	*100*	*100*	*100*	*100*

Table 2 - Phase Distribution of Schedule: Oraganic Mode

Phase	***Small (2 KDSI)***	***Intermediate(8 KDSI)***	***Medium(32 KDSI)***	***Large(128 KDSI)***
Plans & Requirements	*10%*	*11%*	*12%*	*13%*
Product Design	*19*	*19*	*19*	*19*
Detailed Design & ***Code & Unit Test***	*63*	*59*	*55*	*51*
Integration & Test	*18*	*22*	*26*	*30*
Total:	*100*	*100*	*100*	*100*

Using table 1 and table 2 we can calculate the number of staff needed for programming (Detailed Design & Code and Unit Test) phase of the previous example:

Programming Effort :

*(0.62) (152 **SM**)= 94 **Staff-Months***

Programming Schedule:

*(0.55) (17)= 9 **months***

Average Staffing:

*94 **staff-months / 9 months =** 10.4 **FSP (F**ull Time Equivalent **S**oftware **P**ersonnel **)***

Phase Distribution of Effort and Schedule for Other Modes:

Table 3 and Table 4 present the percentage distribution of the basic software effort and schedule within the development phases of an semidetached mode product:

Table 3 - Phase Distribution of Effort: Semidetached Mode

Phase	***Small (2 KDSI)***	***Intermediate(8 KDSI)***	***Medium(32 KDSI)***	***Large(128 KDSI)***
Plans & Requirements	*7%*	*7%*	*7%*	*7%*
Product Design	*17*	*17*	*17*	*17*
Detailed Design	*27*	*26*	*25*	*24*
Code & Unit Test	37	35	33	31
Integration & Test	19	22	25	28
Total:	100	100	100	100

Table 4 - Phase Distribution of Schedule: Semidetached Mode

Phase	Small (2 KDSI)	Intermediate(8 KDSI)	Medium(32 KDSI)	Large(128 KDSI)
Plans & Requirements	16%	18%	20%	22%
Product Design	24	25	26	27
Detailed Design & Code & Unit Test	56	52	48	44
Integration & Test	20	23	26	29
Total:	100	100	100	100

Table 5 and Table 6 present the percentage distribution of the basic software effort and schedule within the development phases of an embedded mode product:

Table 3 - Phase Distribution of Effort: Embedded Mode

Phase	Small (2 KDSI)	Intermediate(8 KDSI)	Medium(32 KDSI)	Large(128 KDSI)
Plans & Requirements	8%	8%	8%	8%
Product Design	18	18	18	18
Detailed Design	28	27	26	25
Code & Unit Test	32	30	28	26
Integration & Test	22	25	28	31
Total:	100	100	100	100

Table 4 - Phase Distribution of Schedule: Embedded Mode

Phase	Small (2 KDSI)	Intermediate(8 KDSI)	Medium(32 KDSI)	Large(128 KDSI)
Plans & Requirements	24%	28%	32%	36%
Product Design	30	32	34	36
Detailed Design & Code & Unit Test	48	44	40	36
Integration & Test	22	24	26	28
Total:	100	100	100	100

By comparing tables 1 through 6 we can see some differences between the effort and schedule distribution of the products developed in different modes. The main differences are:

The embedded-mode project consumes considerably more effort in the integration and test phase. This results from the need to follow and verify software requirements and interface specifications more carefully in the embedded and semidetached mode.
The embedded-mode project consumes proportionally less effort in the code and unit test phase. This results from the proportionally higher effort required for the other development phases.
The embedded-mode project consumes considerably more schedule in both the plans and requirement phase and the product design phase. This is because of the project's need for more thorough, validated requirements and design specifications, and the greater need to perform these phases with a relatively small number of people.
The embedded-mode project consumes considerably less schedule in the programming phase. This results from the strategy of employing a great many people programming in parallel, in order to reduce the project's overall schedule.

Intermediate COCOMO Model:

The Intermediate COCOMO is an extension of the basic COCOMO model. Here we use the same basic equation for the model. But coefficients are slightly different for the effort equation.Also in addition to the size as the basic cost driver we use 15 more predictor variables. these added cost drivers help to estimate effort and cost with more accuracy.

