DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE FILE

Software Engineering

(ESTD - 1999)

Vidya Jyothi Institute of Technology

(An Autonomous Institution)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to J.N.T. University, Hyderabad)

(Aziz Nagar, C.B.Post, Hyderabad -500075)

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

REGULATION:	**R18**
BATCH:	**2018-2022**
ACADEMIC YEAR:	**2019-2020**
PROGRAM:	**B.Tech (COMPUTER SCIENCE AND ENGINEERING)**
YEAR/SEM:	**II/II**
COURSE NAME:	**SOFTWARE ENGINEERING**
COURSE CODE:	**A14510**

PRE REQUISITE : Programming languages C,Python,DS

COURSE COORDINATOR: G.Kalpana

COURSE INSTRUCTORS:

1.Swarna

2.Aruna kumara

3.M.Venkateswarulu

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE FILE INDEX

1	Syllabus
2	Text Books & Other References
3	Time Table
4	Program Outcomes (Po's) ,Program Specific Outcomes (Pso's) & PEO's
5	Mapping Of Course Outcomes (CO's) With Program Outcomes (PO's) & Program Specific Outcomes (PSO's)
6	Academic Calendar
7	Course Schedule
8	Lesson Plan
9	Assignment Questions
10	Mid Question Papers I & II
11	Unit Wise Questions
12	Minutes of Course Review Meeting
13	Lecture Notes
14	Power Point Presentations
15	Semester End Question Papers
16	Extra Topics delivered (if any)
17	Innovations In Teaching and Learning
18	Assessment Sheet – Co Wise (Direct Attainment)
19	Course End Survey Form

Syllabus

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AN ENGINEERING

II Year B.Tech. CSE - II Sem

L	T	P	C
3	0	0	3

SOFTWARE ENGINEERING

Course Outcomes:

At the end of the course student would be able to

1. Outline the framework activities for a givenproject.
2. Examine Right process model for a givenproject.
3. Analyze various system models for a givenContext.
4. Understand various testing techniques for a givenproject.
5. Identify various risks in projectdevelopment.

UNIT I: Introduction to Software Engineering: The evolving role of software, Changing Nature of Software, Software myths. A Generic view of process: Software engineering- A layered technology, a process framework, The Capability Maturity Model Integration (CMMI), personal and team process models.

UNIT II: Process Models: The waterfall model, Incremental process models, Evolutionary process model, Unified process model,agile process model. Software Requirements: Functional and non-functional requirements, the software requirements document. Requirements engineering process: Feasibility studies, Requirements elicitation and analysis, Requirements validation, Requirements management.

UNIT III: System models: Context Models, Behavioral models, Data models, Object models, structured methods. Design Engineering: Design process and Design quality, Design concepts, the design model, Modeling component level design: design class based components, conducting component level design. User interface design: Golden rules.

UNIT IV: Testing Strategies: A strategic approach to software testing, test strategies for conventional software, Black-Box and White-Box testingtechniques, Validation testing, System testing.

Product Metrics: Software Quality, Metrics for Analysis Model- function based metrics, Metrics for Design Model-object oriented metrics, class oriented metrics, component design metrics, Metrics for source code, Metrics for Testing, Metrics for maintenance.

UNIT V: Risk Management: Reactive vs. Proactive Risk strategies, software risks, Risk identification, Risk projection, Risk refinement, RMMM, RMMM Plan.

Quality Management: Quality concepts, Software Reviews, Formal technical reviews, Software reliability, The ISO 9000 quality standards.

Text Books And Other References

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Text Books	
1	Software Engineering, A practitioner's Approach- Roger S. Pressman, 6th edition,TMGH
2	Software Engineering- Sommerville, 7th edition, Pearson education.
Suggested / Reference Books	
1.	Software Engineering- K.K. Agarwal&Yogesh Singh, New Age International Publishers.
2.	Software Engineering, an Engineering approach- James F. Peters, WitoldPedrycz, JohnWiely.
3.	Systems Analysis and Design- ShelyCashmanRosenblatt,Thomson Publications.
4.	Software Engineering principles and practice- Waman S Jawadekar, The McGraw-Hill Companies.
Other Resources	
1	http://nptel.iitm.ac.in/
2	www.computersocity.org
3	http://www.cse.iitm.ac.in/~sdas/
4.	http://freevideolectures.com/Course/2275/Software Engineering
5.	http://nptel.iitk.ac.in/courses/Comp_Sci_Engg/IIT%20Madras/software engineering.htm

Time Table

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Sec: **CSE-D**

Year/Sem: **II - II** W.E.F: **09/12/2019** ROOM NO: **C305**

DAY	9.00 – 9.55	9.55 – 10.50	10.50 – 11.45	11.45 – 12.30	12.30 – 1.25	1.25 – 2.20	2.20 – 3.15	3.15-4.05
MON	SE	CO	DBMS	LUNCH	ES	JAVA	DAA(T)	**Value Added**
TUE	ES	SE	JAVA		DAA	DBMS(T)	CO(T)	**CISCO**
WED	DAA	JAVA	SE		DBMS LAB			**Value Added**
THU	CO	JAVA	DBMS		JAVA LAB			**NPTEL**
FRI	DBMS	DAA	SE		JAVA(T)	CO(T)	MC-II	**NPTEL**
SAT	CO	DBMS	DAA		MC-II	SE(T)	DAA/ DBMS	**CISCO**

Subject **Name of the Faculty**

JAVA	Java Programming	Dr.B.Vijaya Kumar
DAA	Design and Analysis of Algorithms	Ms. K.Samatha
CO	Computer Organization	Ms.K.Neha
DBMS	Database Management Systems	Mr.Zeeshan
SE	Software Engineering	Ms.G.Kalpana
ES	Environmental Science	Ms. Y.Suneetha
MC-II	Professional Communication	Ms.Hepsiba
JAVA LAB	Java Programming Lab	Dr.B.Vijaya Kumar/ Ms.K.Samatha/ Ms.M.Kavya
DBMS LAB	Database Management Systems lab	Ms. B.Sailaja / Mr.Zeeshan/ Ms.K.Keerthi

Class Incharge	Ms.K.Samatha
II YEAR Coordinator	Ms. B.Sailaja

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-50075.

[illegible]

Time Table I/C **H.O.D**

Programme Educational Objectives(PEO's) & Programme Specific Outcomes (PSOs) & PEO's

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Programme Outcomes (PO's):

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modelling to complex engineering activities, with an understanding of the limitations.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Program Specific Objectives (PSO's):

PSO1: Successful career: Able to excel in professional career with sound problem solving skills for providing IT solutions by proper plan, analysis, design, implementation and validation.

PSO2: Lifelong learning: Able to be an employee, entrepreneur and researcher using scientific, technical and communication base to cope with latest technology and zest for higher studies.

Program Educational Objectives (PEO's):

PEO1: Graduates will be able to software professionals

PEO2: Graduates will be able to develop analytical and computational ability to solve software problems, by applying innovative technical tools in ever changing world.

PEO3: Graduates will be able to work in multidisciplinary project teams with effective communication skills and leadership qualities.

PEO4: Graduates will be able to embrace life-long learning with professional ethics.

Mapping of Course Outcomes(Co's) with Program Outcomes(PO's) &Program Specific Outcomes(PSO's)

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Mapping of CO's and PO's and Mapping of CO's with PSO's

COURSE OUTCOMES:

After completing this course the student must demonstrate the knowledge and ability to	
1.	Analyze and apply the framework activities for a given project.
2.	Choose a process model to apply for given project requirements
3.	Design various system models for a given scenario.
4.	Apply various testing techniques for a given project.
5.	Identify various risks in project development.

CO –PO MAPPING:

	PO 1	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**
CO 1	3	2	2	2	2	3	3	3	3	3	3	3
CO 2	3	3	3	3	2	3	3	3	3	3	3	3
CO 3	3	2	2	2	3	3	3	3	2	3	3	3
CO 4	3	2	2	2	3	3	3	3	2	2	3	3
CO 5	2	2	3	2	1	3	3	3	2	3	3	2
AVG	**2.8**	**2.2**	**2.4**	**2.2**	**2.2**	**3**	**3**	**3**	**2.4**	**2.8**	**3**	**2.8**

CO - PSOMAPPING:

	PSO1	PSO2
CO1	3	3
CO2	3	3
CO3	3	3
CO4	3	3
CO 5	3	3
AVG	3	3

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-50075.

Course Coordinator.

CSE-HOD

Academic Calendar

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved by A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

II B.Tech I & II Semester Academic Calendar for the Academic Year 2019-20

II YEAR I SEMESTER		Commencement of Class Work 17.06.2019	
	From	To	Duration
I Spell of Instruction	17.06.2019	10.08.2019	8 WEEKS
I Mid Examinations	13.08.2019	17.08.2019	4 DAYS
II Spell of Instruction	19.08.2019	05.10.2019	7 WEEKS
Dussehra Holidays	07.10.2019	12.10.2019	1 WEEK
II Spell of Instruction Continuation	14.10.2019	19.10.2019	1 WEEK
II Mid Examinations	21.10.2019	24.10.2019	4 DAYS
Practical Examinations	25.10.2019	29.10.2019	4 DAYS
Betterment Examinations	30.10.2019	01.11.2019	3 DAYS
End Semester Examinations	02.11.2019	18.11.2019	2 WEEKS
Supplementary Examinations	19.11.2019	04.12.2019	2 WEEKS
Mid-II Examinations (For Lateral Entry)	18.11.2019	21.11.2019	4 DAYS
Practical Examinations (For Lateral Entry)	22.11.2019	26.11.2019	4 DAYS
Betterment Examinations (For Lateral Entry)	27.11.2019	30.11.2019	4 DAYS
II YEAR II SEMESTER		Commencement of Class Work 02.12.2019	
I Spell of Instruction	02.12.2019	10.01.2020	6 WEEKS
Pongal Holidays	11.01.2020	15.01.2020	5 DAYS
Technical/Sports fest	16.01.2020	18.01.2020	3 DAYS
I Spell of Instruction Continuation	20.01.2020	01.02.2020	2 WEEKS
I Mid Examinations	03.02.2020	08.02.2020	1 WEEK
II Spell of Instruction	10.02.2020	04.04.2020	8 WEEKS
II Mid Examinations	06.04.2020	09.04.2020	4 DAYS
Practical Examinations	13.04.2020	17.04.2020	4 DAYS
Betterment Examinations	18.04.2020	22.04.2020	4 DAYS
End Semester Examinations	23.04.2020	08.05.2020	2 WEEKS
Supplementary Examinations	11.05.2020	23.05.2020	2 WEEKS
Commencement of classes will be from	15.06.2020		

[illegible]
DIRECTOR

Course Schedule

Text Books:

1. Software Engineering, A practitioner's Approach- Roger S. Pressman,

6 th edition McGraw Hill International Edition.

2. Software Engineering- Sommerville, 7th edition, Pearson education.

Unit No	TOPIC	Text Book 1/Page No.	Text Book 2/page No	Reference Book
1	Introduction to Software Engineering:	T1: 3-30		
	A Generic view of process:	T1: 31-35		
2	Process Models:	T1: 38-48,67-89		
	Software Requirements:	T1:120-127	T2:139-163	
	Requirements engineering process	T1: 127-144	T2:164-189	
3	System models:	T1: 149-182	T2:191-209	
	Design Engineering:	T1: 216-237, 313-317		
	component level design:	T1: 277-307		
4	Testing Strategies	T1: 482-502		
	Product Metrics	T1: 614-641, 667-686		
5	Risk Management:	T1: 745-757		
	Quality management	T1: 399-403		

Lesson Plan

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AN ENGINEERING

Lesson Plan

S.No	TOPIC	Teaching Learning Process
	UNIT 1(CO1)	
1	Introduction to Software Engineering, Evolving role of s/w	Chalk & Board
2	What is Software, changing nature of s/w	Chalk & Board
3	s/w applications, s/w myths	Chalk & Board
4	Layered technology	Chalk & Board
5	A process frame work	Power point presentation
6	CMMI, A generic process model	Chalk & Board
7	Process patterns, pattern types	Chalk & Board
8	Process Assessment and Improvement	Chalk & Board
9	Personal Software Process (PSP)	Chalk & Board
10	Team Software Process (TSP)	Power point presentation
	UNIT II(CO2)	
11	Prescriptive Models, The water fall model	Chalk & Board
12	The incremental process Models	Chalk & Board
13	Evolutionary process models	Power point presentation
14	Agile process model	Power point presentation
15	S/w requirements	Power point presentation
16	Functional and Non Functional requirements	Power point presentation
17	The s/w requirement document	Chalk & Board , case based learning
18	Requirements engineering process	Power point presentation
19	feasibility study, Elicitation and analysis	Power point presentation
20	Requirements validation	Power point presentation
21	Requirements management	Power point presentation
	UNIT III(CO3)	
22	System models, context models	Chalk & Board
23	Behavioral models	Chalk & Board
24	Data models,	Power point presentation
25	Object models, structure model	Chalk & Board ,case based learning
26	Design Engineering, Introduction	Power point presentation
27	Design process	Power point presentation
28	Design quality	Power point presentation
29	Design concepts	Power point presentation
30	The design model	Power point presentation
31	Modeling component level design	Power point presentation

32	Design class based components	Power point presentation
33	Conducting component level design	Power point presentation
34	User interface design, golden rules	Power point presentation
	UNIT IV(CO4)	
35	Software testing Strategies	Chalk & Board
36	A strategic approach to testing	Chalk & Board
37	Test strategies for conventional software	Chalk & Board
38	Black box and white box testing	Chalk & Board
39	Validation testing	Power point presentation
40	System testing	Power point presentation
41	Product metric ,s/w quality	Power point presentation
42	Metrics for Analysis model	Power point presentation
43	Function based metrics	Chalk & Board, think share pair
44	Metrics for Design Model	Power point presentation
45	OO Metrics, class oriented	Power point presentation
46	Component design metrics	Power point presentation
47	Metrics for source code	Power point presentation
[illegible]	Metrics for maintenance	Power point presentation
	UNIT V(CO5)	
49	Risk management	Chalk & Board
50	Reactive Vs proactive risk strategies	Chalk & Board
51	Software risks	Power point presentation
52	Risk Identification	Power point presentation
53	Risk Projection, Risk refinement	Power point presentation
54	RMMM,RMMM plan	Chalk & Board, case based learning
55	Quality management	Power point presentation
56	Quality concepts	Power point presentation
57	S/w reviews	Power point presentation
58	Formal Technical Reviews	Power point presentation
59	S/w Reliability, The ISO 9000 quality standard	Power point presentation

Assignment Questions

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Assignment -1:

SI.No	Question Number	Marks	CO	BL	PO's
1	Define software and explain the various characteristics of software?	5	1	L1	1-12
2	Explain in detail the capability Maturity Model Integration (CMMI)?	5	1	L2	1-12
3	Describe with the help of the diagram discuss in detail waterfall model	5	2	L2	1-12
4	Explain briefly on (a) the incremental model (b) The RAD Model?	5	2	L2	1-12
5	Compare functional requirements with nonfunctional requirements?	5	1	L4	1-12

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Assignment 2:

SI.No	Question Number	Marks	CO	BL	PO's
1	Discuss briefly on behavioral models?	5	3	L2	1-12
2	Describe the way of conducting a component level design?	5	3	L2	1-12
3	What is software testing strategy? Explain the characteristics in detail.	5	4	L1	1-12
4	Compare validation testing and system testing in detail.	5	4	L4	1-12
5	Write a short notes on a) Cost of quality b) ISO 9000 quality standards c) Software reviews d) Review guidelines	5	5	L1	1-12

Mid Question Papers

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

II Year B.Tech II Semester 1st Mid Exam

Branch: CSE	Duration: 90Min
Sub: Software Engineering	Marks: 20
Date:18-02-02020	Session: AN

Course Outcomes:

1.Outline the framework activities for a given project
2.Apply right process model for a given project
3.Design various system models for a given Context
4.Apply various testing techniques for given project
5.Identify various risks in project development

Bloom's Level:

Remember	I
Understand	II
Apply	III
Analyze	IV
Evaluate	V
Create	VI

PART-A (3Q×2M =6Marks)		Course Outcomes		Bloom's Level	Marks
ANSWER ALL THE QUESTIONS		CO	PO		
1.i)	Define legacy software.	1	1	I	2
	[OR]				
ii)	List out the categories of computer software.	1	1	I	2
2.i)	What is the purpose of feasibility study?	2	2	II	2
	[OR]				
ii)	What is the goal of Agile Modeling?	2	2	II	2
3.i)	Define System Modeling.	3	2	II	2
	[OR]				
ii)	Draw a neat Context Diagram for ATM system.	3	3	III	2

PART-B (5+5+4= 14 Marks)		Course Outcomes		Bloom's Level	Marks
ANSWER ALL THE QUESTIONS		CO	PO		
4.i.a)	Classify the characteristics of Software and Hardware.	1	2	III	2
b)	Explain in detail about capability Maturity Model Integration (CMMI)?	1	2	IV	3
	[OR]				
ii.a)	What is Software engineering?Explain its layered technology with diagram.	1	2	II	2.5
b)	Discuss about Personal Software Process.	1	3	III	2.5
5. i.a)	Explain the Incremental Model with neat diagrams.B60	2	3	IV	2.5
b)	Write about Requirement Management indetail.	2	3	III	2.5
	[OR]				
ii.a)	Explain Non Functional Requirements.	2	3	IV	2.5
b)	Explain in detail about Requirement Validation.	2	3	IV	2.5
6.i)	Discuss about various object models with examples.	3	4	IV	4
	[OR]				
ii)	Write about any two Behavioural models.	3	4	IV	4

VJIT(A)

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

II Year B.Tech II Semester 1st Mid Exam

Branch: CSE	Duration: 90Min
Sub: Software Engineering	Marks: 20
Date:18-02-02020	Session: AN

Course Outcomes:

1.Outline the framework activities for a given project
2.Apply right process model for a given project
3.Design various system models for a given Context
4.Apply various testing techniques for given project
5.Identify various risks in project development

Bloom's Level:

Remember	I
Understand	II
Apply	III
Analyze	IV
Evaluate	V
Create	VI

	PART-A (3Q×2M =6Marks)	Course Outcomes		Bloom's Level	Marks
	ANSWER ALL THE QUESTIONS	CO	PO		
1.i)	What is Software Engineering?	1	1	I	2
	[OR]				
ii)	Write phases of Team software process.	1	1	I	2
2.i)	List out the techniques of Requirement Elicitation and Analysis.	2	2	II	2
	[OR]				
ii)	Define Requirement Management.	2	2	II	2
3.i)	Draw a neat data flow diagram for insulin pump.	3	2	III	2
	[OR]				
ii)	List out the various types of System Models.	3	2	III	2

	PART-B (5+5+4= 14 Marks)	Course Outcomes		Bloom's Level	Marks
	ANSWER ALL THE QUESTIONS	CO	PO		
4.i.a)	Explain the Changing Nature of Software.B56	1	2	III	2
b)	What is "Software myth"? Explain the types of myths.	1	2	III	3
	[OR]				
ii.a)	Write about evolving role of software.	1	2	II	3
b)	Explain The Process Framework in detail.	1	3	III	2
5. i.a)	Explain Requirement Engineering Process with neat diagram.	2	3	III	3
b)	Explain spiral models in detail with diagram.	2	3	III	2
	[OR]				
ii.a)	What is Agile Process ? Explain any one model in detail.	2	3	III	2.5
b)	Explain Waterfall Model along with diagram and advantages.	2	3	IV	2.5
6.i)	Write short notes on a)Data flow diagrams b)State machine diagrams.	3	4	IV	4
	[OR]				
ii)	Explain the Components of case tool for Structured Method support.	3	4	IV	4

VJIT(A)

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

II Year B.Tech II Semester II nd Mid Examination, October-2020

Subject : Software Engineering — BRANCH: CSE

Time : 1 Hour — Max marks: 20

Note:

This question paper contains two ***Parts A and B***

Part A is compulsory which carries 6 Marks

Part B consists of 4 questions. Answer any two questions.

Bloom Levels:	
Remember	L1
Understand	L2
Apply	L3
Analyze	L4
Evaluate	L5
Create	L6

	PART-A (1X6=6M) **COMPULSORY QUESTION**	**Bloom Levels**	**Marks**
I a)	Explain the design concepts in software engineering.	L2	6M
	[OR]		
b)	Explain the goals of the user interface design.	L2	6M

	PART-B (2X7=14) **ANSWER ANY TWO QUESTIONS**	**Bloom Levels**	**Marks**
2)	Explain Conventional software testing strategy with the diagram.	L2	7M
3)	Write short notes on a) Acceptance testing b) Unit testing	L2	7M
4)	Differentiate between reactive and proactive risk strategies in detail.	L4	7M
5)	Explain the activities under Formal Technical Reviews.	L2	7M

VJIT(A)

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad-500075)

II Year B.Tech II Semester II nd Mid Examination,October-2020

Subject : Software Engineering	BRANCH: CSE
Time : 1 Hour	Max marks: 20

Note:

This question paper contains two **Parts A and B**

Part A is compulsory which carries 6 Marks

Part B consists of 4 questions. Answer any two questions.