An estimator looks closely at many factors of a project such as amount of external storage required, execution speed constraints, experience of the programmers on the team, experience with the implementation language, use of software tools, etc., for each characteristic, the estimator decide where on the scale of "very low" , " low", " Nominal", "High", "Very High" and "High" the project falls. Each characteristic gives an adjustment factor(from the table 7) and all factors are multiplied together to to give an Effort Adjustment Factor (EAF).If a project is judged normal in some characteristic the adjustment factor will be 1 for that characteristic (Nominal column in Table 7), which means that that factor has no effect on overall EAF. The effort equation for the intermediate model has the form of:

SM = EAF * a * (KDSI)b

If we assume that the project is "Nominal" in every aspect then all adjustment factors would be 1, which results in EAF=1, and the effort equation would have the same form as the Basic mode.
in addition to the EAF the model parameter "a" is slightly different in Intermediate COCOMO, but the "b" parameter is the same.The effort equation for different modes of Intermediate COCOMO is given in the following table:

Development Mode	Intermediate Effort Equation
Organic:	SM = EAF * 3.2 * (KDSI)1.05
SemiDetached	SM = EAF * 3.0* (KDSI)1.12
Embedded	SM = EAF * 2.8* (KDSI)1.20

Bohem in Software Engineering Economics defines each of the cost drivers, and defines the Effort Multiplier associated with each rating (Table 7).
Table 7 - Project Characteristic Table

Cost Driver	Very Low	Low	Nominal	High	Very High	Extra High
ACAP Analyst Capability	1.46	1.19	1.00	0.86	0.71	--
AEXP Application Experience	1.29	1.13	1.00	0.91	0.82	--
CPLX Product Complexity	0.70	0.85	1.00	1.15	1.30	1.65
DATA Database Size	--	0.94	1.00	1.08	1.16	--
LEXP Language Experience	1.14	1.07	1.00	0.95	--	--

MODP Modem Programming Practices	1.24	1.10	1.00	0.91	0.82	--
PCAP Programmer Capability	1.42	1.17	1.00	0.86	0.70	--
RELY Required Software Reliability	0.75	0.88	1.00	1.15	1.40	--
SCED Required Development Schedule	1.23	1.08	1.00	1.04	1.10	--
STOR Main Storage Constraint	--	--	1.00	1.06	1.21	1.56
TIME Execution Time Constraint	--	--	1.00	1.11	1.30	1.66
TOOL Use of Software Tools	1.24	1.10	1.00	0.91	0.83	--
TURN Computer Turnaround Time	--	0.87	1.00	1.07	1.15	--
VEXP Virtual Machine Experience	1.21	1.10	1.00	0.90	--	--
VIRT Virtual Machine Volatility	--	0.87	1.00	1.15	1.30	--