Bloom Levels:	
Remember	L1
Understand	L2
Apply	L3
Analyze	L4
Evaluate	L5
Create	L6

PART-A (1X6=6M) **COMPULSORY QUESTION**		**Bloom Levels**	**Marks**
I a)	Elaborate modeling component level design.	L2	6M
	[OR]		
b)	Discuss briefly the following fundamental concepts of design: a) Abstraction b) Modularity c) Information hiding	L2	6M

PART-B (2X7=14) **ANSWER ANY TWO QUESTIONS**		**Bloom Levels**	**Marks**
2)	Compare Black Box testing with white Box testing in detail.	L4	7M
3)	Explain the metrics for source code and maintenance?	L2	7M
4)	Explain RMMM in RMMM plan.	L2	7M
5)	Write short notes on a) Risk Identification , b) Risk refinement ,c) Risk projection	L2	7M

VJIT(A)

Unit wise Questions

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT I

PART – A Short Answer Questions

1. **Define** software.
2. **What** is Software Engineering?
3. **List** the categories of computer software.
4. **State** management myths.
5. **What** are customer myths?
6. **List** practitioner's myths.
7. **Discuss** the architecture of layered technology.
8. **List** all the umbrella activities in process framework.
9. **What** is process pattern?
10. **List** the types of software models.
11. **List** the measures of software tracking and control.
12. **List** the models in CMMI.
13. **Write** the levels in continuous model in CMMI.
14. **Write** short notes onstaged model in CMMI.
15. **What** is open source software?
16. **List** out the process standards.
17. **Define** legacy software.
18. **What** is the need of software engineering?
19. **What is** Team software process (TSP)?List out the phases of it.
20. **What is** Personal software process (PSP)?List out the phases of it.

PART – B Long Answer Questions

1. **a.Explain**the changing nature of software.
   b.Write about evolving role of software.
2. **a)Define** software and explain the various characteristics of software.

b) **Explain** process framework in detail.

3. **What** is "Software myth"? Explain various types of Myths.
4. **Explain** in detail about Capability Maturity Model Integration (CMMI).
5. **Explain** Personal and Team Software Process models.

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT II

PART – A Short Answer Questions

1. **What** are the merits and demerits of water fall models?
2. **What** are thedisadvantages of prototype model?
3. **What** is spiral model and list out advantages?
4. **What is** concurrent development model?
5. **Define** agile process.
6. **List** out the human factors of agile process.
7. **Write** short notes onfunctional requirement.
8. **List** out non-functional requirements
9. **What** aredomain requirements?
10. **List** out the classification of nonfunctional requirements.
11. **What** is System requirements specification (SRS)?
12. **What** are the sub process of requirement engineering process?
13. **What** is feasibility study?
14. **List** out the techniques ofrequirement elicitation and analysis.
15. **What** are the process activities in requirement elicitation and analysis?
16. **What** are view points?
17. **What** are the characteristics of effective interviewers?
18. **What** is usecase? Give an example scenario.
19. **What** is ethnography?
20. **What** is requirements management?

PART – B Long Answer Questions

1. **a.Explain** waterfall model along with diagram and advantages.

   b.Explain the incremental process models with neat diagrams.
2. **a.Explain**the evolutionary process models.

   b.Listand explain any three agile process models.
3. **Compare** Functional Requirements with Non-Functional Requirements.
4. **Explain** Requirement Engineering Process.
5. **a.Explain** in detail about requirement validation.

   b.Write about Requirement Management in detail.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT III

PART – A (Short Answer Questions)

1. **Define** system modeling.
2. **What is** context model?
3. **What** is object model?
4. **What** are the system models for analysis?
5. **Draw** a neat context diagram for ATM system.
6. **Draw** a neat data flow diagram for insulin pump.
7. **Draw** the state machine model for simple microwave oven.
8. **Construct** the semantic data model for library system.
9. **Why** design is important in design engineering?
10. **Write** short notes ondesign model.
11. **List** the design concepts.
12. **Justify** the importance of refactoring.
13. **Write** short notes on coupling.
14. **List** out the steps for conducting component level design.
15. **Write** short notes on cohesion.
16. **Design** the class based components.
17. **List** out the golden rules for interface design.
18. **Write** short notes on interface design steps.
19. **List** out all the design issues.
20. **What** is user interface design?

PART – B (Long Answer Questions)

1. a) **Explain** any 3 system models with example.

   b) **Explain** the components of case tool for structured method support.
2. **a. Discuss** briefly about behavioural models with examples.

   b. Discuss about various object models with examples.
3. **Discuss** briefly the following fundamental concepts of design:

   a) Abstraction b) Modularity c) Information hiding
4. **a. Explain** the goals of the user interface design.

   b. Explain the design concepts in software engineering.
5. **a. Elaborate** modelling component level design.

   b. Describe the way of conducting a component level design.

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT IV

PART – A (Short Answer Questions)

1. **What** is testing?
2. **What** is the importance of test strategy?
3. **List** out the activities incorporate in test strategy.
4. **Define** black box and white box testing?
5. **What** is Regression testing?
6. **What** are the various types of integration testing?
7. **What** is Smoke testing?
8. **Write** short notes on Validation testing?
9. **What** is Cyclomatic Complexity?
10. **What** is Glass-box testing?
11. **What** is Basis path testing?
12. **What** is the metric for software quality?
13. **List** out software quality factors?
14. **List** out the metrics for object oriented design
15. **What** is function point?
16. **Define** the terms Measure and Metrics.
17. **Write** the metrics for source code .
18. **List** out the component level design metrics.
19. **What** is software metric?
20. **Compute** the function point value for a project with the following information domain characteristics:

    Number of user inputs: 32

    Number of user outputs: 60

    Number of user inquiries: 24

    Number of files: 8

    Number of external interfaces: 2

Assume that all complexity adjustment values are average.

PART – B (Long Answer Questions)

1. a)**Explain** Conventional software testing strategy? Along with the diagram
   b)**Explain** the methods of White box testing
2. a)**Compare** Black Box testing with white Box testing in detail.
   b)**Write** short notes on a)Acceptance testing b)Unit testing
3. **Compare** validation testing and system testing.
4. a)**Explain** the metrics for Software quality.
   b) **Explain** metrics for analysis model

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT V

PART – A (Short Answer Questions)

1. **What** is Risk management?
2. **What** are the various types of Software Risks?
3. **Define** Reactive and Proactive Risks?
4. **Whet** is meant by Risk identification?
5. **What** is meant by Risk Projection?
6. **What** is the goal of RMMM?
7. **What** are the issues of RMMM?
8. **Write** short notes on risk refinement.
9. **What is** RMMM Plan?
10. **Write** short notes on Quality concepts?
11. **What** is quality management?
12. **What** are software reviews?
13. **Write** short notes on formal technical reviews?
14. **What** is Risk mitigation?
15. **What** is RMMM?
16. **What** is software reliability?
17. **Demonstrate** risk identification?
18. **Discuss** what kind of risks can be documented using RIS(Risk Information Sheet).
19. **Write** about software reviews in brief?
20. **What** is the purpose of formal technical reviews.

PART – B (Long Answer Questions)

1. Differentiate between reactive and proactive risk strategies in detail

2. a)Explain RMMM in RMMM plan

b) **Explain** Software reliability

3. **Write** a short notes on

    a) Cost of quality

    b) ISO 9000 quality standards

    c) Software reviews

    d) Review guidelines

4. **Write** short notes on the following

    a) Risk Identification

    b) Risk refinement

    c) Risk projection

5. **a)Explain** software reviews in detail

    b) **Explain** the activities under Formal Technical Reviews.

Minutes of Review Meeting

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Meeting 1

Date: 31/12/2019

Details of Meeting No – 1	
Date of Meeting	31/12/2019
Member's Present	1.Swarna Section – A 2.Aruna kumari CSE,Section- B 3.M.Venkateswarulu CSE,Section-C 4.Mrs G.Kalpana CSE,Section-D
Details	Points discussed in the meeting: • Discussion on Preparation of Unit wise questions and give assignment to students • Discussion on Teaching Learning Practices • Discussion on Status of Syllabus coverage of Mid I and instructions to complete the syllabus
Signatures	1. [illegible] 2. [illegible] 3. [illegible]

[illegible]

Course Coordinator

[illegible]

CSE-HOD

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-500075.

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Meeting 2

Date: 26/2/2020

Details of Meeting No – 2	
Date of Meeting	26/2/2020
Member's Present	1.Swarna Section – A 2.Aruna kumar~~e~~a CSE,Section- B 3.M.Venkateswarulu CSE,Section-C 4.Mrs G.Kalpana CSE,Section-D
Details	• Discussion on Preparation of Unit wise questions and give assignment to students • Discussion on Status of Syllabus coverage of Mid II and instructions to complete the syllabus • Discussion on Identify slow and advanced learners • Discussion on Plan for remedial classes
Signatures	1. [illegible] 2. [illegible] 3. [illegible]

[illegible]

Course Coordinator

[illegible]

CSE-HOD

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-500075.

Lecture Notes

UNIT-I S E

Prepared by
G. Kalpana
Dept of MCA

Introduction to software engineering :-

> Computer software is the single most important technology on the world stage.

> Now-a-days national infrastructures and utilities rely on computer-based systems; industrial manufacturing and distribution is completely computerized. So, the software plays vital role in the field of technology.

⇒ where as in the year 1950's no one could have predicted that s/w would become an indispensable technology for business, science and ~~technology~~. ~~like engineering. No one could have fore seen that~~ the s/w would become embedded in systems like transportation, medical, telecommunications, military, industrial and entertainment etc.

⇒ Computer software :- It is not just the program but also associated documentation and configuration data that is needed to make these programs operate correctly.

⇒ Computer software is the product that software professionals build and then support over the long term.

→ generally, software engineers build and support the software and every one in the industrialized world uses it either directly (or) indirectly.

→ "Software products" may be developed for a particular customer (or) may be developed for a general market

→ software engineering is an engineering discipline whose focus is the cost effective development of high-quality software systems.

→ Producing and maintaining the software cost-effectively is essential for functioning of national and international economies.

→ Definition for software engineering :- "software engineering" is an engineering discipline which is concerned with all aspects of software production. i.e (from early stages of system specification to maintaining the system after it has gone into use)

→ The ~~word~~ notation software engineering was first proposed in 1968 at a conference held to discuss the 'software crisis'. (s/w development was not good enough and much cost, unreliable, was difficult to maintain & performed very poorly. So s/w development was in crisis).

The evolving role of software :-

→ Today, software takes on a dual role. It is both a Product and a vehicle for delivering a product.

→ As a product, it delivers the computing potential embodied by computer hardware or by a network of computers that are accessible by local hardware.

→ Example for a product is cellular phone, it is an informatic transformer. Software resides within a cellular phone.

→ As the vehicle for delivering the product, software acts as the basis for the control of the computer (operating systems); the communication of information (networks), and the creation and control of other programs (software tools and environments).

→ The role of computer software has undergone significant changes over a span of little more than 50 years. Dramatic improvements in hardware performance, profound changes in computing architectures, vast increases in memory and storage capacity, and a wide variety of exotic input and output options have all precipitated more sophisticated and complex - computer based systems.

→ Sophistication and complexity can produce dazzling results when a system succeeds, but they can also pose huge problems for those who must build complex systems.

→ Today, a huge software industry has become a dominant factor in the economies of the industrialized world.

The lone programmer of an earlier era has been replaced by team of s/w specialists, each focusing on one part of the technology required to deliver a complex application. And yet, the questions that were asked of the lone programmer are the same questions that are asked when modern computer based systems are built:

→ • why does it take so long to get software finished?

→ • why are development costs so high?

→ • why can't we find all errors before we give the software to our customers?

→ • why do we spend so much time and effort maintaining existing programs?

→ • why do we continue to have difficulty in measuring progress as software is being developed and maintained?

→ These questions and many others demonstrate the industry's concern about software and the manner in which it is developed - a concern that has lead to the adoption of software engineering practice.

Software :-

→ A textbook definition of software might take the following form :

1. Software is instructions (computer programs) that when executed provide desired features, function, and performance.
2. Software is data structures that enable the programs to adequately manipulate information.
3. Documents that describe the operation and use of the programs.

→ Software is a logical rather than a physical system element. Therefore, software has characteristics that are considerably ~~different than those of hardware~~

diff h/w or s/w.

→ 1. Software is developed (or) engineered : it is not manufactured in the classical sense :- Although some similarities exist between development and hardware manufacturing, the two activities are fundamentally different. In both activities, high quality is achieved through good design. Both activities require the construction of a product but approaches are different.

→ 2. Software doesn't "wearout" :- Hardware exhibits relatively high failure rates in its life. (due to manufacturing defects etc). Defects are corrected then failure rate is steady - state level for some period of time. As time passes, the failure rate rises again as h/w components suffer from dust, abuse, temparature extremes etc., This stated simply h/w begins to wearout.

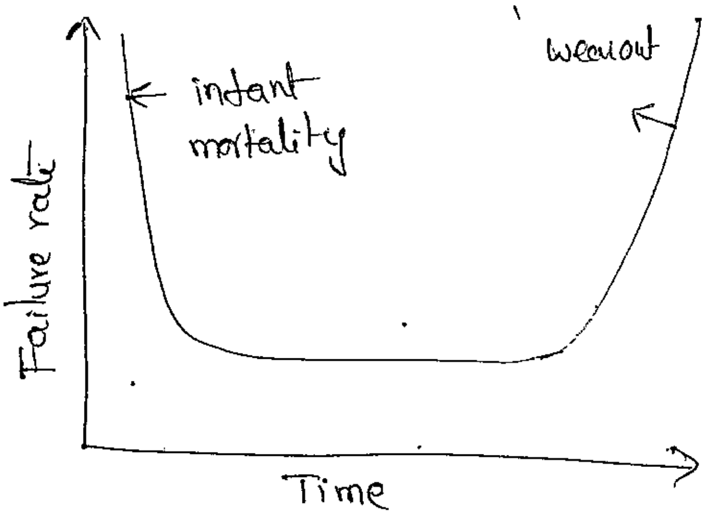


fig: failure cure for h/w

→ ~~B Software is not susceptible to the~~ environmental maladies. (norm)
so doesn't wearout. The failure rate curve for s/w take the form of the 'idealized curve'.

→ The s/w is deteriorating (worse) due to change.

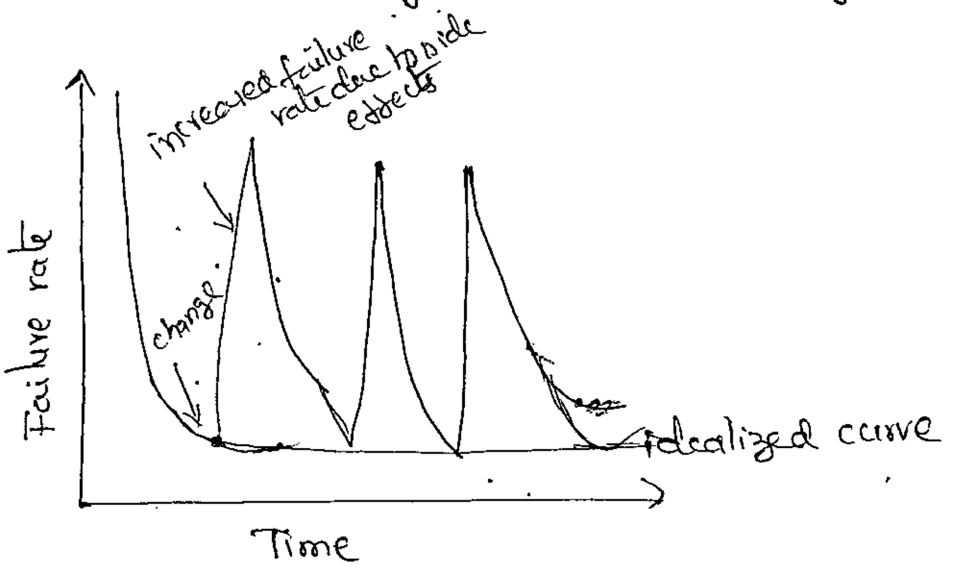


fig: failure curve for s/w

3. Although the industry is moving toward component based construction, most s/w continues to be custom built :-

→ In the h/w world, component reuse is a natural part of the engineering process. In the s/w world, it has only begun to be achieved on a broad scale.

→ ex:- user interfaces built with reusable components.

→ The changing Nature of Software :- (types of s/w)

→ Today, seven broad categories of computer software present continuing challenges for software engineers

→ 1. System software :- System software is a collection of programs written to service other programs.

→ Some system software (eg: compilers, editors, and file management utilities) processes complex, but determinate information structure

→ Software is determinate if the order and timing of inputs, processing, and outputs is predictable.

→ Other system applications (eg: operating system components, drivers, networking software, telecommunications processors) process largely indeterminate data.

→ Software is indeterminate if the order and timing of inputs, processing and outputs cannot be predicted in advance.

→ 2. Application software :- Application software consists of standalone programs that solve a specific business need.
→ application software is used to control business functions in real-time. eg:- (1) real-time manufacturing process control.
(2) Sales transaction processing.

→ 3. Engineering /scientific software :- formerly characterized by "number crunching" algorithms, engineering and scientific software applications. range from astronomy to volcanology, from automotive stress analysis to space shuttle orbital dynamics and from molecular biology to automated manufacturing
→ 'Computer-aided design', 'System simulation' and other interactive applications have begun to take on real-time and even system software characteristics.

→ 4. Embedded software :- embedded software resides within a product (or) system and is used to implement and control features and functions for ~~a~~ the end-user and for the system itself.
→ examples : 1) Keypad control for a microwave oven
2) digital functions in automobiles.

→ 5. Product line software :- Designed to provide a specific capability for use by many different customers, product-line software can focus on a limited and esoteric marketplace (eg: inventory control products) or address mass consumer markets (eg. word processing, Spreadsheets etc)

New syllubus added

Legacy Software :-

Among the seven ~~appl~~ application domains, some latest of those are state - of - the - art software - just released to individuals, industry and government.

→ But other programs are older, in some cases much older. These older programs often referred to as legacy software.

→ These legacy S/w have been the focus of continuous attention and concern since the 1960s.

→ Dayani - Fard describe "legacy s/w systems were developed decades ago and have been continually modified to meet changes in business requirements and computing platforms. The proliferation growth (n) of such systems is causing headaches for large organizations who find them costly to maintain and risky to evolve."

→ Liu extend this description by noting that "many legacy systems remain supportive to core business functions and are indispensable valuable to the business".

longi?

⓪h Hence, legacy S/w is characterized by longevity and business criticality:

→ The Quality of legacy S/w :-

→ Additional characteristic ~~of~~ of legacy s/w is — Poor quality

→ The legacy systems. Sometimes have

i) inextensible designs

ii) convoluted code involved

iii) poor (or) non existent documentation

iv) testcases & results were never archived history

v) A poorly managed change history. etc.

→ even though, these systems support "core business functions and are valuable indispensable to the business"

→ If the legacy s/w meets the needs of its users and run reliably, it isn't broken and does not need to be fixed.

→ As time passes legacy S/w systems often evolve for the following reasons.

→ (i) The s/w must be adapted to meet the change needs of new computing environments (or) technology.

(ii) The s/w must be enhanced to implement new business requirements.

(iii) The s/w must be extended to make it interoperable with more modern systems (or) databases

(iv) s/w must be rearchitected to make it viable usable within a network environment

→ The goal of modern S/w engineering is " S/w systems continually change ; new s/w systems are built from the old ones and ... all must interoperate and cooperate with each other"

Software Evolution

→ Regardless of its application domain, size or complexity, computer S/w will evolve over time.

→ change (often referred to as S/w maintenance) drives this process and occurs when errors are corrected, when the S/w is adapted to a new environment, when the customer requests new features or functions and ~~when~~ when the application re-engineered to provide benefit in a modern content.

→ Manny Lehman develop a unified theory for S/w evolution. He also define some important notable laws. those are as follows

(i) The law of continuing change :– E-type systems must be continually adapted or else they become progressively less satisfactory.

(ii) The Law of Increasing complexity :– An E-type system evolves its complexity increases unless work is done to maintain or reduce it.

(iii) The law of self Regulation :- The E-type system evolution process is self regulating with distribution of product and process measures close to normal.

(iv) The law of conservation of organizational stability :- The average effective ~~[illegible]~~ activity rate in an evolving E-type system is invariant over product lifetime.

(v) The law of conservation of familiarity :- As an E-type system evolves all associated with it, developers, sales personnel, and users, for example, must maintain mastery of its content and behavior to achieve satisfactory evolution.

(vi) The law of continuing Growth :- The functional content of E-type systems must be continually increased to maintain user satisfaction over the system's lifetime.

(vii) The law of Declining quality :- The quality of E-type systems will appear to be declining unless they are rigorously maintained and adapted to operational environment changes.

(viii) The feedback System Law :- E-type evolution processes constitute multilevel, multiloop, multiagent feedback systems and must be treated as such to achieve significant improvement over any reasonable base.

⇒ 6. web applications :- 'web Applications' span a wide array of applications .. web applications can be a set of linked files and limited graphics .

→ There are also integrated with corporate databases and business applications .

ex:- Tomcat, Apachy etc web servers.

→ 7. Artificial intelligence software :-

→ This software makes use of nonnumerical algorithms to solve complex problems that are not easy to computation (or) straight forward analysis .

→ ex:- robotics, pattern recognition , expert systems and game playing)

Software MYThs

→ Beliefs about the software is S/w Myth. And the Process to build it, can be traced to the earliest days of computing.(ie this myth is not reality it is just a Beliefs . This Belief ~~arises~~ has its roots from the earliest days of computing

→ Today most knowledgeable software engineering professionals recognize myths for what they are - misleading attitudes that have caused serious problems for managers and technical people alike .

→ Myths can be classified as below

→ 1. Management Myths
2. Customer myths
3. Practitioners myths

→ Management Myths:- Managers with software responsibility, like managers in most disciplines, are often under pressure to maintain budgets, keep schedules from slipping and improve quality.

→ Myth 1: we already have a book that's full of standards and procedures for building software. won't that provide my people with everything they need to know?

→ Reality:- The book of standards may very well exist, but ~~it may not~~ used and all S/w practitioners may not aware of its existence and ~~it~~ it may not reflect modern software engineering practice and it may not give complete information.

→ Myth 2:- If we get behind schedule, we can add more programmers and catch up. (Sometimes called the Mongolian horde concept).

→ Reality:- Adding people to a late software project delays the project completion. If new people are added, people who are working must spend time educating the new comers. People can be added but only ~~a~~ in a planned and well-coordinated manner.

-> Myths: If i decide to outsource the s/w project to a third party, I can just relax and let that firm build it.

-> Reality: If an organization does not understand how to manage and control s/w projects internally, it will invariably struggle when it outsources s/w projects.

-> Customer Myths :- following are the customer myths.

-> Myth1: A general statement of objectives is sufficient to begin writing programs - we can fill in the details later.

-> Reality :- Although a comprehensive and stable statement of requirements is not always possible. unambiguous requirements are developed only through effective and continuous communication between customer and developer.

-> Myth2 :- project requirements continually change, but change can be easily accommodated because software is flexible.

-> Reality :- It is true that software requirements change, but the impact of change varies with the time at which it is introduced. when requirements changes are requested early (before design or code has been started), cost impact is relatively small. However, as time passes, cost impact grows rapidly.

-> Practioners myths :- Myths that are still believed by s/w practitioners have been ~~fos~~ promoted by over 50 years of programming culture.

→ Myth 1: once we write the program and get it to work, our job is done.

→ Reality :- Someone once said that the sooner you begin writing code, the longer it'll take ~~to~~ you to get done. Industry data indicate that between 60 & 80% of all effort (people) expended on s/w will be expended after it is delivered to the customer for the first time.

→ Myth 2:- until I get the program running, I have no way of assessing its quality.

→ Reality :- one of the most effective s/w quality assurance mechanisms can be applied from the starting of a project — the formal technical review. s/w reviews are a "quality filter" that have been found to be more effective than testing for finding certain classes of s/w errors.

→ ~~The~~ Myth 3 :- The only deliverable work product for a successful project is the working program.

→ Reality :- Documentation provides a foundation for successful engineering and more importantly, guidance for s/w support. ie A working program is only one part of a s/w configuration that includes many elements.

→ Myth 4:- s/w engineering will make us create voluminous and unnecessary documentation and will invariably slow us down.

→ Reality:– s/w engineering is not about creating documents. It is about creating quality. Better quality leads to reduced rework. And reduced rework results in faster delivery time.

→ Recognition of s/w realities in the first step toward formulation of practical solutions for s/w engineering.

→ A generic view of process :–

→ when we want to build a product (or) system, it is important to go through a series of predictable steps. i.e a roadmap that helps us to create (or) build high quality result. These series of steps (or) roadmap which helps to produce good product is known as s/w process.

→ we can simply define a s/w process as a framework for the tasks that are required to build high quality s/w

→ s/w engineers and managers adapt the process to their needs and then follow it.

→ s/w process is important because it provides stability, control and organization to an activity and documentation that are appropriate for the project team and the product.

→ A s/w process defines the approach that is taken as software is engineered. But s/w engineering also encompasses technical methods and automated tools..

✯ SOFTWARE ENGINEERING - A LAYERED TECHNOLOGY

→ The IEEE organization has developed a Comprehensive definition of S/w engineering.

→ "Software Engineering" is the application of a system disciplined, quantifiable approach to the development, operation and maintenance of software.

→ S/w engineering is a layered technology. It mainly consists of a quality focus, process, methods and tools ~~as~~ layers as shown in below figure.

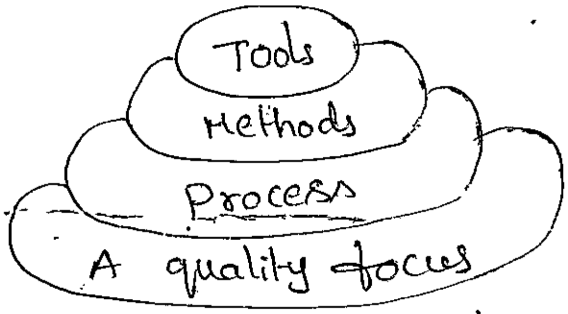


fig: S/w engineering layers

→ However, any engineering approach based on a organization commitment to quality which leads to the development of that engineering approach (S/w engineering so, the bedrock that supports software engineering is a quality focus.

→ The foundation for software engineering is the process layer. This layer is important because it holds the technology layers together.

⇒ Process defines a framework for the tasks that are required to built and effective delivery of s/w engineering technology.

→ Process forms the basis for management control of s/w projects, establishes content, work products are produced (ie models, document, data, report etc) milestones are established, quality is ensured ~~etc~~. and change is properly managed.

→ s/w engineering ~~tools~~ methods provide the technical "how to"s for building software. ie, it includes communication, requirement analysis, design modeling, program construction, testing and support.

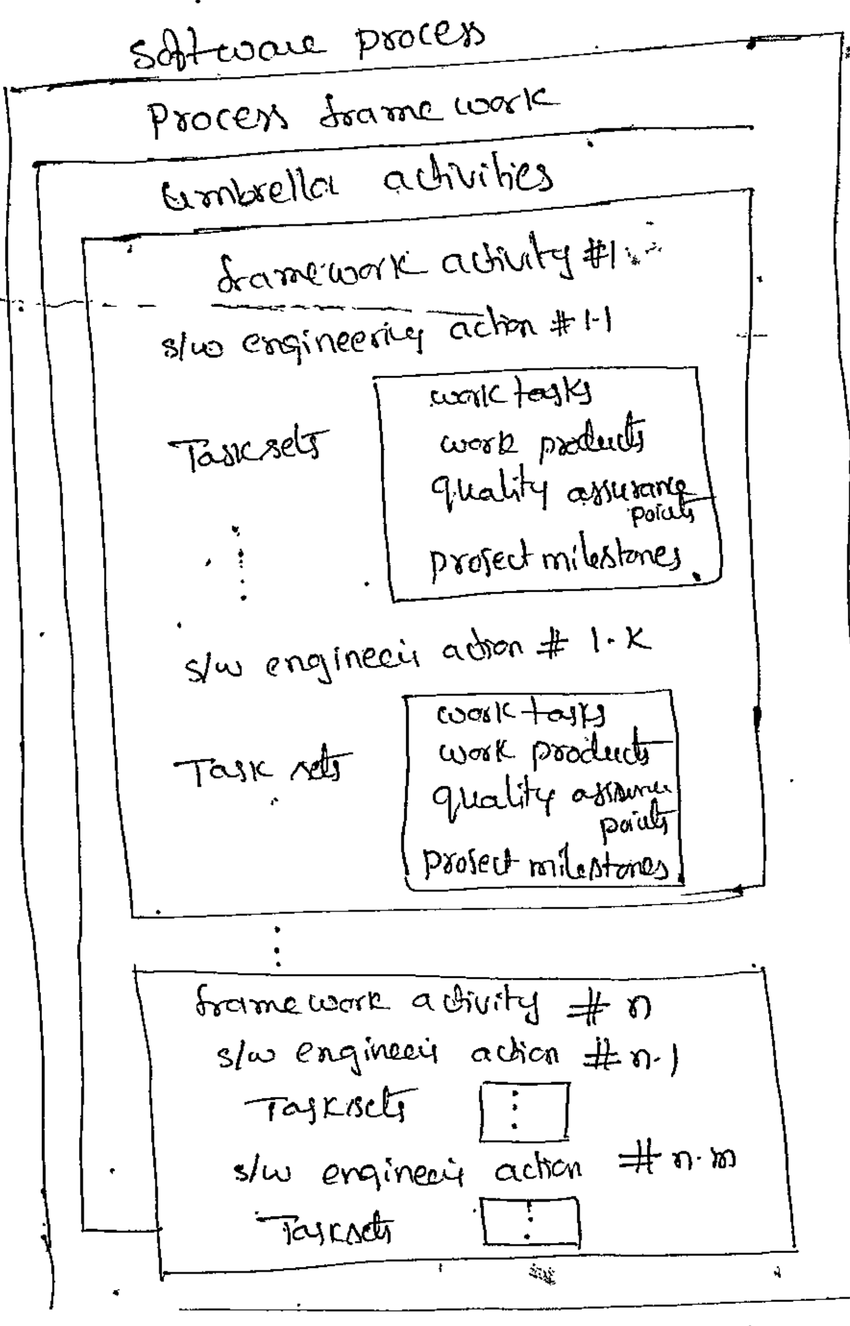
→ s/w engineering tools provide automated (or) semi automated support for the process & methods. when tools are integrated so that information created by one tool can be used by another, a system for the support of s/w development, called computer aided software engineering.

→ A process framework :-

→ A 'process framework' establishes the foundation for a complete s/w process by identifying a small no of frame work activities that are applicable to all s/w projects, regardless of their size (or) complexity.

→ process frame work includes a set of umbrella activities that are applicable across the entire s/w process.

fig: process frame work



→ Each framework activity is populated by a set of s/w engineering action.

→ Each Engineering action – ~~means~~ a collection of related tasks that produces a major S/w engineering work product.
eg:- Design is a s/w engineering action.

→ Each action is populated with individual work tasks that accomplish some part of the work implied by the action.

→ The following generic process framework is applicable to the vast majority of s/w projects.

→ 1. communication :- This framework activity involves heavy communication and collaboration with the customer and it includes ~~requirements gathering and~~ other related activities.

→ 2. Planning :- This activity establishes a plan ~~for~~ the S/w engineering ~~that to~~ work that follows. It describes the technical tasks to be conducted, the risks that are likely, the resources that will be required, the work products to be produced, and a work schedule.

→ 3. Modeling :- This activity includes ~~a plan for the S/w engineering wor~~ the creation of models that allow the developer and the customer to better understand

S/w requirement and the design that will achieve those requirements.

→ 4. Construction :- This activity combines code generation (either manual (or) automated) and the testing that is required to uncover errors in the code.

→ 5. Deployment :- The S/w is delivered to the customer who evaluates the delivered product and provides feedback on the evaluation.

→ These five generic framework activities canbe used during the development of small programs, the creation of large web applications, and for the engineering of ~~large, complex~~ computer-based systems. The details of the S/w process will be different in each case, but the framework activities remain the same.

→ EX :- The modeling activity is composed of two S/w engineering actions – analysis and design. Analysis includes a set of work tasks (e.g: requirements gathering, elaboration, specification and validation etc). ~~that~~ Design includes work tasks (data design, architectural design, interface design, and component level design).

→ Each s/w engineering action is represented by a no of different task sets. and each task set is a collection of s/w engineering work tasks, related work products, quality assurance points, and project milestones. . .

→ The task set that provides the needs of the project and the characterists of project team.

→ s/w engineering action can be adapted to the specific needs of the s/w project and the, characteristics. of the project team.

→ Along with the 5 generic activities, the framework ~~describes~~ the following umbrella activities too.

→ 1. S/w project tracking and control :- which allows the s/w team to assess progress against project plan and take necessary action to maintain schedule.

→ 2. Risk management :- Assesses risks that may effect the outcome of the project (or) the quality of the product.

→ 3. s/w quality assurance : defines and conducts the activities required to ensure s/w quality. { SQA plans, oversight, record keeping, Analysis & Reports }

→ 4. formal technical reviews :- assesses s/w engineering work products in an effort ~~and~~ to uncover and remove errors before they are spread to the next activity.

→ 5. Measurements :- defines and collects process, project and product measures.

→ 6. S/w configuration management :- manages ~~and~~ the effects of change throughout the S/w process.

→ 7. Reusability management :- defines criteria for work product reuse.

→ 8. work product preparation and production :- consists of the activities required to create work products such as models, document, logs, forms and lists.

→ All process models can be characterized within the process frame work.

The Capability Maturity Model Integration (CMMI)

→ The S/w Engineering Institute (SEI) has developed a process-meta model, CMMI guidelines.

→ It defines (in 700 pages) the process characteristics that should exist if an organization wants to establish a S/w process that is complete.

→ The CMMI represents a process meta-model in two different ways.

1. continuous model and
2. Staged model.

→ * Each s/w engineering action is represented by a no of different task sets. and each task set is a collection of s/w engineering work tasks, related work products, quality assurance points, and project milestones.

→ The task set that provides the needs of the project and the characterists of project team.

→ S/w engineering action can be adapted to the specific needs of the s/w project and the characteristics of the project team.

→ Along with the 5 generic activities, the framework describes the following umbrella activities too

→ S/w project tracking and control which allows the s/w team to assess progress against project plan and take necessary action to maintain schedule.

→ Risk management :- Assesses risks that may effect the outcome of the project (or) the quality of the product.

→ s/w quality assurance : defines and conducts the activities required to ensure s/w quality.

→ formal technical reviews :- assesses s/w engineering work products in an effort ~~and~~ to uncover and remove errors before they are spread to the next activity.

→ Measurements :- defines and collects process, project and product measures.

→ S/w configuration management :- manages ~~and~~ the effects of change throughout the S/w process.

→ Reusability management :- defines criteria for work product reuse.

→ work product preparation and production :- consists of the activities required to create work products such as models, document, logs, forms and lists.

→ All process models can be characterized within the process frame work.

The Capability Maturity Model Integration (CMMI)

→ The S/w Engineering Institute (SEI) has developed a Process-meta model, CMMI guidelines.

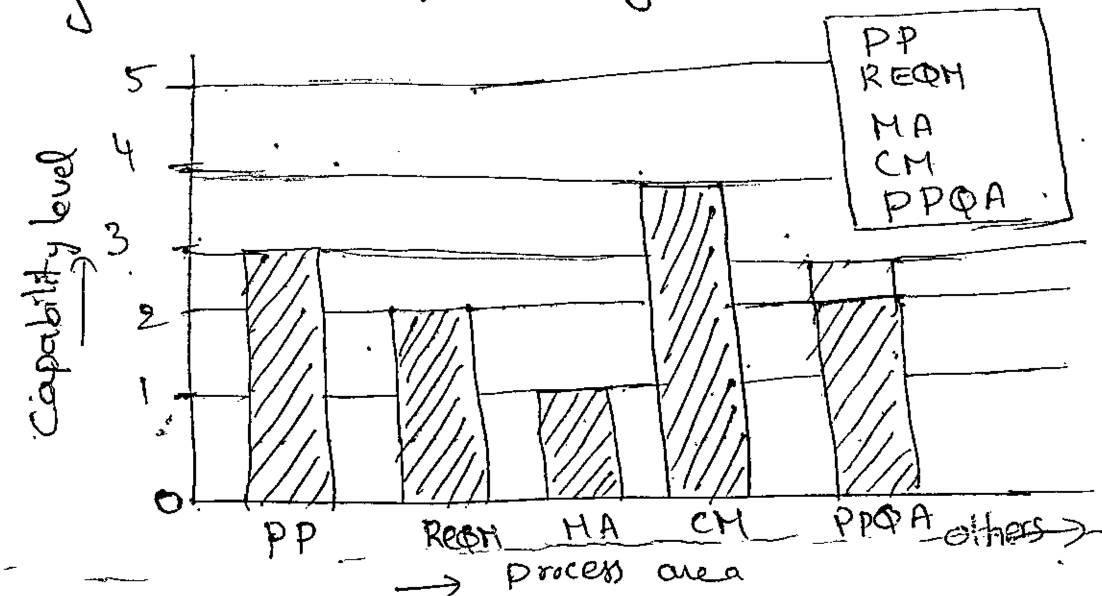
→ It defines (in 700 pages) the process characteristics that should exist if an organization wants to establish a S/w process that is complete.

→ The CMMI represents a process meta-model in two different ways.

1. continuous model and
2. Staged model.

→ The continuous CMMI meta model describes a Process in two dimensions as shown in below figure.

→ Each process area (eg: project planning) is formally assessed against specific goals and is rated according to the following capability levels.



fig① CMMI process area - capability profile.

→ In the above figure PP means project planning, REQM means Requirement Management ; MA - Measurement and Analysis , CM - configuration Management , and PPQA - process & product QA

→ Level 0 - Incomplete :- The process area (eg : req Management) is either not performed (or) does not achieve all goals and objectives defined by the CMMI for level 1 capability.

→ Level 1: Performed :- All of the Specific goals of the Process area (as defined by CMMI) have been satisfied.

→ level 2 : Managed :- All level 1 criteria have been Satisfied. In addition All work tasks and work products are "monitored, controlled, and reviewed, and evaluated for follow the process description"

→ Level 3: Defined :- All level 2 criteria have been achieved. In addition, the process is "tailored from the organization's set of standard process according to the guidelines, and contributes work products, ~~measures and~~ other improvement information to the organization assets"

→ level 4: Quantitatively managed :- All level 3 criteria have been achieved. In addition "Quantitative objects for quality & process performance are established and used as criteria in managing the process".

→ Level 5: optimized :- All level 4 criteria have been achieved. In addition, the process area is adapted and optimized using quantitative (statistical) means to meet customer needs and to continually improve the effectiveness of the process area under consideration".

→ CMMI defines each process area interms of "Specific goals" and the "specific practices" required to achieve these goals.

→ Specific practices refine a goal into a set of process related activities.

→ EX:- project planning (CMMI defined 8 process area for "project management" category). The specific goals (SG) and Specific practices (SP) are,

→ SG1 Establish estimates
- SP 1.1-1 estimate the scope of the project
- SP 1.2-1 ~~estim~~ establish estimates of work product and task attributes.
- SP 1.3-1 Define project life cycle
- SP 1.4-1 Determine estimates of effort and cost

SG2 Develop a project plan
- SP 2.1-1 establish the budget and schedule
- SP 2.2-1 Identify project risks
- SP 2.3-1 plan ~~for~~ data management
- SP 2.4-1 plan for project resources
- SP 2.5-1 plan for needed knowledge and skills
- SP 2.6-1 plan stakeholder involvement
- SP 2.7-1 establish the project plan

SG3 obtain commitment to the plan
- SP 3.1-1 Review plans that affect the project
- SP 3.2-1 Reconcile work and resource levels
- SP 3.3-1 obtain plan commitment

→ In addition to specific goals CMMI also defines, a set of 5 generic goals (GG) and generic practices (GP) corresponding to that goal, to achieve one of the capability level. (see text book PNO = 61)

→ The staged CMMI model defines the same process area, goals and practices as the continuous model. The primary difference is that the staged model defines five maturity levels, rather than five capability levels.

→ To achieve a maturity level, the specific goals and practices associated with a set of process areas must be achieved.

→ Process patterns

→ The s/w process can be defined as a collection of patterns that define a set of activities, actions, worktasks, work products, and related behaviours required to develop computer s/w.

→ In general terms, we can say a process pattern provides us with a template.

→ A template is a consistent method for describing an important characteristic of the s/w process.

→ By combining patterns, a s/w team can construct a process that best meets the needs of a project.

→ patterns can be defined at any level of abstraction. In some cases, a pattern might be used to describe a complete process. ex:- prototyping

→ In other situations, patterns can be used to describe an important framework activity ex: planning or a task within a framework activity.

→ Ambler proposed a template for describing process pattern.

(i) patteon name :- The pattern is given by meaningful name that describes ~~its~~ functions within s/w process.
ex :- Customer - communication.

(ii) Intent :- The objective of the pattern is described briefly.
the intent for the ~~above example is " to establish a~~ collaborative relationship with the customer " The intent might be further expanded with additional explanatory text and approriate diagrams if required.

(iii) Type :- 3 types of pattern is specified.
(a) Task patterns - defines s/w engineering task / action.
(b) stage patterns - defines a framework activity for the process.
(c) phase patterns - defines the sequence of framework activities occur during s/w process.

(iv) initial context :- The conditions under which the pattern applied, are described here. i.e. initiation of pattern. i.e. entry state of process.

→ (v) <u>problem</u> :- The problem to be solved by the pattern is described.

→ (vi) <u>Solution</u> :- The implementation of process pattern is described. It describes how the initial state of process is modified after applying the pattern.

→ (vii) <u>Resulting context</u> :- The conditions that will result, once the pattern has been ~~[illegible]~~ successfully implemented i.e exit state of the process.

→ (viii) <u>Related patterns</u> :- A list of all other process ~~[illegible]~~ patterns that are directly related to this pattern (current pattern) are provided as a hierarchy form.

→ Therefore, process pattern provides an effective mechanism for describing any s/w process.

→ process patterns enable an organization to develop a hierarchical process description start at high level abstraction, like phase pattern

$$\begin{tikzcd}[row sep=small] \text{phase pattern} \arrow[d] \\ \text{stage pattern} \arrow[d] \\ \text{task pattern} \end{tikzcd}$$

→

→ S/w team uses the patterns as building blocks for the process model. (They can be reusable).
ex:- example for proces pattern see in pno 66, Pressman

→ Process ASSESSMENT

evaluation
→ Assesment attempts to understand the current state of the S/w process with the intent of improving it.

→ sometimes, the existence of s/w process doesn't give any guarantee of that s/w will be delivered on time, or that it will meet customer needs. (or) quality characteristics:

→ Process patterns must be added with s/w engineering practice and the process itself should be assessed to ensure a successful s/w engineering.

→ A no of different approaches to s/w process assessment have been proposed.

Standard CMMI Assessment method for process improvement (SCAMPI) : provides a five step process assessment model that incorporates initiating, diagnosing, establishing, acting and learning. The SCAMPI method uses the SEI CMMI as the basis for assessment.