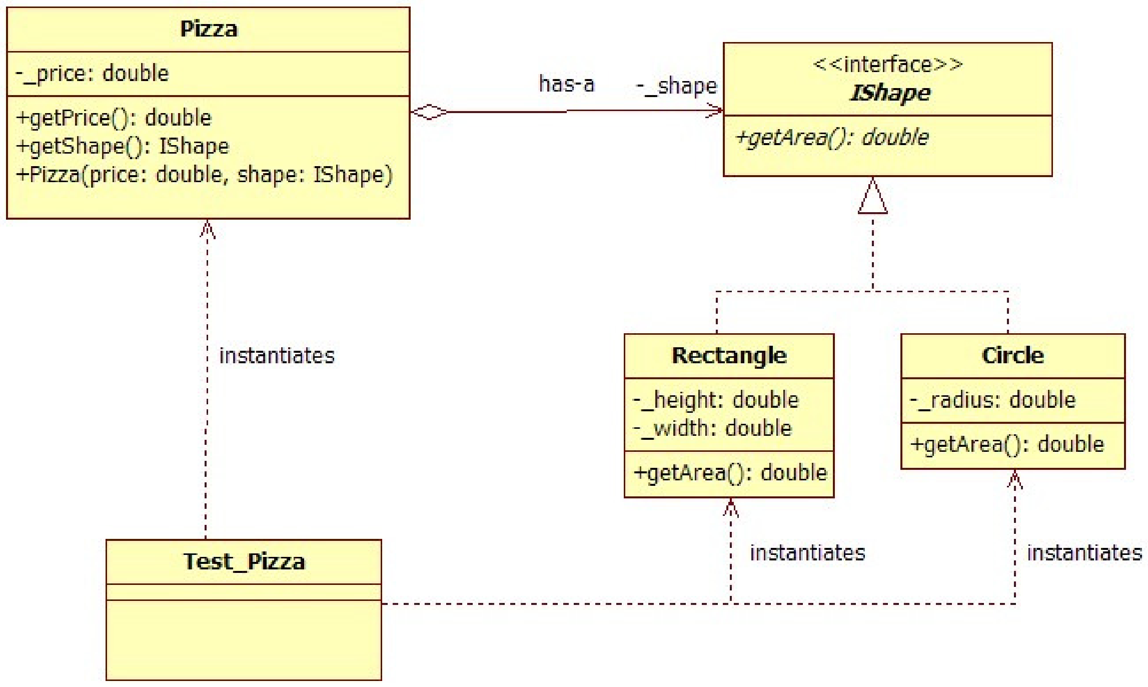
Example:

If your project is rated Very High for Complexity (Effort Multiplier of 1.30), and Low for Tools Use (Effort Multiplier of 1.10), and all of the other cost drivers are rated to be Nominal, these Effort Multipliers are used to calculate the Effort Adjustment Factor, which is used in the effort equation:

EAF = 1.30 * 1.10 = 1.43
Effort = EAF * 3.2 * 31.05 = 14.5 SM
TDEV = 2.5 * 14.50.38 = 6.9 Months
Average staffing:
14.5 / 6.9 = 2.1 people

There are two reasons why Intermediate model produces better results than the Basic Model. First, It considers the effect of more cost drivers. Second, in the Intermediate mode the system can be divided into "components". DSI value and Cost Drivers can be chosen for individual components, instead of for the system as a whole.COCOMO can estimate the staffing, cost, and duration of each of the components--allowing you to experiment with different development strategies, to find the plan that best suits your needs and resources.

Practical No. 5 : Class diagram using StarUML



Practical No. 6 : Use Case diagram using StarUML

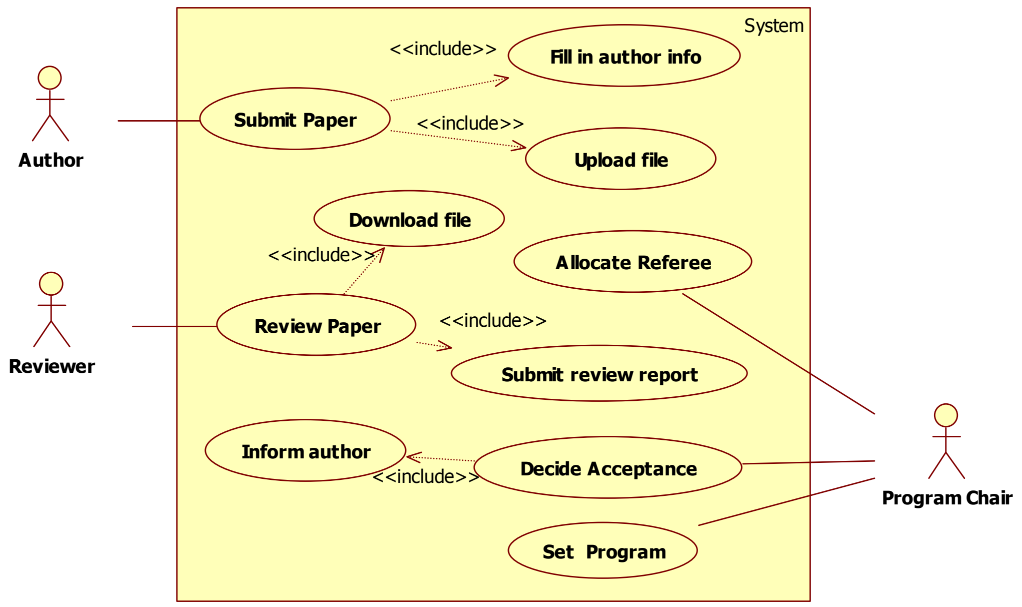
Use case Modelling I

Aim: To Analyse the below text and then draw a use case diagram using an UML modelling tool such as StarUML.

The following is a narrative description of the business process of organisation of conferences with regards to the submitting, reviewing and accepting papers.

The author completes an online form that requests the user to input author name, correspondence address, email and, title of paper. The system validates this data and, if correct, asks the author to submit the paper. The author then browses to find the correct paper on their system and submits it. Once received and stored, the system returns to the author a reference number for the paper. Authors may submit as many papers as they like to be considered for acceptance to the conference up until the deadline date for submissions. Papers are allocated to referees for assessment. They review each paper and submit to the system their decision. Once the programme organiser has agreed the decisions authors are informed by email. Accepted papers are then schedule to be delivered at a conference. This involves allocating a date, time and place for the presentation of the paper.

Solution:



Practical No. 7 : Acitvity description for the project

Activity Diagram for the prepared Usecase Modelling

Aim : Derive an activity diagram from the narrative text by following the steps outlined below.

- Translate the above narrative text into an activity list. An activity list consists of a sequences of short sentences in the form of

name + *verb* + *noun* [+ *condition*],

where the *condition* is optional. Such sentences are called activities. For example, the following is an activity.

'*the author completes an online form*'.

1. Transform each activity in the list into an activity/action node of the activity diagram;
2. Identify the actors involved in the business process, and associate the actors to each action/activity node;
3. Identify the sequential execution order, parallelism/concurrency and synchronisation as well between the actions/activities, and draw the control flows between the action/activity nodes, add synchronisation bar as necessary;
4. Draw a swimlane for each actor and place the action/activity nodes in appropriate swimlane;
5. Identify the objects passed between the activities, and add object nodes and object flows into the diagram.