→ CMM-Based Appraisal for internal process improvement (CBAIPI):- Provides a diagnostic technique for assessing the maturity of s/w organization. It uses the SEI CMM as the basis for the assessment.

s/w process improvement & capability determination — International electro technical commission

→ SPICE (ISO/IEC 15504) :- An ISO/IEC standard defines a set of requirements for s/w process assessment. The intention is to assist organizations in development of s/w process.

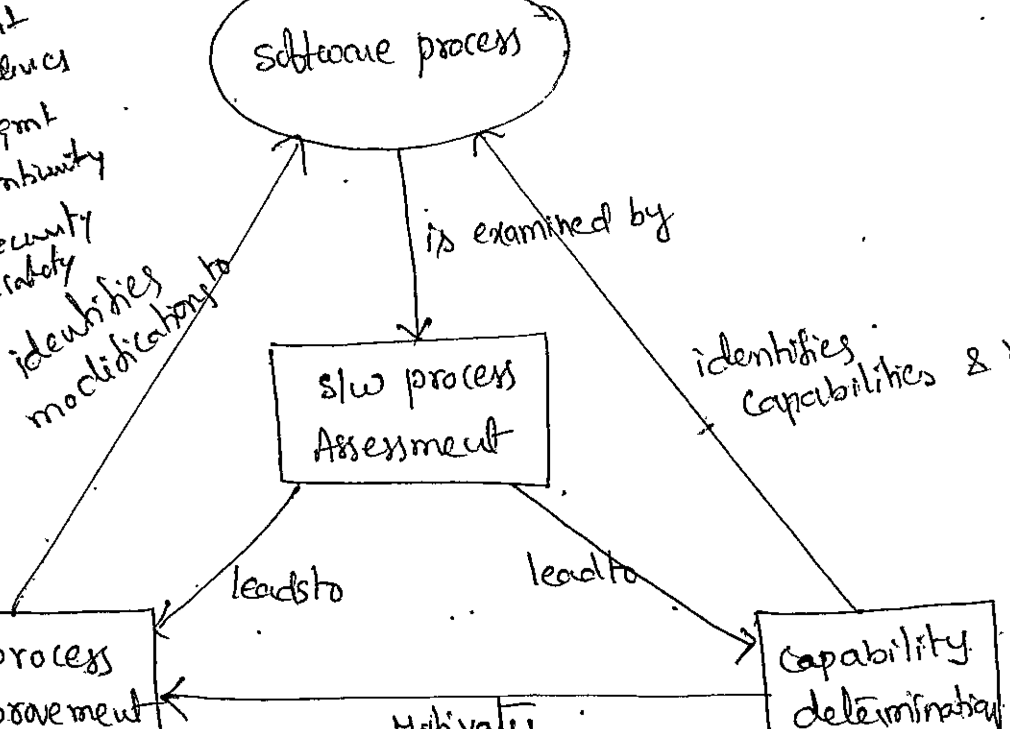
plan-do-check-act

→ ISO 9001 : 2000 for s/w :- This is a generic standard that applies to any organization that wants to improve the overall quality of the products, systems, services that it provides and improve customer satisfaction.

→ ISO 9001 : 2000 is widely used on international scale.

→ The relationship between s/w process and the methods applied for assessment is shown in below figure.

ISO 9001 - quality Mgmt
ISO 13485 - medical devices
ISO 14001 - envi mgmt
ISO 22301 - Business continuity
ISO/IEC 27001 - Inf security
ISO 45001 - health & safety
ISO 22000 - food safety



→ Personal and Team process models :-

→ The best s/w process is one that is close to the people who will be doing the work.

→ In an ideal setting, each s/w engineer would create a process that best fits his (or) her needs, and at the same time ~~meets~~ the needs of the team and organization.

→ So it is possible to create a "personal s/w process" and a "team s/w process". Both require hard work, training and coordination, but both are achievable.

→ (i) personal s/w process (or) model (psp model) :- Every developer uses some process to build computer s/w.

→ The personal s/w process emphasizes personal measurement of both the work product that is produced and the resultant quality of the work product.

→ The psp process model defines five framework activities.

(a) planning :- This activity isolates ~~gather~~ requirements and, based on these, develops both size and resource estimates. In addition, a defect estimate is made. All metrics are recorded on worksheets (or) templates. finally development tasks are identified and a project schedule is created.

→ weakness :- psp is intellectually challenging & demanding a level of commitment that is not always possible

→ training is lengthy, training costs are high,

→ (b) high-level design :- External Specifications for each component to be structured are developed and a component design is created. prototypes are built when uncertainty exists. All issues are recorded and traced.

→ (b) High-level design review :- formal verification methods are applied. to uncover errors in the design. metrics are maintained for all important tasks & work results.

→ (c) Development :- The component level design is refined and reviewed. Code is generated, reviewed, compiled, and tested. Metrics are maintained for all important tasks and work results.

→ (d) postmortem :- using the measures and metrics collected, the effectiveness of the process is determined. these should provide guidance for modifying the process to improve its effectiveness.

→ PSP emphasizes the need to record and analyze the types of errors that S/w engineer make.

→ PSP. represents a disciplined, metrics based approach, when it properly introduced, the resulting improvement in s/w engineering productivity & s/w quality are significant.

→ In the industry psp has not been widely adopted, because psp has more to do with human nature and organizational inertia. Training is lengthy and training costs are high. The required level of measurement is difficult for many s/w people. So, at personal level this model is effective s/w process.

→ (ii.) Team s/w process (TSP) :-

→ Many industry-grade s/w projects are addressed by a team practitioners.

→ The goal of TSP is to build a "self-directed" project team that organizes itself to produce high-quality software.

→ The objectives of TSP are

→ a) Build self-directed teams that plan and track their work, establish goals, and own their processes and plans. These can be pure s/w teams or integrated product teams (IPT) of 3 to about 20 engineers.

→ b) Show managers how to coach and motivate their teams and how to help them for performance.

→ c) Accelerate s/w process improvement by making CMM level 5 behavior normal and expected.

→ d) provide improvement guidance to high-maturity organizations.

→ e. facilitate university teaching of industrial grade team skills.

⇒ TSP form a self directed team which has consistent understanding of its overall goals and objectives.

→ It defines roles and responsibilities for each team member (identifies team process, assesses the risks and reacts to it, manages & reports project status etc).

→ TSP defines 6 framework activities.

~~launch, high level design, implementation, integration~~ test and postmortem.

→ launch script includes (i) Review project objectives with management and agree on and document team goals. (ii) establish team roles (iii) define the teams development process (iv) Make a quality plan and set quality targets. (v) plan for the needed support facilities (vi) produce an overall development strategy. (vii) Make a development plan for the entire project (viii) Make detailed plans for each engineer for the next phase. (ix) Merge the individual plans into a team plan and asses project risks etc..

→ TSP makes use of variety of scripts, forms and standards that serve to guide team members.

SE

UNIT - 2 Process Models

Prepared by G. Kalpana

→ Prescriptive process models were originally proposed to bring order to the ~~chaos~~ disorder of S/w development.

→ History, has indicated that these conventional models have brought a certain amount of useful structure to s/w engineering work and have provided a reasonably effective roadmap for S/w teams.

→ Prescriptive process models define a distinct ~~set of activities,~~ actions, tasks, milestones, and work products that are required to engineer high-quality s/w.

→ S/w engineers and their managers adapt a prescriptive process model to their needs and then follow it.

→ In this prescriptive process approach the order and project consistency are dominant issues.

→ Every S/w engineering organization should describe a unique set of framework activities for the S/w process it adopts [ie depending on the nature of the project or product]

→ Now, we will discuss various no of prescriptive s/w process models. We call them "prescriptive" because they prescribe (set out rules for) a set of process elements such as framework activities, s/w engineering actions, tasks, work products, quality assurance etc:

→ each process model also prescribes a workflow - ie, the manner in which the process elements are interrelated to one another.

→ All s/w process models can accommodate the generic framework activities but each applies in a different manner.

☆ ✓ The Waterfall Model

→ The waterfall model, sometimes called the classic life cycle. It is oldest paradigm for s/w Engineering.

→ There are times when the requirements of a Problem are reasonably well understood - when work flows from communication through deployment in a reasonably linear fashion.

→ waterfall model suggests a systematic, sequential approach to s/w development that begins with customer specification of requirements and progresses through planning, modeling, construction, and deployment, culminating in on-going support. ~~of~~ the completed s/w.

reach highest point

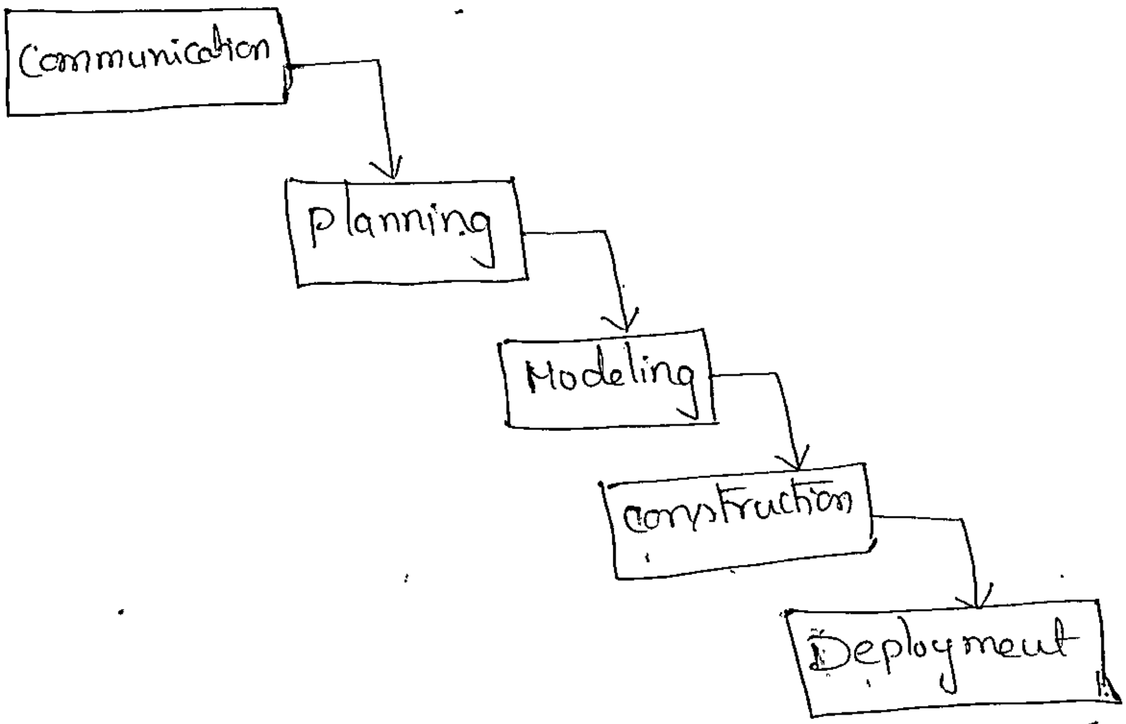


fig: waterfall model (proposed by Winston Royce)

(i) Communication :- project initiation, requirements gathering included here.

→ work begins by establishing requirements to s/w. (ie gathering) and allocate these requirements to s/w. This s/w must interact with other elements like h/w, people and databases.

(ii) planning :- estimating, scheduling, tracking included here.

→ Estimate the resources required and identify the tasks finally and schedule is created.

(iii) Modeling :- Analysis and design described here

→ understand the nature of s/w to be built

required functions, behaviour, performance etc.

→ S/w design is multistep process which mainly focuses on 4 attributes - data structure, s/w architecture, interface representations & procedural or Algorithemic details

→ requirements and design is documented.

(iv) Construction :- coding and testing described here.

→ the design must be translates into machine ~~readable form. Code should be generated.~~

→ once code has been generated program testing begins to uncover errors and ensure the result produced according to requirements (or) not.

(v) Deployment :- Delivery, support & feedback included here

→ The finished s/w product is delivered to customer

→ S/w will undergo changes after it delivered to customer. (to meet the needs of customer)

→ customer evaluates the s/w product in the form of feedback.

Criticism of waterfall model

→ The problems that are encountered when the waterfall model applied are

(i) Real projects rarely follow the sequential flow. Although this linear model can accomodate iteration indirectly. So, the changes can cause confusion as the project team proceeds.

(ii) It is difficult for the customer to state all requirements explicitly.

(iii) The customer must have - patience. A working version of programs not be available until late in project ~~time span~~

→ The linear nature of the waterfall model leads to "blocking states" in which some team member must wait for other members to complete tasks.

✓ (i) Incremental. process models < Incremental model / RAD Model

☆ → The incremental model combines the elements of waterfall model applied in an iterative fashion.

→ The incremental model applies linear sequences in a staggered (i.e uneven) fashion as calender time progresses.

→ each linear sequence produces deliverable "increments" of the s/w. In other words this model delivers a series of releases, called increments which provide more functionality for customer (at each increment).

→ ex:- word processing s/w developed using this model.

1st increment - deliver file management, editing and document production functions

2nd increment - more sophisticated editing and document production function.

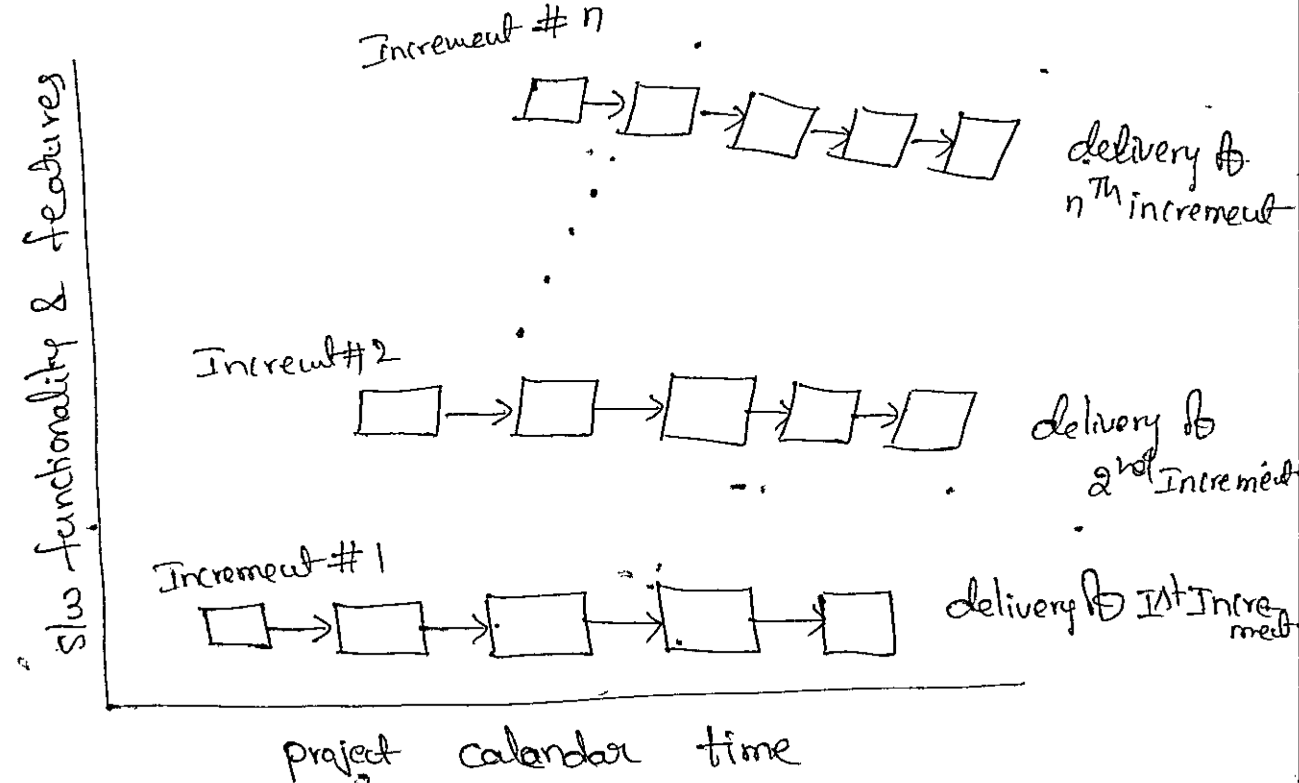
3rd increment - spelling and grammer checking etc.

~~→ When an~~ incremental model is used, the first increment is often a core product. ie basic requirements are addressed, but many features remain undelivered. This core product is used by customer.

→ As a result of use, a plan is developed for the next increment, which better meet the needs of customer and the delivery of additional features and functionality. This process is repeated until the complete product is produced.

→ Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline.

fig : The incremental model



Increment # n
delivery of nTh increment
Increment#2
delivery of 2nd Increment
Increment #1
delivery of 1st Increment
s/w functionality & features
project calendar time

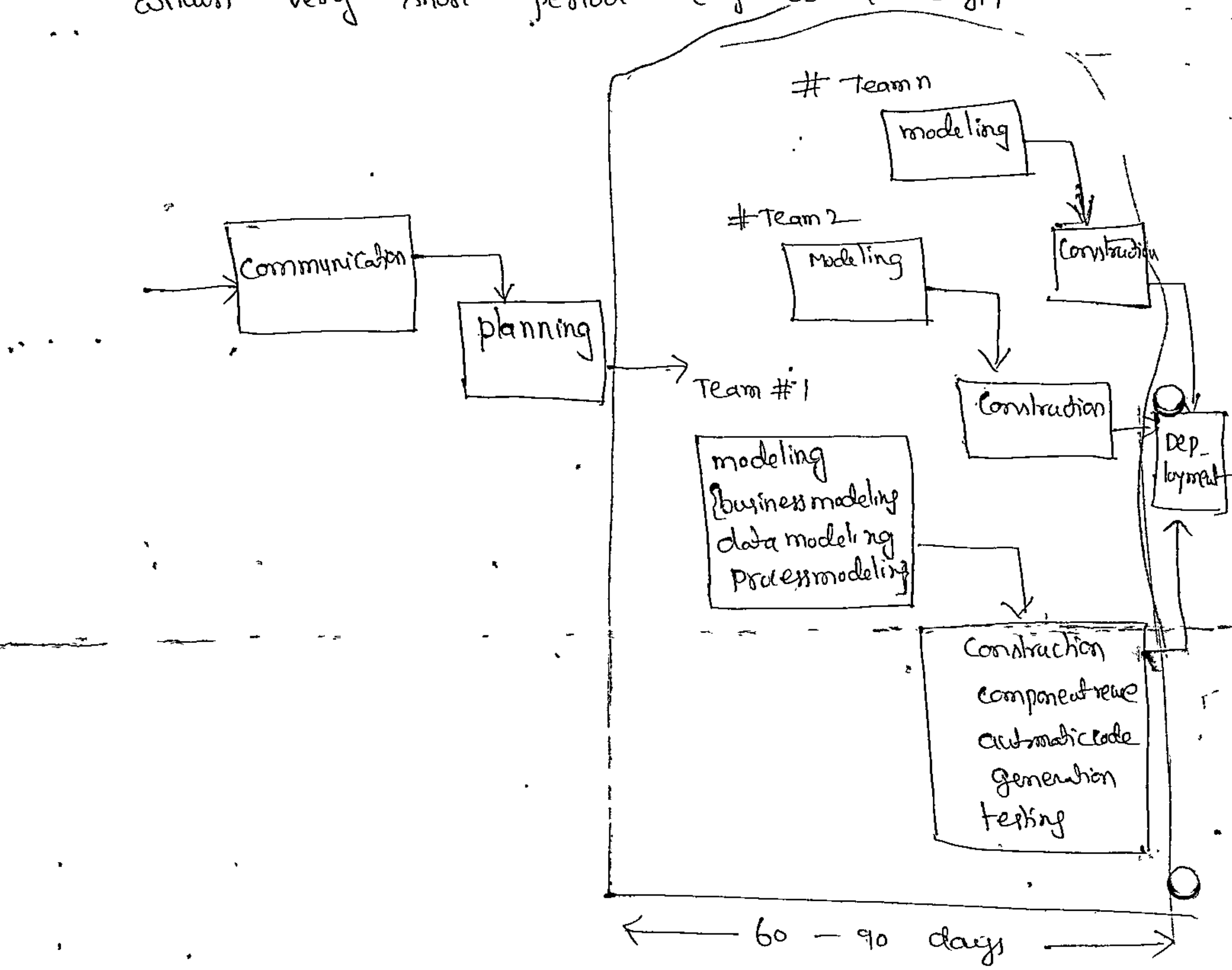
→ Increments can be planned to manage technical risks.
ex:- . A major system requires a new H/w that is under development & delivery date is uncertain.

(ii) The RAD Model

→ Rapid Application development is another incremental s/w process model that consisting of a short development cycle.

→ The RAD model is a "high-speed" adaption of waterfall model. Here the Rapid development is achieved by using a component based construction approach.

→ If the requirements are well understood, then this model enables to create " fully functional system" within very short period. (eg: 60 – 90 days)



→ The RAD process model also accomodate the generic framework activities. ie communication, planning, Modeling, Construction and deployment.

→ multiple s/w teams works in parallel on different system functions.

→ Here modeling encompasses three phases or sub activities ie business modeling, data modeling and process modeling and establish design.

→ construction includes component reuse, code generation, and testing.

→ Deployment includes integration, delivery, feedback activities.

→ each major function is addressed by a seperate RAD team and then integrated to form a whole.

Drawbacks of RAD are

(i) RAD requires ~~Sufficient~~ human resources to create the RAD teams

(ii) if developers and customers are not committed to rapid-fire activities to complete s/w then it will fail.

iii) If the system cannot be properly modularized, component building is problematic.