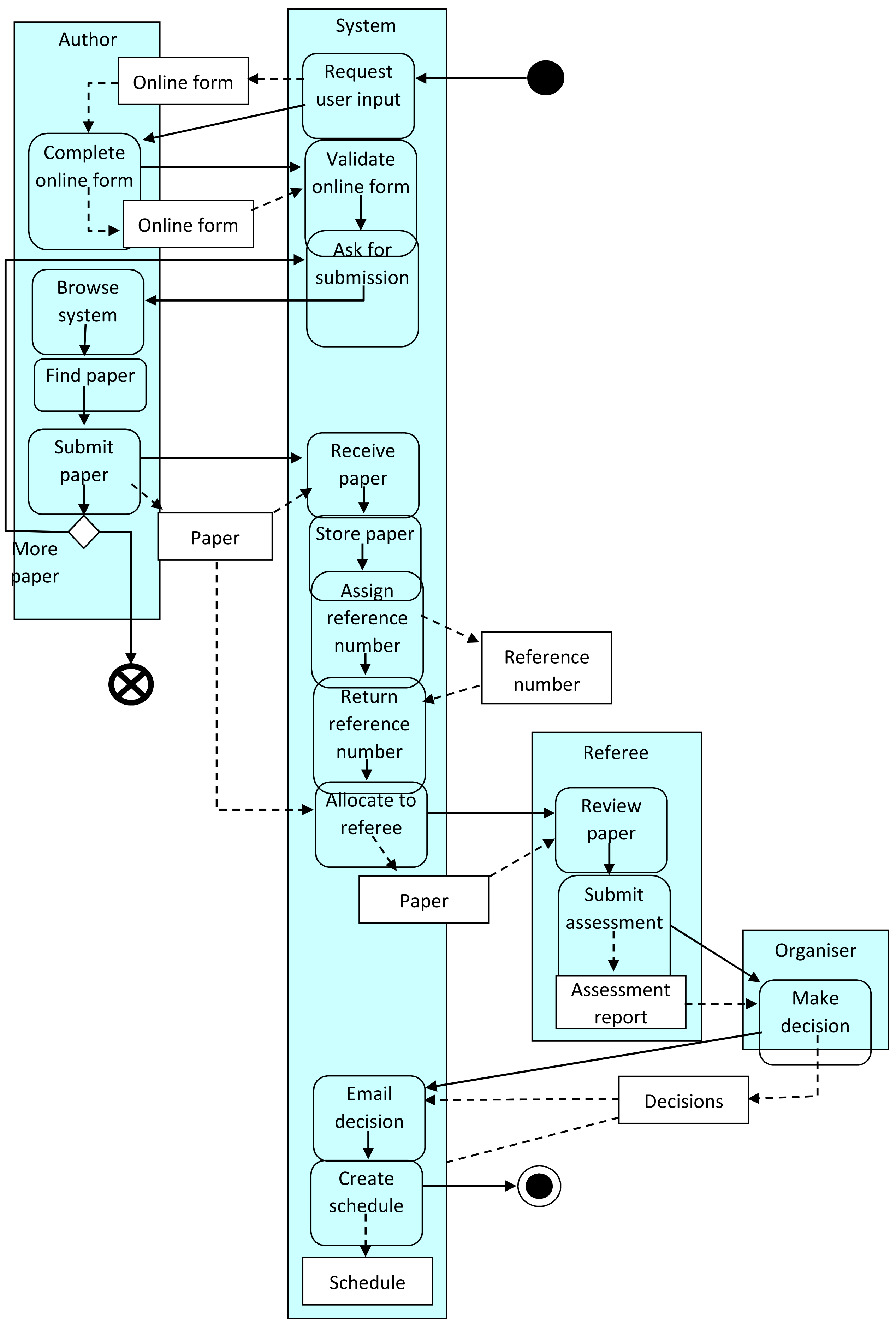
Solution:

- The activity list is given below.
  a) The system **requests the user to input** author name, correspondence address, email and, title of paper if the date is before the deadline.
  b) The author **completes an online form** that contains the data.
  c) The system **validates this data**.
  d) The system **asks the author to submit** the paper, if the data is correct.
  e) The author **browses his system**.
  f) The author **finds the correct paper** on his system.
  g) The author **submits the paper**.
  h) The system **receives the paper**.
  i) The system **stores the paper**.
  j) The system **assigns a reference number** for the paper.
  k) The system **returns the reference number** to the author.
  l) The author goes back to activity 4) if they want to submit another paper.
  m) System **allocates papers to referees** for assessment.
  n) Referee **reviews each paper**.
  o) Referee **submits his assessments** to the system.
  p) The programme organiser **make decisions** on each paper's acceptance.
  q) The system **emails the author** of each paper of the decision.
  r) The system **creates a schedule** for delivering accepted papers at a conference by allocating a date, time and place for the presentation of each paper.
- The action/activity nodes are marked on the activity list as bold font.
- The actors are the name part of the activities.
- In this exercise, the actions are all sequentially ordered as in the activity list. The parallelism and concurrency only occur in the form of different referees can perform their reviews and assessments in parallel.
- The actors are:

(a) Author; (b) The system; (c) Referee; (d) Conference programme organiser;

- The objects involved in this process are:

(a) Online form; (b) paper; (c) paper's reference number; (d) assessment report; (e) Decision; (f) conference schedule.


Author
System
Request user input
Online form
Complete online form
Validate online form
Online form
Ask for submission
Browse system
Find paper
Submit paper
Receive paper
Paper
Store paper
More paper
Assign reference number
Reference number
Return reference number
Referee
Allocate to referee
Review paper
Paper
Submit assessment
Organiser
Assessment report
Make decision
Email decision
Decisions
Create schedule
Schedule

Practical No. 8 : Activity description and diagram for the project

Use case document for a project

Aim : TO draw the activity diagram for the given problem of Use case

Solution :

The following is a use case description of the examination paper preparation support system. Draw a UML activity diagram according to the description.