(iv) RAD. approach may not give high performance.

(v) RAD may not be appropriate when technical risks are high. (eg: when a new application makes heavy use of new technology)

III Evolutionary process models

fig (1) :- prototyping model

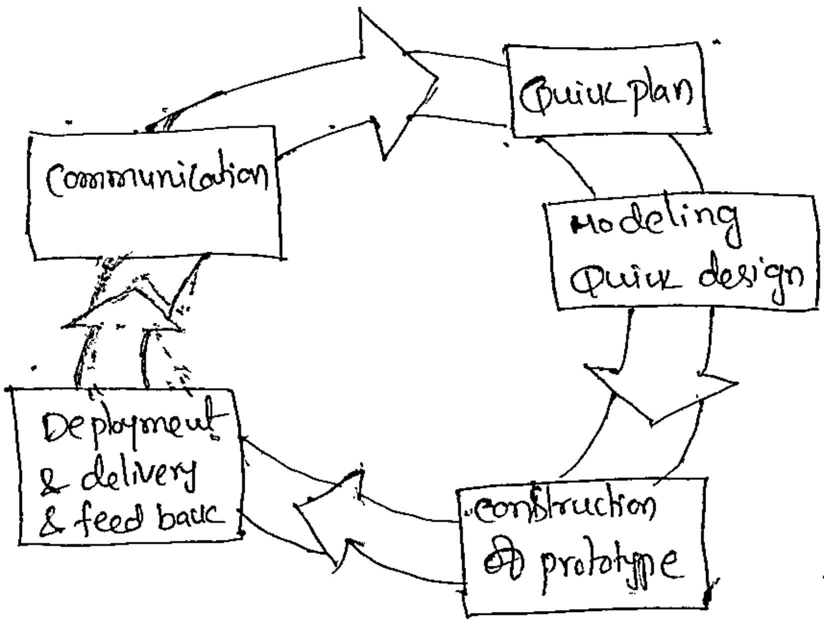
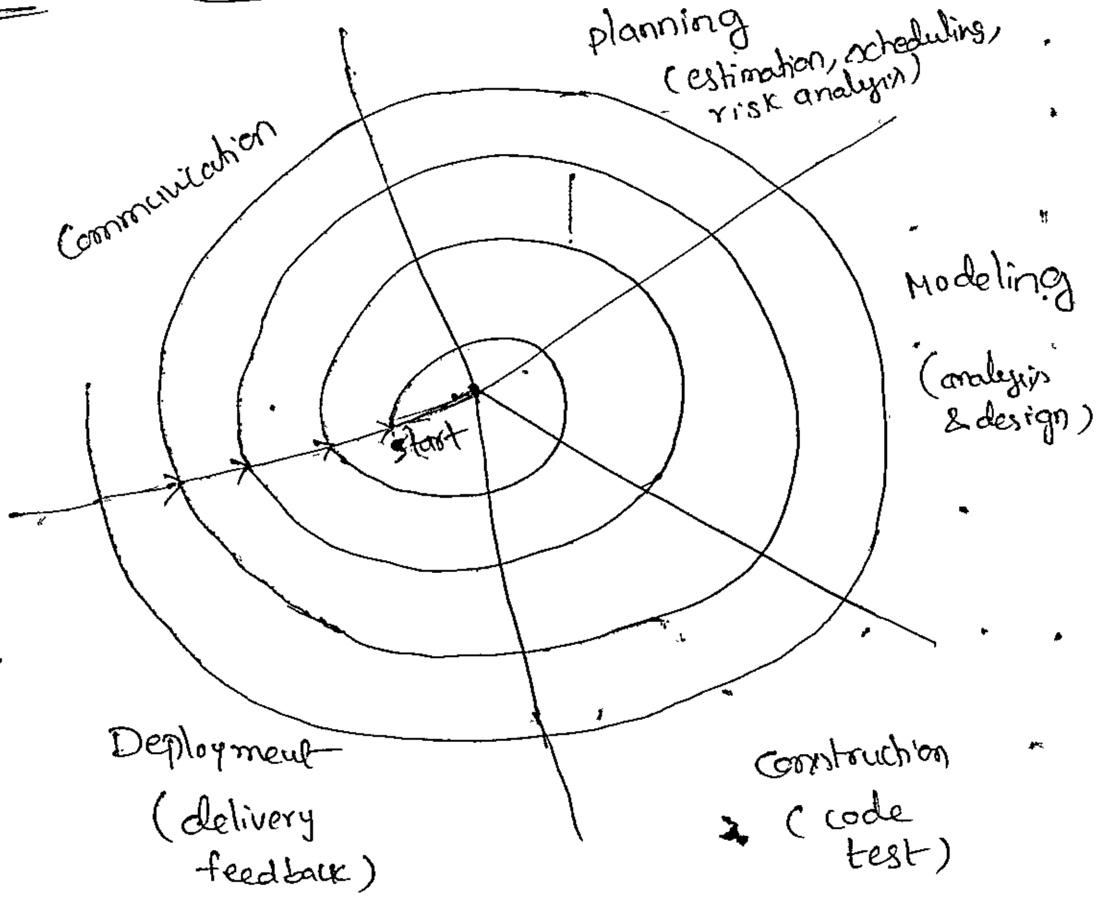


fig (2) :- A spiral model



planning (estimation, scheduling, risk analysis)
Communication
Modeling (analysis & design)
start
Deployment (delivery feedback)
Construction (code test)

III : Evolutionary Process Models

→ S/w, like all complex systems, evolves over a period of time.

→ Business and product requirements often change as development proceeds and tight market deadlines makes completion of a comprehensive S/w product impossible. In these situations, s/w engineers need a process model that has been explicitly designed to accommodate a product that evolves over time.

→ Evolutionary models are iterative. They are characterized in a manner that enables s/w engineers to develop ~~increasingly more~~ complete versions of the S/w.

→ Evolutionary process models
- (i) prototyping
- (ii) The Spiral model
- (iii) The concurrent development model.

(i) prototyping model :-

→ Some times a customer defines a set of general objectives for S/w, but he may not identify detailed input, processing, or output requirements.

→ Some times the developer may be unsure of the efficiency of an algorithm, the adaptability of an operating system etc.. In these situations prototyping model is best approach.

→ Simply we can say "the prototyping paradigm (model) assists the s/w engineer and the customer to better understand what is to be built when requirements are fuzzy".

→ As shown in the figure (1) prototyping paradigm begins with communication. s/w engineer and customer meet & define objectives and identify the requirements. and outline areas where further definition is mandatory.

→ A prototyping iteration is planned quickly and modeling occurs in the form of a "quick design".

→ The quick design focuses on the aspects of s/w that will be visible to end-user and leads to the construction of prototype.

→ This prototype is deployed and evaluated by the customer/end user.

→ feedback is used to refine requirements of s/w.

→ Iteration occurs until the prototype satisfies the customer and s/w developer.

→ According to "Brooks" the prototype can serve as "the first system", the first system built is barely usable. ie it may too slow, big, awkward in use. There is no alternative, but needs start again smarting and redesigned version by solving the problem in the first system.

Drawbacks of prototyping

→ (a) unaware that to get it working, we haven't considered overall s/w quality (or) long-term maintainability.

→ (b) The developer makes implementation compromise inorder to get prototype working quickly. ie an inefficient algorithm may be implemented to demonstrate capability.

→ Although problems can occur, this model can be effective, because customer and developer both agree to built prototype for defining requirements then it is discarded and the actual s/w is engineered.

(ii) The Spiral model :-

→ It was developed by Boehm, as an evolutionary s/w process model that combines the iterative nature of prototyping with the controlled and systematic aspects of the waterfall model.

→ It provides the potential for rapid development of increasingly more complete versions of the s/w.

→ using the Spiral model, s/w is developed in a series of evolutionary releases. During early iterations, the release might be a paper model or prototype. During later iterations, increasingly more complete versions of the engineered system are produced.

→ A spiral model is divided into a set of framework activities defined by s/w engineering team.

→ consider generic framework activities ~~as shown~~ in figure (2) each activity represents one segment of the spiral path.

→ As this evolutionary process begins, the s/w team performs activities that are implied by a circuit around the spiral in a clockwise direction, beginning at the center:

→ Risk is considered as each evolution is made.

→ Anchor point milestones - a combination of work products and conditions that are attained along the path of the spiral are noted for each evolutionary pass.

→ unlike other process models that end when s/w is delivered, the spiral model can be adapted to apply throughout the entire life cycle of an application, from concept to maintenance of s/w.

→ The circuit (1) around the Spiral represents a concept development project: which starts at the core of spiral and perform all activities (communication, planning, modeling, construction & deployment) until concept development complete.

→ If the concept is developed into actual product further proceeds the process as circuit (2) ie "new product development project" begins.

→ This new product proceed through no of iterations around the spiral and further project may required as mentioned with circuit (3) ie "product enhancement project" and further it proceeds represents as circuit (4) "product maintenance project"

→ when ever a change is ~~is~~ initiated the process starts at the appropriate entry point (eg: product enhancement)

Advantages

→ The spiral model is a realistic approach to the development of large-scale system & s/w because s/w evolves as the process proceeds (or) progresses. So, the customer & developer better understand & react the risks at each level.

→ The developer enables to apply the prototyping approach at any stage in evaluation of product not only as risk reduction mechanism.

→ It maintains systematic approach of waterfall model and also includes iterative framework activities, which reflects the realworld.

Drawbacks

→ The spiral model is not familiar like other models because, it may be difficult to convince customers that the evolutionary model is controllable.

→ If a major risk is not uncovered and managed problems will occur.

→ If the management demands fixed-budget development then this model can be problematic, because as each circuit is completed project cost is revised.

(iii) The concurrent Development Model

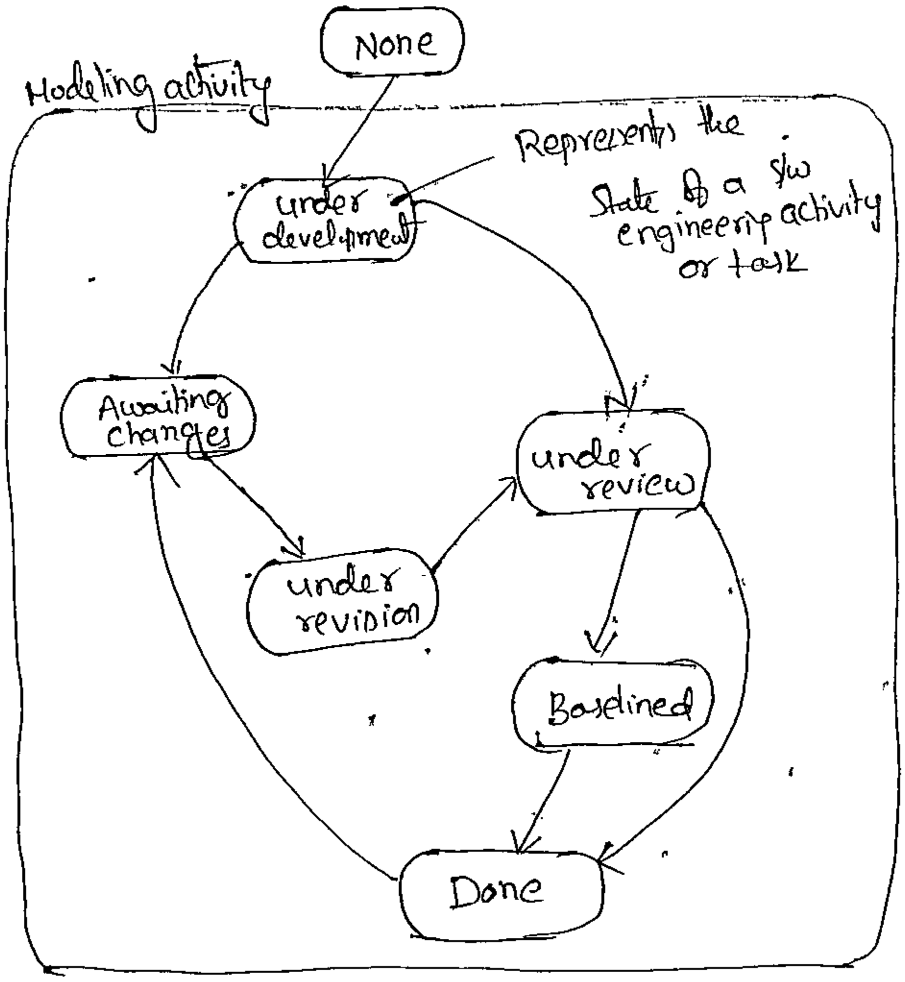
→ The concurrent development model, sometimes called concurrent engineering ;

→ It can be represented schematically as a series of framework activities, s/w engineering actions and tasks, and their associated states.

→ for example :- the modeling activity defined for the spiral model is accomplished by invoking the actions - prototyping, or analysis modeling and specification and design.

→ The following figure represents the schematic representation of ~~an~~ a s/w engineering task within the modeling activity for the concurrent process model.

→



→ Here the modeling activity may be in any one of the state noted at any given time.

→ Similarly other activities can be represented analogous manner.

→ As shown in above figure the modeling activity existed in none state initially (while communication was completed) now makes a transition into the under development state. Then according to customer needs it moves awaiting changes, then reviewed finally done state.

→ All activities exist concurrently but reside in different states.

~~Advantages~~

✓ → The concurrent model is often more appropriate for system engineering projects where different engineering teams are involved.

Advantages

→ Inconsistency in the analysis model is uncovered; because concurrent process model defines a series of events that will trigger transitions from state to state. (observable behaviour is represented)

→ This model is applicable to all types of s/w development and provides ~~to all types of s/w~~ an accurate picture of current state of project.

→ It defines a network of activities / tasks / actions (instead of series). So, the events triggered at one point in the process network trigger transitions among the states.

Ex :-
→ The concurrent process model is often used as the paradigm for the development of client/server applications

→ A client/server system is composed ~~of a set of~~ functional components.

→ When applied to client/server, the concurrent process model defines activities in two dimensions ie a system and a component dimension.

→ system level issues are addressed using three activities - design, assembly and use.

→ The component level ~~co~~ dimension is addressed with two activities - design and realization.

→ concurrency achieved in two ways (i) system and component activities occur simultaneously and can be modeling using the state oriented approach described above.

(ii) A typical client/server application is implemented with many components, each of which can be designed and realized concurrently.

The unified process

→ The unified process is a "use-case driven, architecture-centric, iterative and incremental" s/w process designed as a framework for UML methods & tools.

→ This process has best features and characteristic of conventional process models.

→ The unified process recognizes the importance of customer communication and ~~streamlined methods~~ for describing the customer's view of a system ie (use case -" is a text ~~ma~~ narrative or template that describes a system function (or) feature from the user's point of view).

→ It also emphasizes the important role of S/w architecture and "helps the architect focus on the right goals, such as understandability, reliance trust to future change, and reuse".

→ It also suggests a process flow that is iterative and incremental, ~~proving~~ providing the evolutionary feel that is essential in modern s/w development.

History of unified process

→ During 1980's and 1990's, object oriented (OO) methods and programming languages are widespread through the s/w engineering community.

→ so, a variety of OOA (object oriented Analysis) and OOD (object oriented Design) methods were proposed in that period. and the object oriented process model was introduced.

→ In 1990 James Rumbaugh & Grady Booch began working on a "unified method" which combines best features of their individual methods and adopt additional features by experts in ~~OO field.~~

→ So, they proposed UML - a unified modeling language that combines a robust notation for the modeling and development of OO systems.

→ UML provides the necessary technology to support s/w engineering approach, but it does not provide the process framework to guide s/w project teams.

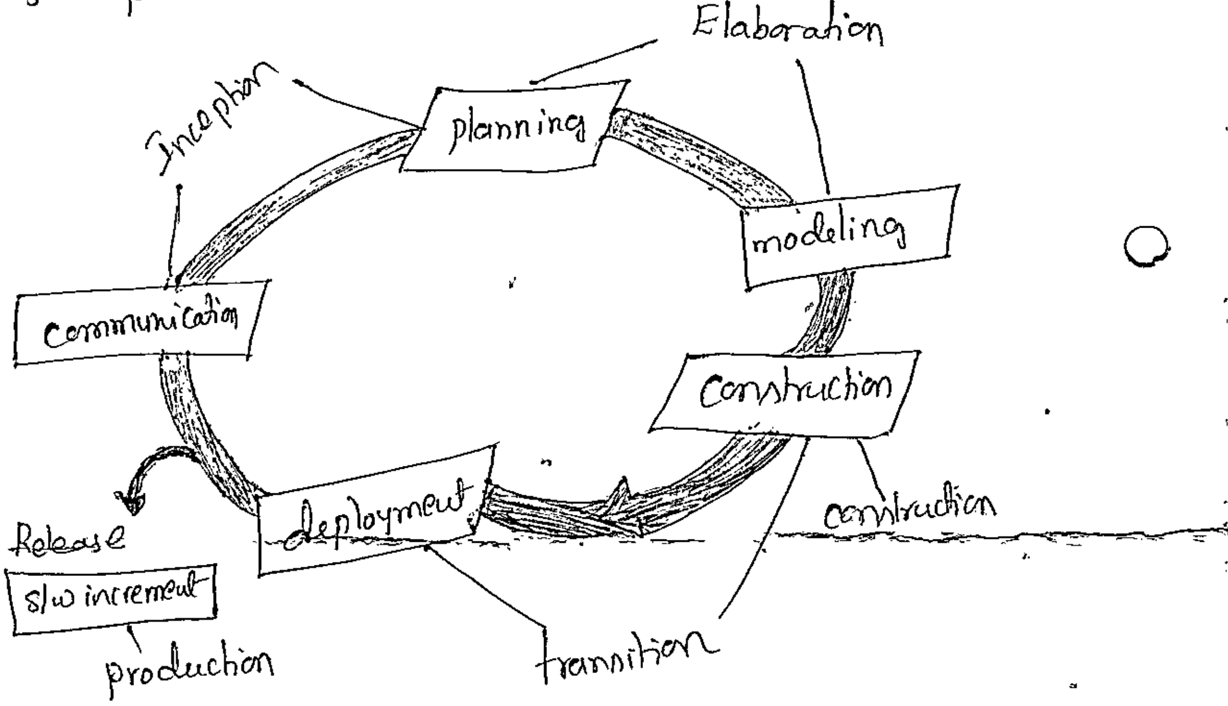
→ over the next few years Jacobson, Rumbagh, and Booch developed the unified process, a framework for object oriented s/w engineering using UML.

→ Today, the unified process and UML are widely used on OO projects of all types.

phases of unified process

→ unified process is an incremental model in which 5 phases are defined.

→



→ figure :- The unified process

(i) Inception :- This phase includes both customer communication and planning activities.

→ By collaborating with the customer (end-user) fundamental business requirements are described through a set of preliminary use-cases.

→ A rough architecture for the system is proposed and a plan for the iterative, incremental nature of the project is developed.

→ A usecase describes a sequence of actions that are performed by an actor (may be a person, machine , another system) as the actor interacts with the s/w .

→ use-cases help to identify the scope of the project and provide a basis for project planning.

(ii) Elaboration :- This phase encompasses the customer communication and modeling activities. This phase refines and expands the preliminary use-cases and expands the architectural representation which includes 5 views of s/w - the usecase model, analysis, design & tere implementation and deployment model. (ie focusing on creation of analysis and design models).

→ The plan is carefully reviewed to ensure scope, risks and delivery dates reasonable. Modifications may be made.

(iii) Construction :- This phase develop code (or) acquires the s/w components that will make each use-case operational for end users.

→ All necessary features and functions of s/w increment are then implemented in source code.

→ As components are being implemented, unit tests are designed and executed for each.

→ Integration activities are conducted here.

(iv) Transition :- The s/w transfered from the developer to the end-user ~~for~~ beta-testing and acceptance. In (Beta testing is a controlled testing action in which the s/w is used by end-user with the intent of uncover defects) the form of feedback.

→ At the ending of this phase the s/w increment becomes a usable s/w release.

(v) Production :- This phase ~~coind~~ coincides with the deployment activity. In this phase on going monitoring, support are provided and defects reported and requests for changes are submitted.

→ A s/w ~~engineering~~ workflow is distributed across all unified process phases. The workflow identifies the tasks required to accomplish the s/w engineering action and work products.

unified process work products

Inception phase :- vision document, initial usecase model, initial project glossary, initial business case, initial risk assessment, project plan, ~~proje~~ Business model if necessary one or more prototypes.