Use case name: submit question
Participant: lecturer
Entry conditions:

1. The question is ready and stored in a file
2. The lecturer is assigned to the module

Exit conditions:

1. The file is uploaded to the system
2. The module leader is notified of the availability of the question
3. The event is logged by the system

Flow of Events:

1. The lecturer logs into the system by entering his/her username and password;
2. The system checks the username and password;
3. The system displays the list of modules of which he/she is the lecturer, module leader and/or internal examiner;
4. The lecturer selects a module and his/her role in the module as a lecturer;
5. The system prompts the user to enter the file name and location on his/her computer, and additional information if any;
6. The lecturer enters file name and location, and types in the additional information;
7. The lecturer submits the questions and the file is uploaded to the system;
8. The system saves the file;
9. The system confirms the success of uploading the file.
10. The system notifies the module leader of the submission of the questions.

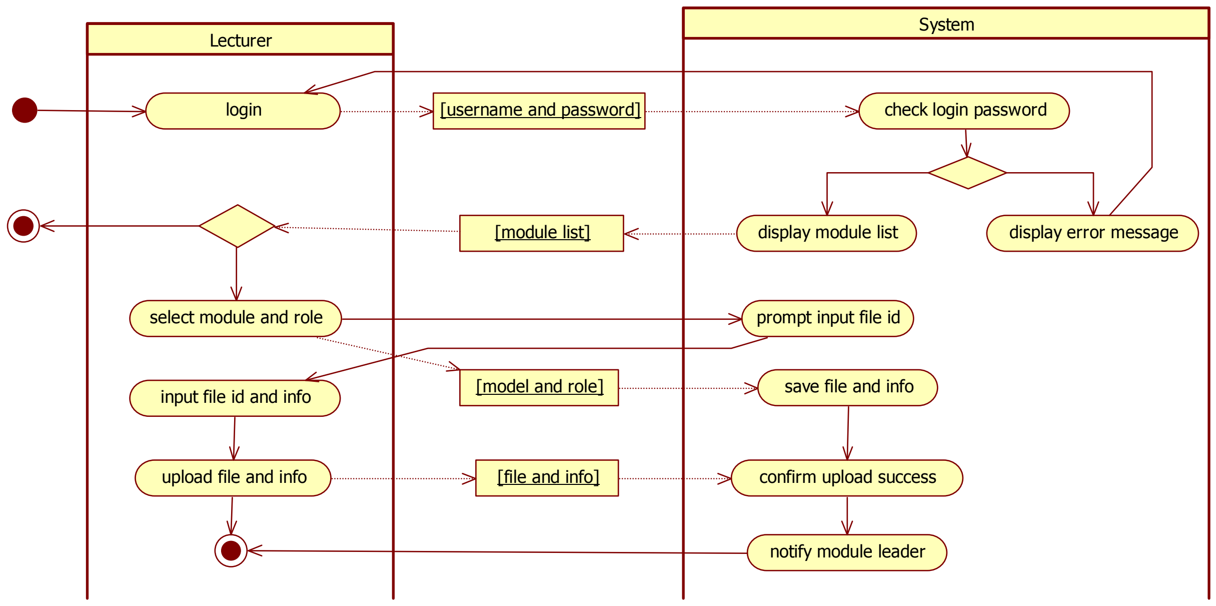
Exceptional conditions and alternative flow of events:

When the username and password is not correct:
3.1: display error message, go back to step 1;
When the lecturer is not listed on the module:
4.1: quit the system;

Special requirements:

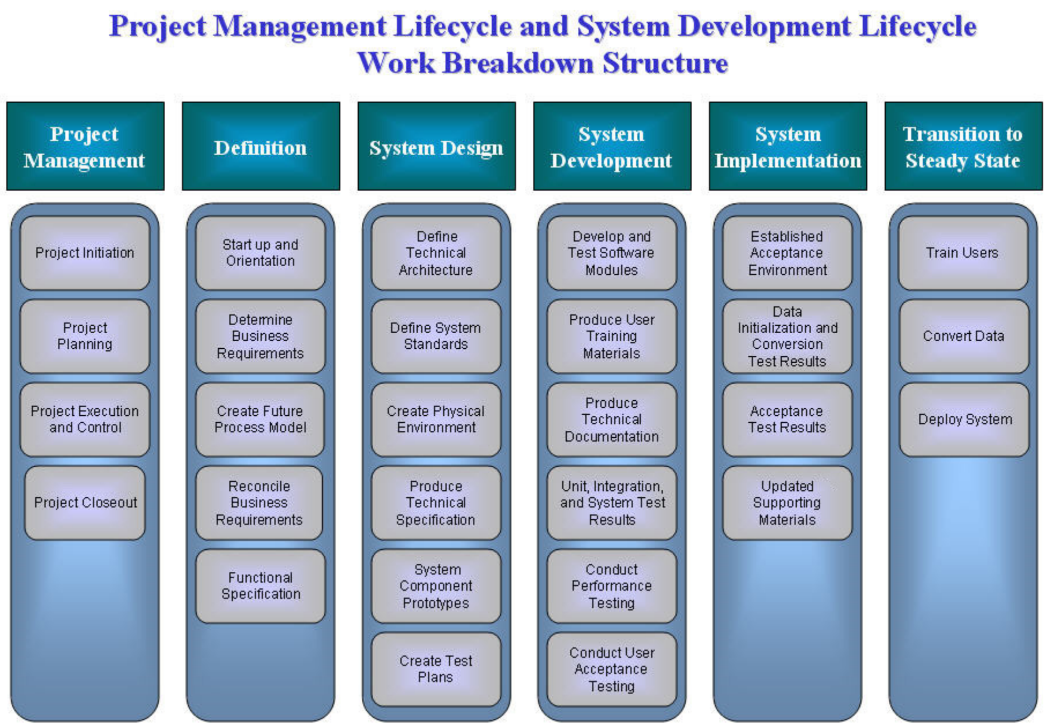
1. The file should be encrypted when transmitted from lecturer's computer to the server
2. The notification of success in uploading the file should be within 20 seconds
3. The event should be recorded in a log file to contain the following information:
   a) name of the lecturer,
   b) date and time of the event,
   c) the name of the event (upload exam question),
   d) the file on the server that stores the questions.