Elaboration phase :- usecase model, supplementary requirements including non functional, Analysis model, s/w architecture description, executable architectural prototype, preliminary design model, revised risk list, project plan including

Agile process Models :- there are 4 sub models

(1) Extreme programming (XP)

(2) Adaptive S/w Development (ASD)

(3) Dynamic S/m s Development Method (DSDM)

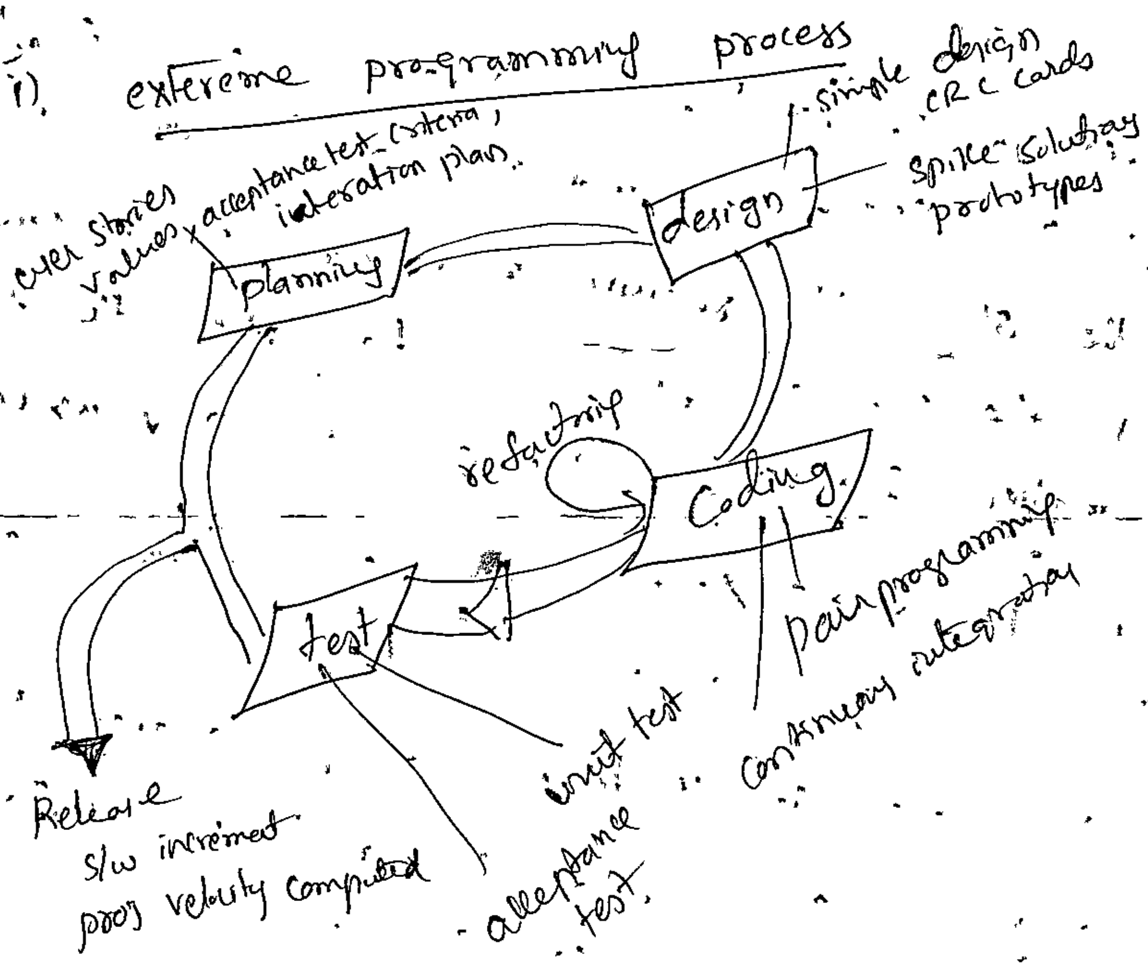
(4) Scrum

(5) Crystal

(6) Feature Driven Development (FDD)

(7) Agile Modeling (AM)

(i) exterme programming process



user stories values, acceptance test criteria, iteration plan
planning
simple design CRC cards
design
spike solutions prototypes
refactoring
Coding
pair programming
continuous integration
test
unit test
acceptance test
Release s/w increment proj velocity computed

→ work started in the year 1980 but it is was finalized (or) published by Kent Beck in 1999.

→ XP uses an object oriented approach ~~as its~~ preferred development model/paradigm.

→ XP encompasses a set of rules and practices that occur within the context of 4 framework activities ie planning, design, coding, testing planning :- the planning activity begins with the creation of a set of stories that describe required features and functionality for s/w to built.

→ each story (||r to use-cases) is written by the customer and is placed on an index card.

→ the customer assigns a value (i.e a priority) to the Story based on the overall business value of the feature or function.

→ Members of the Xp team then assess each story and assign a cost - measured in development weeks - to it.

→ if the story will require more than 3 development weeks the customer is asked to split the story in to smaller stories; and the assignment of value, cost again.

→ It is important to note that new Stories can be written at any time.

→ customer and the Xp team work together to decide how to group Stories in to the next release (next increment) to be developed by the Xp team.

→ once a basic commitment is made for a release, the Xp team orders the stories that will be developed in one of 3 ways.

(1) All stories will be implemented immediately (few weeks)

(2) Stories with heighest value will be implemented first.

(3) the riskiest stories will be moved up in the schedule and implemented first.

→ After the first. project, release has been delivered, the xp team computes project velocity.

→ ie no of customer. Stories implemented during Ist release.

→ this project velocity can be used ~~to~~ to

(i) help estimate delivery dates and schedule for subsequent releases. and

(ii) determine whether an over - commitment has been made ^it occurs, the content of releases is modified or end - delivery dates, are changed..

→ As development work proceeds, the customer can add stories, change the value of an existing story, split stories, or eliminate them.

→ the Xp team then reconsiders all remaining releases and modifies its plans accordingly.

Design :- Xp design follows KIS (Keep it simple) principle.

→ A simple design is always preferred over a more complex representation.

→ the design provides implementation guidance for a story as it is written - nothing less, nothing more.

→ Xp encourages the use of CRC cards as an effective mechanism for. ~~object oriented~~ about the s/w in an object oriented content.

→ CRC (class, Responsiblity, collaborator) cards identify and organize the object oriented classes that are relevant to the current s/w increment.

→ the CRC cards are the only design work products produced as part of the xp process.

→ If a difficult design problem is encountered as a part of design of a story. XP recommends the immediate creation of an operational prototype of that portion of the design. called a

→ Spike solution the design prototype is implemented and evaluated.

→ XP encourages refactoring - a construction technique that is also a design technique.

→ Flowei describes the Refactoring is the process of changing a s/w system in such a way that it does not alter the external behavior of the code yet improves the internal structure.

→ It is a disciplined way to clean up code that minimizes the chances of introducing bugs.

→ In essence, when you refactor you are improving the design of the code after it has been written.

→ A central notion in XP is that design occurs both before and after coding commences.

→ Refactoring means the design occurs continuously as the s/w is constructed.

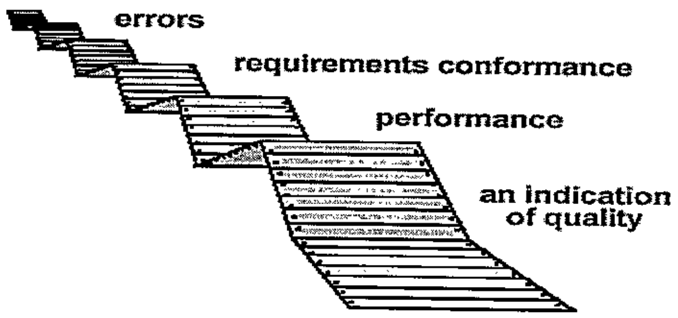
→ the construction activity itself will provide the XP team with guidance on how to improve the design.

Unit 4.TESTING STRATEGIES

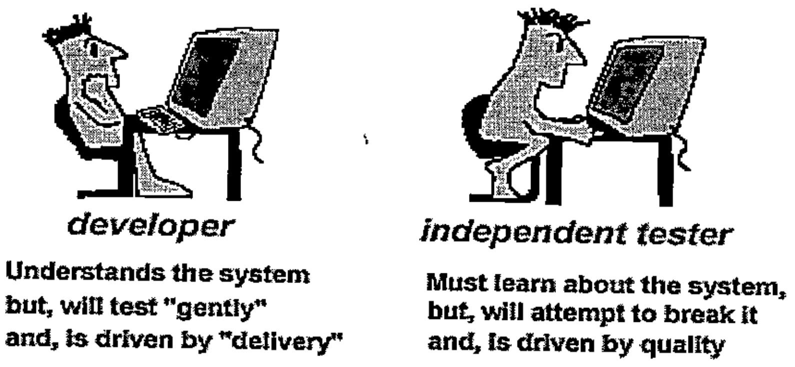
Software Testing

- Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.

What Testing Shows



Who Tests the Software?


developer
Understands the system
but, will test "gently"
and, is driven by "delivery"
independent tester
Must learn about the system,
but, will attempt to break it
and, is driven by quality

A STRATEGIC APPROACH TO SOFTWARE TESTING

- A number of software testing strategies have proposed in the literature. All provide the software developer with a template for testing and all have the following generic characteristics:

To perform effective testing, a software team should conduct effective formal technical reviews. By doing this, many errors will be eliminated before testing commences.

Testing begins at the component level and works "outward" toward the integration of the entire computer-based system.
Different testing techniques are appropriate at different points in time.
Testing is conducted by the developer of the software and (for large projects) an independent test group.
Testing and debugging are different activities, but debugging must be accommodated in any testing strategy.

Verification and Validation

- Verification refers to the set of activities that ensure that software correctly implements a specific function. Validation refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

    - Verification: Are we building the product right?
    - Validation: Are we building the right product?

- Verification and validation encompasses a ;wide array of SQA activities that include formal technical reviews, quality and configuration audits, performance monitoring, simulation, feasibility study, documentation review, database review, algorithm analysis, development testing, usability testing, qualification testing, and installation testing.

Organizing for software testing

The people who have built the software are now asked to test the software.
Unfortunately, developers have a vested interest in demonstrating that the program is error free, that it works according to customer requirements, and that it will be completed on schedule and within budget.
There are often a number of misconceptions

that the developer of software should do no testing at all,

that the software should be "tossed over the wall" to strangers who will test it mercilessly,

that testers get involved with the project only when the testing steps are about to begin.

In many cases, the developer also conducts integration testing-a testing step that leads to the construction of the complete software architecture. Only after the software architecture is complete does an independent test group become involved.
The role of an independent test group (ITG) is to remove the inherent problems associated with letting the builder test the thing that has been built.
The developer and the ITG work closely throughout a software project to ensure that thorough tests will be conducted. While testing is conducted, the developer must be available to correct errors that are uncovered.

A software Testing Strategy for conventional software

- The module (component) is our initial focus
- Integration of modules follows

A software Testing Strategy for Object Oriented software

- our focus when "testing in the small" changes from an individual module (the conventional view) to an OO class that encompasses attributes and operations and implies communication and collaboration

Criteria for completion of testing

Question arises every time software testing is discussed: when are we done testing-how do we know that we've tested enough?

One response to the question is: You're never done testing; the burden simply shifts from you to your customer. Every time the customer/user executes a computer program, the program is being tested.

You're done testing when you run out of time or you run out of money.

By collecting metrics during software testing and making use of existing software reliability models, it is possible to develop meaningful guidelines for answering the question: when are we done testing?

STRATEGIC ISSUES

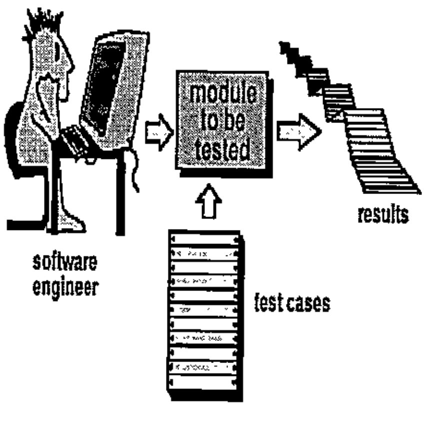
State testing objectives explicitly.
Understand the users of the software and develop a profile for each user category.
Develop a testing plan that emphasizes "rapid cycle testing."
Build "robust" software that is designed to test itself
Use effective formal technical reviews as a filter prior to testing
Conduct formal technical reviews to assess the test strategy and test cases themselves.
Develop a continuous improvement approach for the testing process.

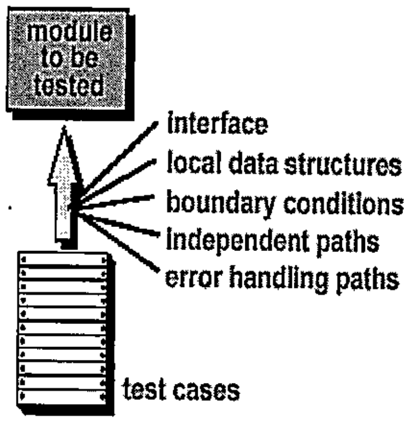
TEST STRATEGIES FOR CONVENTIONAL SOFTWARE

- There are many strategies that can be used to test software:
  - A software team could wait until the system is fully constructed and then conduct tests on the overall system in hopes of finding errors. This approach, although appealing, simply does not work. It will result in buggy software that disappoints the customer and end-user.
  - A software engineer could conduct tests on a daily basis, whenever any part of the system is constructed. This approach, although less appealing to many, can be very effective.

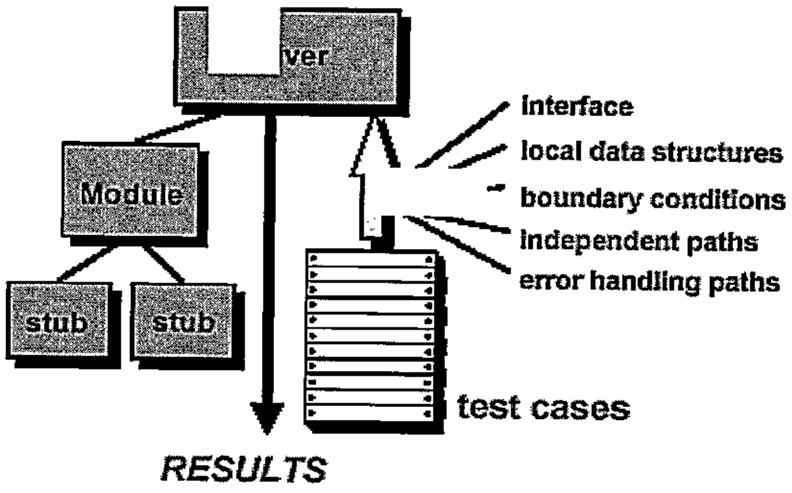
Unit Testing

- Unit testing focuses verification effort on the smallest unit of software design- the software component or module.



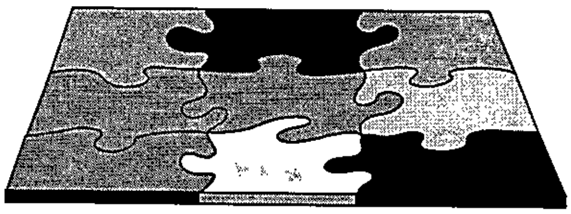


Unit Test Environment



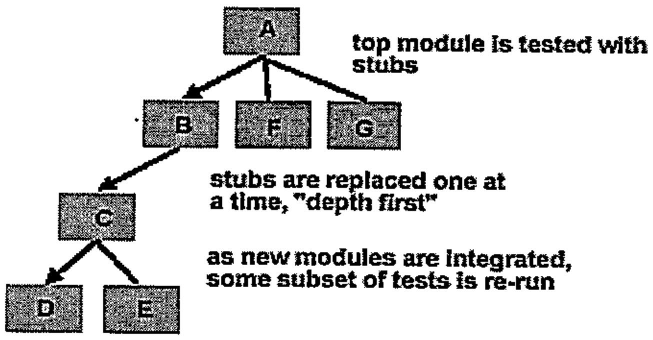
Integration Testing

- Integration testing is a systematic technique for constructing the software architecture while at the same time conducting tests to uncover errors associated with interfacing. **Options:**
    - the "big bang" approach
    - an incremental construction strategy



- **Top Down Integration**
    - Top-down integration testing is an incremental approach to construction of the software architecture.

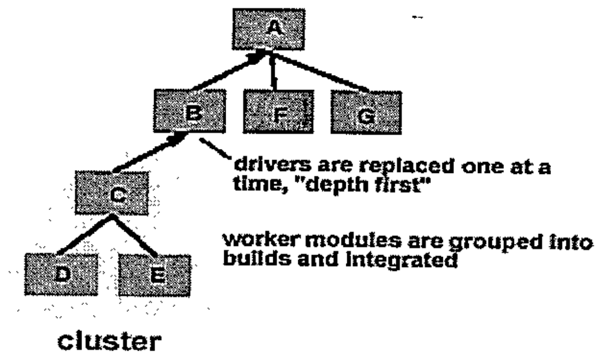
- Modules are integrated by moving downward through the control hierarchy, beginning with the main control module.


A
B
F
G
C
D
E
top module is tested with stubs
stubs are replaced one at a time, "depth first"
as new modules are integrated, some subset of tests is re-run

- The integration process is performed in a series of <u>five</u> steps.
    - The main control module is used as a test driver, and stubs are substituted for all components directly subordinate to the main control module.
    - Depending on the integration approach selected (i.e., depth or breadth first), subordinate stubs are replaced one at a time with actual components.
    - Tests are conducted as each component is integrated.
    - On completion of each set of tests, another stub is replaced with the real component.
    - Regressing testing may be conducted to ensure that new errors have not been introduced.

Bottom-Up Integration

- Bottom-up integration testing as its name implies, begins construction and testing with atomic modules.
- A bottom-up integration strategy may be implemented with the following <u>four</u> steps:
    - Low-level components are combined into clusters that perform a specific software subfunction.
    - A driver is written to coordinate test case input and output.
    - The cluster is tested.
    - Drivers are removed and clusters are combined moving upward in the program structure.


A
B
F
G
C
D
E
drivers are replaced one at a time, "depth first"
worker modules are grouped into builds and integrated
cluster

■ Regression testing

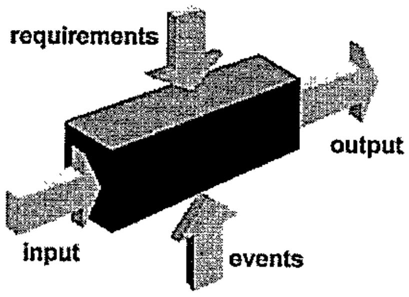
- Regression testing may be conducted manually, by re-executing a subset of all test cases or using automated capture/playback tools.
- The regression test suite contains three different classes of test cases.
    - A representative sample of tests that will exercise all software functions.
    - Additional tests that focus on software functions that are likely to be affected by the change.
    - Tests that focus on the software components that have been changed.

■ Smoke Testing

- A common approach for creating "daily builds" for product software.
- Smoke testing steps:
    - Software components that have been translated into code are integrated into a "build."
        - A build includes all data files, libraries, reusable modules, and engineered components that are required to implement one or more product functions.
    - A series of tests is designed to expose errors that will keep the build from properly performing its function.
        - The intent should be to uncover "show stopper" errors that have the highest likelihood of throwing the software project behind schedule.
    - The build is integrated with other builds and the entire product (in its current form) is smoke tested daily.
        - The integration approach may be top down or bottom up.

BLACK-BOX TESTING

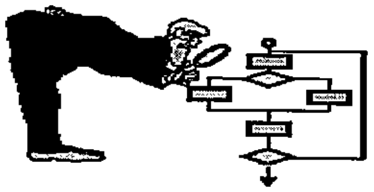
- Black-Box Testing alludes to tests that are conducted at the software interface. A black-box test examines some fundamental aspect of a system with little regard for the internal logical structure of the software.



How is functional validity tested?
How is system behavior and performance tested?
What classes of input will make good test cases?
Is the system particularly sensitive to certain input values?
How are the boundaries of a data class isolated?
What data rates and data volume can the system tolerate?
What effect will specific combinations of data have on system operation?

WHITE-BOX TESTING OR GLASS-BOX TESTING

- White-Box Testing of software is predicated on close examination of procedural detail. Logical paths through the software and collaborations between components are tested by providing test cases that exercise specific sets of conditions and/or loops.



... our goal is to ensure that all statements and conditions have been executed at least once ...

Why Cover?

Logic errors and incorrect assumptions are inversely proportional to a path's execution probability

We often believe that a path is not likely to be executed; in fact, reality is often counter intuitive.

Typographical errors are random; it's likely that untested paths will contain some.

VALIDATION TESTING

- Focus is on software requirements

■Validation Test Criteria
■Configuration review
■Alpha/Beta testing

- Focus is on customer usage

SYSTEM TESTING

- Focus is on system integration

Recovery testing

- forces the software to fail in a variety of ways and verifies that recovery is properly performed

Security testing

- verifies that protection mechanisms built into a system will, in fact, protect it from improper penetration

Stress testing

- executes a system in a manner that demands resources in abnormal quantity, frequency, or volume

Performance Testing

- test the run-time performance of software within the context of an integrated system.

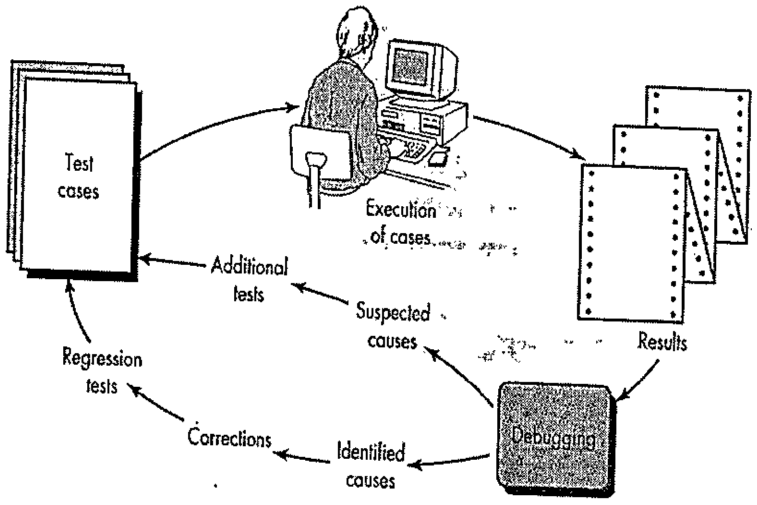
THE ART OF DEBUGGING

- Debugging occurs as a consequence of successful testing. That is, when a test case uncovers an error, debugging is an action that results in the removal of the error.

The Debugging Process:

- Debugging will always have one of <u>two</u> outcomes:
  - The cause will be found and corrected
  - The cause will not be found. In the latter case, the person performing debugging may suspect a cause, design one or more test cases to help validate that suspicion, and work toward error correction in an iterative fashion.

witl
pro'



to do
bugs

s, the
may
upled

The symptom may disappear when another error is corrected.
The symptom may actually be caused by non errors.
The symptom may be caused by human error that is not easily traced.
The symptom may be a result of timing problems, rather than procession problems.
It may be difficult to accurately reproduce input conditions.
The symptom may be intermittent. This is particularly common in embedded systems that couple hardware and software inextricably.
The symptom may be due to causes that are distributed across a number of tasks running on different processors.

- During debugging, we encounter errors that range from mildly annoying to catastrophic.

Psychological Considerations

- Debugging is one of the more frustrating parts of programming. It has elements of problem solving or brain teasers, coupled with the annoying recognition that you have made a mistake. Heightened anxiety and the unwillingness to accept the possibility of error increases the task difficulty. Fortunately, there is a great sigh of relief and a lessening of tension when the bug is ultimately corrected.

Debugging Strategies

- Regardless of the approach that is taken, debugging has one overriding objective: to find and correct the cause of a software error. The objective is realized by a combination of systematic evaluation, intuition and luck.
- In general, three debugging strategies have been proposed
  - Brute force
  - Backtracking
  - Cause elimination.
- Debugging tactics:
  - The brute force category of debugging is probably the most common and least efficient method of isolating the cause of a software error.
  - Backtracking is a fairly common debugging approach that can be used successfully in small programs.
  - Cause elimination is manifested by induction or deduction and introduces the concept of binary partitioning.

Automated debugging

- Each of these debugging approaches can be supplemented with debugging tools that provide semi-automated support for the software engineer as debugging strategies are attempted.

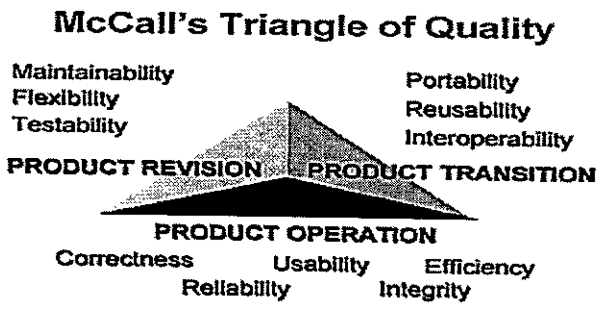
Unit 5. PRODUCT METRICS

SOFTWARE QUALITY

- Software quality is conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.
- The definition serves to emphasize three important points:
    - Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.
    - Specified standards define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.
    - There is asset of implicit requirements that often goes unmentioned. If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.
- Software quality is a complex mix of factors that will vary across different applications and the customers who request them.

■ McCall's Quality Factors

- The factors that affect software quality can be categorized in two broad groups:
    - Factors that can be directly measured.
    - Factors that can be measuring only indirectly. In each case measurement should occur. We must compare the software to some datum and arrive at an indication of quality.
- McCall, Richards and Walters propose a useful categorization of factors that affect software quality. These software quality factors, shown in figure, focus on three important aspects of a software product: Its operational characteristics, its ability to undergo change, and its adaptability to new environments.
- Referring to the factors noted in figure, McCall and his colleagues provide the following descriptions:



Correctness: The extent to which a program satisfies its specification and fulfills the customer's mission objectives.

Reliability: The extent to which a program can be expected to perform its intended function with required precision.

Efficiency: The amount of computing resources and code required by a program to perform its function.

Integrity: The extent to which access to software or data by unauthorized persons can be controlled.

Usability: The effort required to learn, operate, prepare input for, and interpret output of a program.

Maintainability: The effort required to locate and fix and error in a program.

Flexibility: The effort required to modify an operational program.

Testability: The effort required to test a program to ensure that it performs its intended function.

Portability: the effort required to transfer the program from one hardware and/or software system environment to another.

- *Reusability:* the extent to which a program can be reused in other applications-related to the packaging and scope of the functions that the program performs.

- *Interoperability:* The effort required to couple one system to another.

A Comment

- McCall's quality factors were proposed in the early 1970s. They are as valid today as they were in that time. It's likely that software built to conform to these factors will exhibit high quality well into the 21st century, even if there are dramatic changes in technology.

ISO 9126 Quality Factors:

- The ISO 9126 standard was developed in an attempt to identify quality attributes for computer software. The standard identifies six key quality attributes:
  - *Functionality:* The degree to which the software satisfies stated needs as indicated by the following sub-attributes: suitability, accuracy, interoperability, compliance and security.
  - *Reliability:* The amount of time that the software is available for use as indicated by the following sub-attributes: maturity, fault tolerance, recoverability.
  - *Usability:* the degree to which the software is easy to use as indicated by the following sub-attributes: understandability, learnability, operability.
  - *Efficiency:* The degree to which the software makes optimal use of system resources as indicated by the following sub-attributes: time, behavior, resource behavior.
  - *Maintainability:* The ease with which repair may be made to the software as indicated by the following sub-attributes: analyzability, changeability, stability, testability.
  - *Portability:* The ease with which the software can be transposed from one environment to another as indicated by the following sub-attributes: adaptability, installability, conformance, replaceabilty.

The Transition to a Quantitative View

- We examine a set of software metrics that can be applied to the quantitative assessment of software quality. In all cases, the metrics represent indirect measures; that is, we never really measure quality but rather some manifestation of quality. The complicating factor is the precise relationship between the variable that is measuring and the quality of software.

METRICS FOR THE ANALYSIS MODEL

- *Function-based metrics:* use the function point as a normalizing factor or as a measure of the "size" of the specification
- *Specification metrics:* used as an indication of quality by measuring number of requirements by type

Function-Based Metrics

- The *function point metric* (FP), first proposed by Albrecht, can be used effectively as a means for measuring the functionality delivered by a system.
- Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity
- Information domain values are defined in the following manner:
    - *Number of external inputs (EIs)*
    - *Number of external outputs (EOs)*
    - *Number of external inquiries (EQs)*
    - *Number of internal logical files (ILFs)*
    - *Number of external interface files (EIFs)*

- **Function Points**
    - To compute function points (FP), the following relationship is used:

$$\boldsymbol{FP = count\ total\ X\ [0.65 + 0.01\ X \sum (F_i)]} \quad (1)$$

Where count total is the sum of all FP entries obtained from figure.

Information Domain Value	Count		Weighting factor simple	average	complex		
External Inputs (EIs)	☐	X	3	4	6	=	☐
External Outputs (EOs)	☐		4	5	7	=	☐
External Inquiries (EQs)	☐		3	4	6	=	☐
Internal Logical Files (ILFs)	☐	X	7	10	15	=	☐
External Interface Files (EIFs)	☐		5	7	10	=	☐
Count total						→	☐

- The F_i(i= 1 to 14) are *value adjustment factors* based on responses to the following questions:

1. Does the system require reliable backup and recovery?
2. Are specialized data communications required to transfer information to or form the application?
3. Are there distributed processing functions?
4. Is performance critical?
5. Will the system run in an existing, heavily utilized operational environment?
6. Does the system require on-line data entry?
7. Does the on-line data entry require the input transaction to be built over multiple screens or operations?
8. Are the ILFs updated on-line?
9. Are the inputs, outputs, files or inquiries complex?
10. Is the internal processing complex?

11. Is the code designed to be reusable?
12. Are conversion and installation included in the design?
13. Is the system designed for multiple installations in different organizations?
14. Is the application designed to facilitate change and for ease of use by the user?

Each of these questions is answered using a scale that ranges from **0 to 5.**

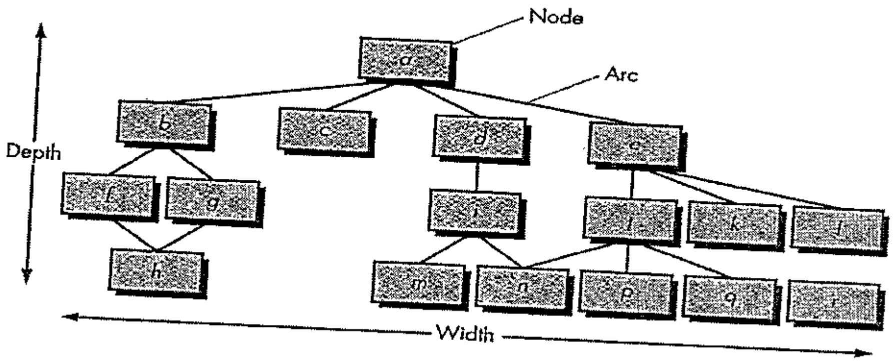
The constant values in *equation-1* and the weighting factors that are applied to information domain counts are determined empirically.

METRICS FOR THE DESIGN MODEL

- Design metrics for computer software, like all other software metrics, are not perfect. Debate continues over their efficacy and the manner in which they should be applied.

Architectural Design Metrics

- Architectural design metrics focus on characteristics of the program architecture with an emphasis on the architectural structure and the effectiveness of modules or components within the architecture.
- Architectural design metrics
  - Structural complexity = g(fan-out)
  - Data complexity = f(input & output variables, fan-out)
  - System complexity = h(structural & data complexity)
- HK metric: architectural complexity as a function of fan-in and fan-out
- Morphology metrics: a function of the number of modules and the number of interfaces between modules



- For hierarchical architectures structural complexity of a module I is defined in the following manner:
  - $S(i) = f^2_{out}(i)$

    Where $f_{out}(i)$ is the fan-out of module i.

- Data complexity provides an indication of the complexity in the internal interface for a module I and is defined as
  $D(i) = v(i) / [f_{out}(i) + 1]$

  Where $v(i)$ is the number of input and output variables that are passed to and from module i.

  Finally, system complexity is defined as the sum of structural and data complexity specified as

  $$C(i) = S(i) + D(i)$$

- As each of these complexity values increases, the overall architectural complexity or the system also increases. This leads to a greater likelihood that integration and testing effort will also increase.
- Fenton suggests a number of simple morphology metrics that enable different program architectures to be compared using a set of straight forward dimensions. Referring to the call-and-return architecture in figure. The following metric can be defined:

size $= n + a$

where n is the number of nodes and a is the number of arcs.

- The U.S. Air Force Systems command has developed a number of software quality indicators that are based on measurable design characteristics of a computer program. The Air Force uses information obtained form data and architectural design to derive a design structure quality index that ranges from 0 to 1. The following values must be ascertained to compute the DSQI.

$S_1 =$ the total number of modules defined in the program architecture

$S_2 =$ the number of modules whose correct function depends on the source of data input or that produce data to be used elsewhere.

$S_3 =$ the number of modules whose correct function depends on prior processing.

$S_4 =$ the number of database items.

$S_5 =$ the total number of unique database items.

$S_6 =$ the number of database segments

$S_7 =$ the number of modules with a single entry and exit.

- *Program Structure*: D_1, where $D_1$is defined as follows: If the architectural design was developed using a distinct method, then D_1= 1, otherwise D_1=0.

  Module independence: $D_2 = 1 - (S_2/S_1)$

  Modules not dependent on prior processing: $D_3 = 1 - (S_3/S_1)$

  Database size: D_4= 1- (S_5/S_4)

  Database compartmentalization: $D_5 = 1 - (S_6/S_4)$

  Module entrance/exit characteristic: D_6= 1 – (S_7/S_1)

  With these intermediate values determined, the DSQI is computed in the following manner:

  DSQI = $\sum w_i D_i$

  Where i = 1 to 6, w_i is the relative weighting of the importance of each of the intermediate values, and $\sum w_i = 1$

Metrics for Object-Oriented Design

- Whitmire describes nine distinct and measurable characteristics of an OO design:
  - Size
    - Size is defined in terms of four views: population, volume, length, and functionality
  - Complexity
    - How classes of an OO design are interrelated to one another
  - Coupling
    - The physical connections between elements of the OO design
  - Sufficiency
    - "the degree to which an abstraction possesses the features required of it, or the degree to which a design component possesses features in its abstraction, from the point of view of the current application."
  - Completeness
    - An indirect implication about the degree to which the abstraction or design component can be reused
  - Cohesion
    - The degree to which all operations working together to achieve a single, well-defined purpose
  - Primitiveness
    - Applied to both operations and classes, the degree to which an operation is atomic
  - Similarity
    - The degree to which two or more classes are similar in terms of their structure, function, behavior, or purpose

- Volatility
  - Measures the likelihood that a change will occur

Class-Oriented Metrics

The class is the fundamental unit of an object oriented system.

- *Proposed by Chidamber and Kemerer Metrics Suite*
  weighted methods per class
  depth of the inheritance tree
  number of children
  coupling between object classes
  response for a class
  lack of cohesion in methods
- *The MOOD Metrics Suite*
  Method inheritance factor:
  Coupling factor
  Polymorphism factor

Proposed by Lorenz and Kidd:
class size
number of operations overridden by a subclass
number of operations added by a subclass
specialization index

Component-Level Design Metrics

- *Cohesion metrics:* a function of data objects and the locus of their definition
- *Coupling metrics:* a function of input and output parameters, global variables, and modules called
- *Complexity metrics:* hundreds have been proposed (e.g., cyclomatic complexity)

Operation-Oriented Metrics

Proposed by Lorenz and Kidd:

- average operation size
- operation complexity
- average number of parameters per operation

User Interface Design Metrics

- *Layout appropriateness:* a function of layout entities, the geographic position and the "cost" of making transitions among entities.

METRICS FOR SOURCE CODE

- Also called Halstead's Software Science: a comprehensive collection of metrics all predicated on the number (count and occurrence) of operators and operands within a component or program
  - It should be noted that Halstead's "laws" have generated substantial controversy, and many believe that the underlying theory has flaws. However,

experimental verification for selected programming languages has been performed. The measures are

- n_1 = the number of distinct operators that appear in a program.
- n_2 = the number of distinct operands that appear in a program.
- N_1 = the total number of operator occurrences.
- N_2 = the total number of operand occurrences.
- Halstead shows that length N can be estimated

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

and program volume may be defined.

$$V = N \log_2 (n_1+n_2)$$

- It should be noted that V will vary with programming language and represents the volume of information required to specify a program.
- Theoretically, a minimum volume must exist for a particular algorithm. Halstead defines a volume ratio L as the ratio of volume of the most compact form of a program to the volume of the actual program.

  L must always be less than 1. In terms of primitive measures, the volume ratio may be expressed as

$$L = 2/n_1 \times n_2/N_2$$

METRICS FOR TESTING

- Modules with high cyclomatic complexity are more likely to be error prone than modules whose cyclomatic complexity is lower. For this reason, the tester should expand above average effort to uncover errors in such modules before they are integrated in a system.

Halstead Metrics Applied to Testing

- Testing effort can also be estimated using metrics derived from Halstead measures
  - Using the definitions for program volume, V, and program level, PL, Halstead effort, e, can be computed as

    $PL = 1 / [n_1/2) X (N_2/n_2)]$and $e = V/PL$

- The percentage of overall testing effort to be allocated to a module k can be estimated using the following relationship:Percentage of testing effort $(k) = e(k) / \sum e(i)$
  Where $e(k)$ is computed for module k using equations and the summation in the denominator of equation is the sum of Halstead effort across all modules of the system.

Metrics for Object-Oriented Testing

- Binder suggests a broad array of design metrics that have a direct influence on the "testability" of an OO system.
  - *Lack of cohesion in methods (LCOM):* The higher the value of LCOM, the more states must be tested to ensure that methods do not generate side effects.
  - *Percent public and protected (PAP)*: This metric indicates the percentage of class attributes that are public or protected.
  - *Public access to data members (PAD)*: This metric indicates the number of classes that can be access another class's attributes, a violation of encapsulation.
  - *Number of root classes (NOR)*: This metric is a count of the distinct class hierarchies that are described in the design model.
  - *Fan-in (FIN)*: When used in the OO context, fan-in for the inheritance hierarchy is an indication of multiple inheritance. FIN>1 indicates that a class inherits its attributes and operations from more than one root class.
  - *Number of children (NOC) and depth of the inheritance tree (DIT)*: Superclass methods will have to be retested for each subclass.

METRICS FOR MAINTENANCE

- IEEE Std. 982.1 – 1988 suggests a software maturity index that provides an indication of the stability of a software product. The following information is determined:

M_T = the number of modules in the current release.

F_c = the number of modules in the current release that have been changed.

F_a = the number of modules in the current release that have been added.

F_d = the number of modules from the preceding release that were deleted in the current release.

The software maturity index is computed in the following manner:

$$\text{SMI} = [M_r - (F_a + F_c + F_d)]/M_T$$

As SMI approaches 1.0, the product begins to stabilize. SMI may also be used as a metric for planning software maintenance activities. The mean time to produce a release of a software product can be correlated with SMI, and empirical models for maintenance effort can be developed.

METRICS FOR PROCESS & PROJECTS

- Software process and project metrics are quantitative measures that enable software engineers to gain insight into the efficacy of the software process and the projects that are conducted using the process as a framework.

SOFTWARE MEASUREMENT

- Software measurement can be categorized in two ways.
  - *Direct measures*of the software process and product.
  - *Indirect measures*of the product that includes functionality, quality, complexity, efficiency, reliability, maintainability.

Size-Oriented Metrics

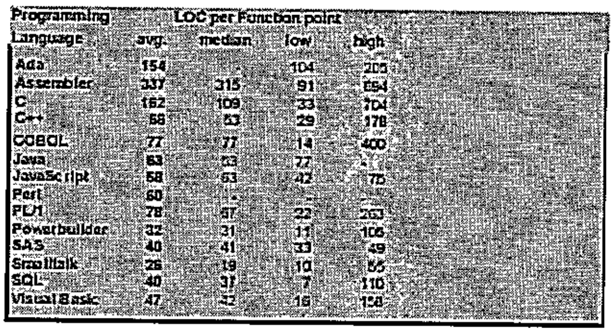
- errors per KLOC (thousand lines of code)
- defects per KLOC
- $ per LOC
- pages of documentation per KLOC
- errors per person-month
- Errors per review hour
- LOC per person-month
- $ per page of documentation

Function-Oriented Metrics

- errors per FP (thousand lines of code)
- defects per FP
- $ per FP
- pages of documentation per FP
- FP per person-month

Reconciling LOC and FP Metrics

- The relationship between lines of code and function points depend upon the programming language that is used to implement the software and the quality of the design. A number of studies have attempted to relate FP and LOC measures.
- The following table provides rough estimates of the average number of lines of code required to build one function point in various programming languages.



Programming	LOC per Function point			
Language	avg.	median	low	high
Ada	154		104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
Java	63	53	77	-
JavaScript	58	63	42	75
Perl	60	-	-	
PL/1	78	67	22	263
Powerbuilder	32	31	11	105
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

Why Opt for FP?

Programming language independent.

Used readily countable characteristics that are determined early in the software process.
Does not "penalize" inventive (short) implementations that use fewer LOC that other clumsier version.
Makes it easier to measure the impact of reusable components.

Object-Oriented Metrics

Number of scenario scripts (use-cases): A Scenario script is a detailed sequence of steps that describes the interaction between the user and the application.

Number of Key classes: Key classes are the "highly independent components" that are defined early in object-oriented analysis.
Number of support classes: Supports Classes are required to implement the system but are not immediately related to the problem domain.
Average number of support classes per key class (analysis class): The average number of support classes per key class were known for a given problem domain, estimating would be much simplified.
Number of subsystems: A subsystem is an aggregation of classes that support a function that is visible to the end-user of a system.

Use-Case Oriented Metrics

- A normalization measure similar to LOC and FP.
- Used for estimation before significant modeling and construction activities.
- Independent of programming languages.
- No standard size for use-case.

Web Engineering Project Metrics

Number of static Web pages (the end-user has no control over the content displayed on the page)
Number of dynamic Web pages (end-user actions result in customized content displayed on the page)
Number of internal page links (internal page links are pointers that provide a hyperlink to some other Web page within the WebApp)
Number of persistent data objects
Number of external systems interfaced
Number of static content objects
Number of dynamic content objects
Number of executable functions

Metrics For Software Quality

- The overriding goal of software engineering is to produce a high quality system, application, or product within a timeframe that satisfies a market need. To achieve this goal:
  Software Engineer must apply effective methods coupled with modern tools.

Software Engineer must measure a high quality is to be realized.
Private metrics collected are assimilated to provide project level results.
This metrics provide the effectiveness of individual and group software quality assurance and control activities.
Error data can also be used to compute defect removal efficiency for each process framework activity.

Measuring Quality

Correctness — the degree to which a program operates according to specification
Maintainability—the degree to which a program is amenable to change
Integrity—the degree to which a program is impervious to outside attack
Usability—the degree to which a program is easy to use

Defect Removal Efficiency (DRE)

- A quality metric that provides benefits at both the project and process level is defect removal efficiency.
- When considered for a project as a whole, DRE is defined in the following manner:
  DRE = $E / (E + D)$

Where E is the number of errors found before delivery of the software to the end-user

D is the number of defects found after delivery.

RISK MANAGEMENT

Project Risks

What can go wrong?

What is the likelihood?

What will the damage be?

What can we do about it?