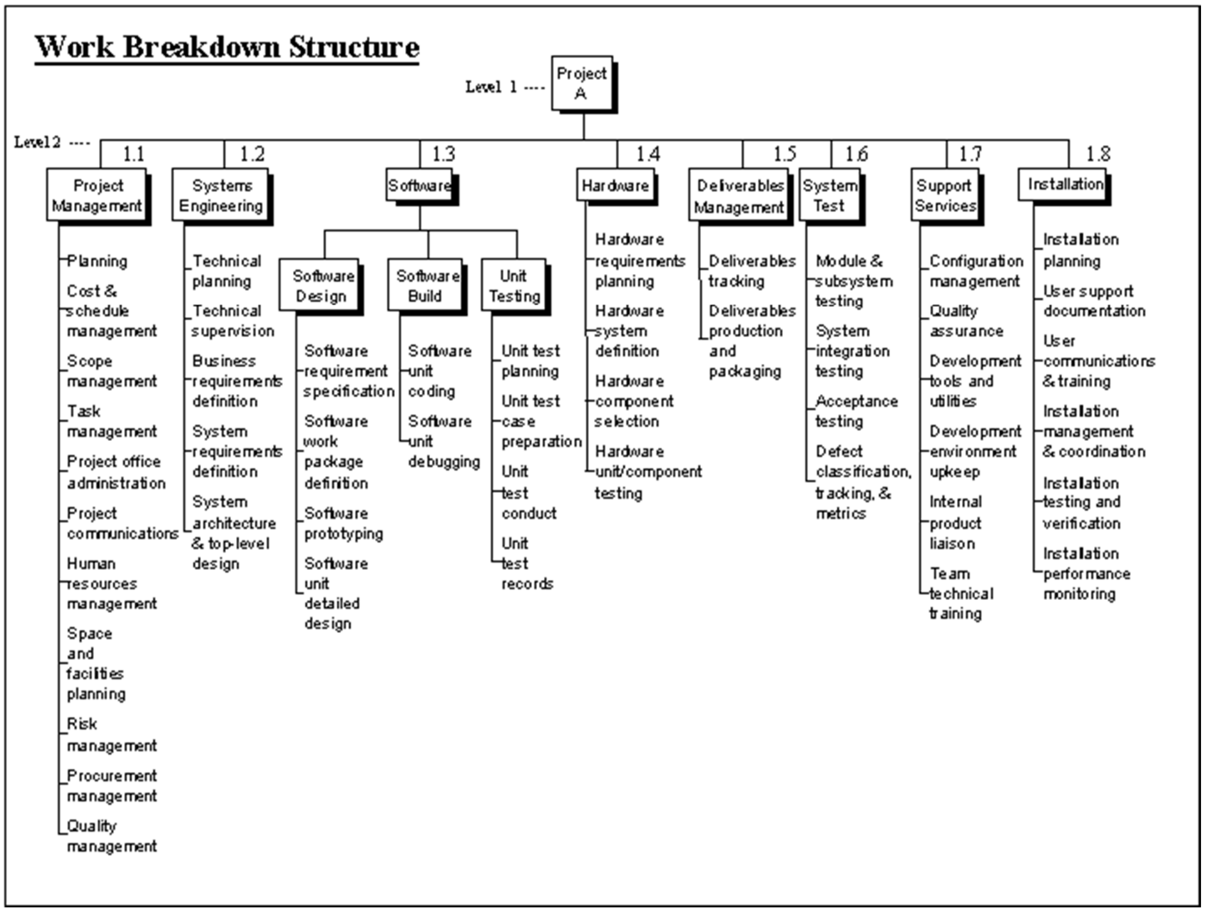
Solution:



Practical No. 9 : Work Breakdown Structure for the given Project

Aim : Create a Work Breakdown Structure


Project Management Lifecycle and System Development Lifecycle
Work Breakdown Structure
Project Management
Project Initiation
Project Planning
Project Execution and Control
Project Closeout
Definition
Start up and Orientation
Determine Business Requirements
Create Future Process Model
Reconcile Business Requirements
Functional Specification
System Design
Define Technical Architecture
Define System Standards
Create Physical Environment
Produce Technical Specification
System Component Prototypes
Create Test Plans
System Development
Develop and Test Software Modules
Produce User Training Materials
Produce Technical Documentation
Unit, Integration, and System Test Results
Conduct Performance Testing
Conduct User Acceptance Testing
System Implementation
Established Acceptance Environment
Data Initialization and Conversion Test Results
Acceptance Test Results
Updated Supporting Materials
Transition to Steady State
Train Users
Convert Data
Deploy System


Work Breakdown Structure
Level 1 ----
Project A
Level2 ----
1.1
Project Management
Planning
Cost & schedule management
Scope management
Task management
Project office administration
Project communications
Human resources management
Space and facilities planning
Risk management
Procurement management
Quality management
1.2
Systems Engineering
Technical planning
Technical supervision
Business requirements definition
System requirements definition
System architecture & top-level design
1.3
Software
Software Design
Software requirement specification
Software work package definition
Software prototyping
Software unit detailed design
Software Build
Software unit coding
Software unit debugging
Unit Testing
Unit test planning
Unit test case preparation
Unit test conduct
Unit test records
1.4
Hardware
Hardware requirements planning
Hardware system definition
Hardware component selection
Hardware unit/component testing
1.5
Deliverables Management
Deliverables tracking
Deliverables production and packaging
1.6
System Test
Module & subsystem testing
System integration testing
Acceptance testing
Defect classification, tracking, & metrics
1.7
Support Services
Configuration management
Quality assurance
Development tools and utilities
Development environment upkeep
Internal product liaison
Team technical training
1.8
Installation
Installation planning
User support documentation
User communications & training
Installation management & coordination
Installation testing and verification
Installation performance monitoring