REACTIVE vs PROACTIVE RISK STRATEGIES

Reactive Risk Management

project team reacts to risks when they occur
mitigation—plan for additional resources in anticipation of fire fighting
fix on failure—resource are found and applied when the risk strikes

- crisis management—failure does not respond to applied resources and project is in jeopardy

Proactive Risk Management

formal risk analysis is performed
organization corrects the root causes of risk

- TQM concepts and statistical SQA

- examining risk sources that lie beyond the bounds of the software
- developing the skill to manage change

SOFTWARE RISKS

Risk always involves two characteristics

- Uncertainty : the risk may or may not happen; that is, there are no 100% probable risks.
- Loss : if the risk becomes a reality, unwanted consequences or losses will occur.

Project risks threaten the project plan. Project risks identify potential budgetary, schedule, personal, resource, stakeholder, and requirements problems and their impact on a software project.

Technical risks threaten the quality and timeliness of the software to be produced. If a technical risk becomes a reality, implementation may become difficult or impossible. Technical risks occur because the problem is harder to solve than we thought it would be.

- *Business risks* threaten the viability of the software to be built.
  - The top five business risks are:

    (1) Building an excellent product or system that no one really ants,

    (2) Building a product that no longer fits into the overall business strategy for the company,

    (3) Building a product that the sales force doesn't understand how to sell,

    (4) Losing the support of senior management due to a change in focus or a change in people and

    (5) Losing budgetary or personnel commitment.

- Predictable risks are extrapolated from past project experience.
- Unpredictable risks are the joker in the deck. They can and do occur, but they are extremely difficult to identify in advance.

Seven principles of risk management

1. *Maintain a global perspective*—view software risks within the context of system and the business problem
2. *Take a forward-looking view*—think about the risks that may arise in the future; establish contingency plans
3. *Encourage open communication*—if someone states a potential risk, don't discount it.
4. *Integrate*—a consideration of risk must be integrated into the software process

5. *Emphasize a continuous process*—the team must be vigilant throughout the software process, modifying identified risks as more information is known and adding new ones as better insight is achieved.
6. *Develop a shared product vision*—if all stakeholders share the same vision of the software, it likely that better risk identification and assessment will occur.
7. *Encourage teamwork*—the talents, skills and knowledge of all stakeholder should be pooled

RISK IDENTIFICATION

Product size—risks associated with the overall size of the software to be built or modified.

Business impact—risks associated with constraints imposed by management or the marketplace.

Customer characteristics—risks associated with the sophistication of the customer and the developer's ability to communicate with the customer in a timely manner.

Process definition—risks associated with the degree to which the software process has been defined and is followed by the development organization.

Development environment—risks associated with the availability and quality of the tools to be used to build the product.

- *Technology to be built*—risks associated with the complexity of the system to be built and the "newness" of the technology that is packaged by the system.
- *Staff size and experience*—risks associated with the overall technical and project experience of the software engineers who will do the work.

Assessing Project Risk

Have top software and customer managers formally committed to support the project?
Are end-users enthusiastically committed to the project and the system/product to be built?
Are requirements fully understood by the software engineering team and their customers?
Have customers been involved fully in the definition of requirements?
Do end-users have realistic expectations?
Is project scope stable?
Does the software engineering team have the right mix of skills?
Are project requirements stable?
Does the project team have experience with the technology to be implemented?
Is the number of people on the project team adequate to do the job?
Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

Risk Components

- *performance risk*—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- *cost risk*—the degree of uncertainty that the project budget will be maintained.
- *support risk*—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.

- *schedule risk*—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

RISK PROJECTION

- *Risk projection*, also called *risk estimation*, attempts to rate each risk in two ways
    - the likelihood or probability that the risk is real
    - the consequences of the problems associated with the risk, should it occur.
- The are four risk projection steps:
    - establish a scale that reflects the perceived likelihood of a risk
    - delineate the consequences of the risk
    - estimate the impact of the risk on the project and the product,
    - note the overall accuracy of the risk projection so that there will be no misunderstandings.

Developing a Risk Table

Risk	Probability	Impact	RMMM
			Risk Mitigation Monitoring & Management

- Estimate the probability of occurrence
- Estimate the impact on the project on a scale of 1 to 5, where
    - 1 = low impact on project success
    - 5 = catastrophic impact on project success
- sort the table by probability and impact

Assessing Risk Impact

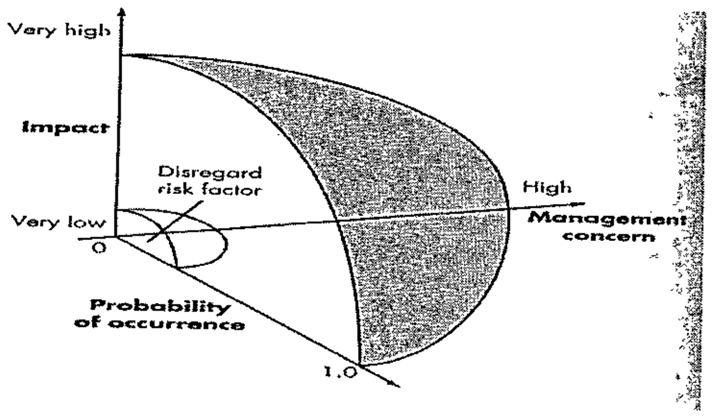
- The overall *risk exposure,* RE, is determined using the following relationship:

  $RE = P \times C$

  where

  P is the probability of occurrence for a risk, and

  C is the cost to the project should the risk occur.



RISK REFINEMENT

- During early stages of project planning, a risk may be stated qu ally. As time passes and more is learned about the project and the risk, it may be possible to refine the risk into a set of more detailed risks, each somewhat easier to mitigate, monitor, and manage.
- One way to do this is to represent the risk in condition transition-consequence format. That is, the risk is stated in the following form:

  - Given that <condition> then there is concern that <consequence>.
- This general condition can be refined in the following manner:
  Subcondition1: Certain reusable components were developed by a third party will no knowledge of internal design standards.
  Subcondition2: The design standard for component interfaces has not been solidified and may not conform to certain existing reusable components.
  Subcondition3: Certain reusable components have been implemented in a language that is not supported on the target environment.

RISK MITIGATION, MONITORING, AND MANAGEMENT

Mitigation—how can we avoid the risk?
Monitoring—what factors can we track that will enable us to determine if the risk is becoming more or less likely?
Management—what contingency plans do we have if the risk becomes a reality?

THE RMMM PLAN

The RMMM plan documents all work performed as part of risk analysis and are used by the project manager as part of the overall project plan.
Some software teams do not develop a formal RMMM document. Rather, each risk is documented individually using a Risk Information Sheet (RIS).
In most cases, the RIS is maintained using a database system, so that creation and information entry, priority ordering, searches, and other analysis may be accomplished easily.
Once RMMM has been documented and the project has begun, risk mitigation and monitoring steps commence.
Risk monitoring is a project tracking activity with three primary objectives:

1. To assess whether predicted risks do, in fact, occur.
2. To ensure that risk aversion steps defined for the risk are being properly applied.
3. To collect information that can be used for future risk analysis.

Risk Information Sheet
Project: Embedded software for XYZ system.
Risk type: schedule risk
Priority (1 low ... 5 critical): 4
Risk factor: Project completion will depend on tests which require hardware component under development. Hardware component delivery may be delayed.
Probability: 60 %
Impact: Project completion will be delayed for each day that hardware is unavailable for use in software testing.
Monitoring approach:
Scheduled milestone reviews with hardware group.
Contingency plan:
Modification of testing strategy to accommodate delay using software simulation.
Estimated resources: 6 additional person months beginning 7-1-96

QUALITY MANAGEMENT

Quality Concepts

- Variation control is the heart of the quality control. A manufacturer wants to minimize the variation among the products that are produced.
- ■ **Quality**
- The *American Heritage Dictionary* defines *quality* as
  - "a characteristic or attribute of something."
- For software, two kinds of quality may be encountered:
  - Quality of design encompasses requirements, specifications, and the design of the system.
  - Quality of conformance is an issue focused primarily on implementation.

- user satisfaction = compliant product + good quality + delivery within budget and schedule

■ **Quality Control**

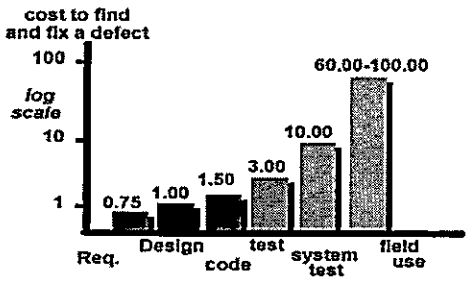
- Quality control involves the series of inspections, reviews, and tests used throughout the software process to ensure each work product meets the requirements placed upon it.
- Quality control includes a feedback loop to the process that created the work product.
- A key concept of quality control is that all work products have defined, measurable specifications to which h we may compare the output of each process.
- The feedback loop is essential to minimize the defects produced.

Quality Assurance

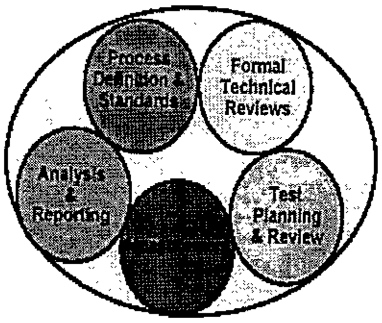
- Quality assurance consists of a set of auditing and reporting functions that assess the effectiveness and completeness of quality control activities.
- The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goals.

Cost of Quality

- *Prevention costs* include
  - quality planning
  - formal technical reviews
  - test equipment
  - Training
- *Internal failure costs* include
  - rework
  - repair
  - failure mode analysis
- *External failure costs* are
  - complaint resolution
  - product return and replacement
  - help line support
  - warranty work



SOFTWARE QUALITY ASSURANCE



- Software Quality can be defined as Conformance to explicitly state functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.
- Definition serves to emphasize three important points:

  - Software requirements are the foundation from which quality is measured. Lack of conformance to requirements is lack of quality.
  - Specified standard define a set of development criteria that guide the manner in which software is engineered. If the criteria are not followed, lack of quality will almost surely result.
  - A set of implicit requirements often goes unmentioned. If software conforms to its explicit requirements but fails to meet implicit requirements, software quality is suspect.

SQA Activities

- Prepares an SQA plan for a project.
  - The plan identifies
    - Evaluations to be performed.
    - Audits and reviews to be performed.
    - Standards that is applicable to the project.
    - Procedures for error reporting and tracking.
    - Documents to be produced by the SQA group.
    - Amount of feedback provided to the software project team.

- Participates in the development of the project's software process description.
- The SQA group reviews the process description for compliance with organizational policy, internal software standards, externally imposed standards (e.g., ISO-9001), and other parts of the software project plan.
- Reviews software engineering activities to verify compliance with the defined software process.

    - Identifies, documents, and tracks deviations from the process and verifies that corrections have been made.
- Audits designated software work products to verify compliance with those defined as part of the software process.
    - Reviews selected work products; identifies, documents, and tracks deviations; verifies that corrections have been made
    - Periodically reports the results of its work to the project manager.
- Ensures that deviations in software work and work products are documented and handled according to a documented procedure.
- Records any noncompliance and reports to senior management.
    - Noncompliance items are tracked until they are resolved.

SOFTWARE REVIEWS

- Software reviews are a "filter" for the software process. That is, reviews are applied at various points during software engineering and serve to uncover errors and defects that can then be removed.
- **What Are Reviews?**

  a meeting conducted by technical people for technical people
  a technical assessment of a work product created during the software engineering process
  a software quality assurance mechanism
  a training ground
- **What Reviews Are Not?**

  A project summary or progress assessment
  A meeting intended solely to impart information
  A mechanism for political or personal reprisal!

Cost impact of software defects

- The primary objective of formal technical reviews is to find errors during the process so that they do not become defects after release of the software.
- The obvious benefit of formal technical reviews is the early discovery of errors so that they do not propagate to the next step in the software process.
- To illustrate the cost impact of early error detection, we consider a series of relative costs that are based on actual cost data collected for large software projects.

Defect Amplification and Removal

- A defect amplification model can be used to illustrate the generation and detection of errors during the preliminary design, detail design, and coding steps of a software engineering process.
- During the step, errors may be inadvertently generated. Review may fail to uncover newly generated errors and errors from previous steps, resulting in some number of errors that are passed through.
- To conduct reviews, a software engineer must expend time and effort, and the development organization must spend money.

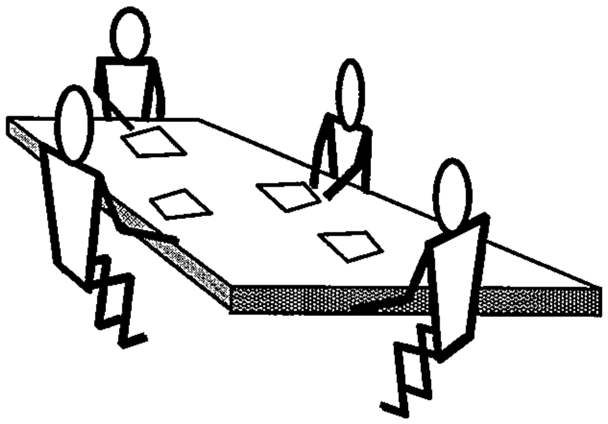
FORMAL TECHNICAL REVIEWS

- A formal technical review is a software quality control activity performed by software engineers.

- The objectives of formal technical reviews are:
    1. to uncover errors in function, logic, or implementation for any representation of the software.
    2. to verify that the software under review meets its requirements.
    3. to ensure that the software has been represented according to predefined standards.
    4. to achieve software that is developed in a uniform manner
    5. to make projects more manageable.

The Review Meeting

- Every review meeting should abide by the following constraints:
    1. Between three and five people should be involved in the review.
    2. Advance preparation should occur but should require no more than two hours of work for each person.
    3. The duration of the review meeting should be less than two hours.



be prepared—evaluate product before the review
review the product, not the producer
keep your tone mild, ask questions instead of making accusations
stick to the review agenda
raise issues, don't resolve them
avoid discussions of style—stick to technical correctness
schedule reviews as project tasks
record and report all review results

Review Reporting and Record Keeping

A review summary report answers three questions:

1. What was reviewed?
2. Who reviewed it?
3. What were the findings and conclusions?

- The review summary report is a single page form.

- The review issues list serves two purposes:

    1. To identify problem areas within the product
    2. To serve as an action item checklist that guides the producer as corrections are made. An issues list is normally attached to the summary report.

- It is important to establish a follow-up procedure to ensure that items on the issues list have been properly corrected.

Review Guidelines

- The following represents a minimum set of guidelines for formal technical reviews:
    1. Review the product, not the producer
    2. Set an agenda and maintain it
    3. Limit debate and rebuttal
    4. Enunciate problem areas
    5. Take written notes
    6. Limit the number of participants and insist upon advance preparation
    7. Develop a checklist for each product that is likely to be reviewed.
    8. Allocate resources and schedule time for FTRs
    9. Conduct meaningful training for all reviewers.
    10. Review your early reviews.

■ **Sample-Driven Reviews (SDRs)**

- SDRs attempt to quantify those work products that are primary targets for full FTRs.
- *To accomplish this ...*

    Inspect a fraction ai of each software work product, *i*. Record the number of faults, fi found within ai.

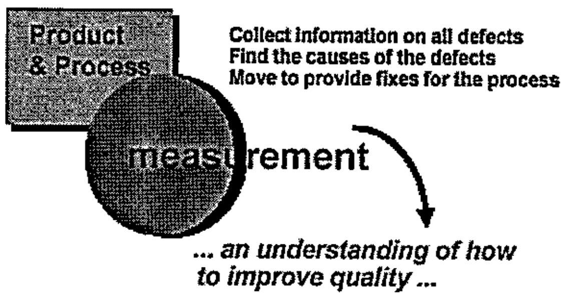
    Develop a gross estimate of the number of faults within work product *i* by multiplying fi by 1/ai.

    Sort the work products in descending order according to the gross estimate of the number of faults in each.

    Focus available review resources on those work products that have the highest estimated number of faults.

STATISTICAL SOFTWARE QUALITY ASSURANCE

- The software statistical quality assurance implies the following steps:
    1. Information about software defects is collected and categorized.
    2. An attempt is made to trace each defect to its underlying cause.
    3. Using the Pareto principle (80% of the defects can be traced to 20% of all possible causes), isolate the 20% (the "vital few").
    4. Once the vital few causes have been identified, move to correct the problems that have caused the defects.


Product & Process
Collect information on all defects
Find the causes of the defects
Move to provide fixes for the process
measurement
... an understanding of how to improve quality ...

Six-Sigma for Software Engineering

- The term "six sigma" is derived from six standard deviations—3.4 instances (defects) per million occurrences—implying an extremely high quality standard.
- The Six Sigma methodology defines three core steps:
    - *Define* customer requirements and deliverables and project goals via well-defined methods of customer communication
    - *Measure* the existing process and its output to determine current quality performance (collect defect metrics)
    - *Analyze* defect metrics and determine the vital few causes.
- If an existing software process is in place, but improvement is required, Six Sigma suggests two additional steps:
    - *Improve* the process by eliminating the root causes of defects.
    - *Control* the process to ensure that future work does not reintroduce the causes of defects.
- If an organization is developing a software process the core steps are augmented as follows:
    - *Design* the process to (1) avoid the root causes of defects and (2) to meet customer requirements.
    - *Verify* that the process model will, in fact, avoid defects and meet customer requirements.

SOFTWARE RELIABILITY

- Software reliability is defined in statistical terms as "the probability of failure-free operation of a computer program in a specified environment for a specified time.

Measures of reliability and Availability

- A simple measure of reliability is *mean-time-between-failure* (MTBF), where

$$\text{MTBF} = \text{MTTF} + \text{MTTR}$$

- The acronyms MTTF and MTTR are *mean-time-to-failure* and *mean-time-to-repair*, respectively.
- *Software availability* is the probability that a program is operating according to requirements at a given point in time and is defined as

$$\text{Availability} = [\text{MTTF}/(\text{MTTF} + \text{MTTR})] \times 100\%$$

Software Safety

- *Software safety* is a software quality assurance activity that focuses on the identification and assessment of potential hazards that may affect software negatively and cause an entire system to fail.
- If hazards can be identified early in the software process, software design features can be specified that will either eliminate or control potential hazards.

- A modeling and analysis is conducted as part of software safety. Initially, hazards are identified and categorized by criticality and risk. For example, some of the hazards associated with a computer-based cruise control for an automobile might be.

  Causes uncontrolled acceleration that cannot be stopped.
  Does not respond to depression of brake pedal.
  Does not engage when switch is activated.
  Slowly loses or gains speed.

THE ISO 9000 QULAITY STANDARDS

A quality assurance system may be defined as the organizational structure, responsibilities, procedures, processes, and resources for implementing quality management.
Quality assurance systems are created to help organizations ensure their products and services satisfy customer expectations by meeting their specifications.
ISO 9000 describes a quality assurance system in generic terms that can be applied to any business regardless of the products or services offered.

The ISO 9001:2000 Standard

ISO 9001:2000 is the quality assurance standard that applies to software engineering.
The standard contains 20 requirements that must be present for an effective quality assurance system.
The requirements delineated by ISO 9001:2000 address topics such as

- Management responsibility, quality system, contract review, design control, document and data control, product identification and traceability, process control, inspection and testing, corrective and preventive action, control of quality records, internal quality audits, training, servicing, and statistical techniques.

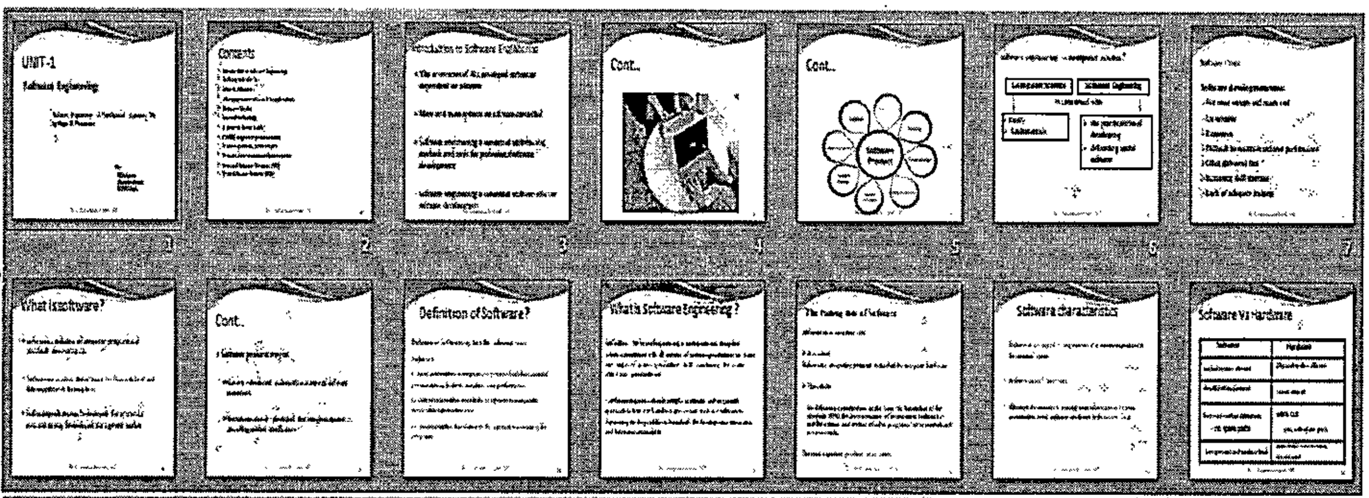
Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



UNIT-1

Software Engineering

Contents

Introduction to Software Engineering

Cont..

Cont..

Software Product

What is Software?

Cont..

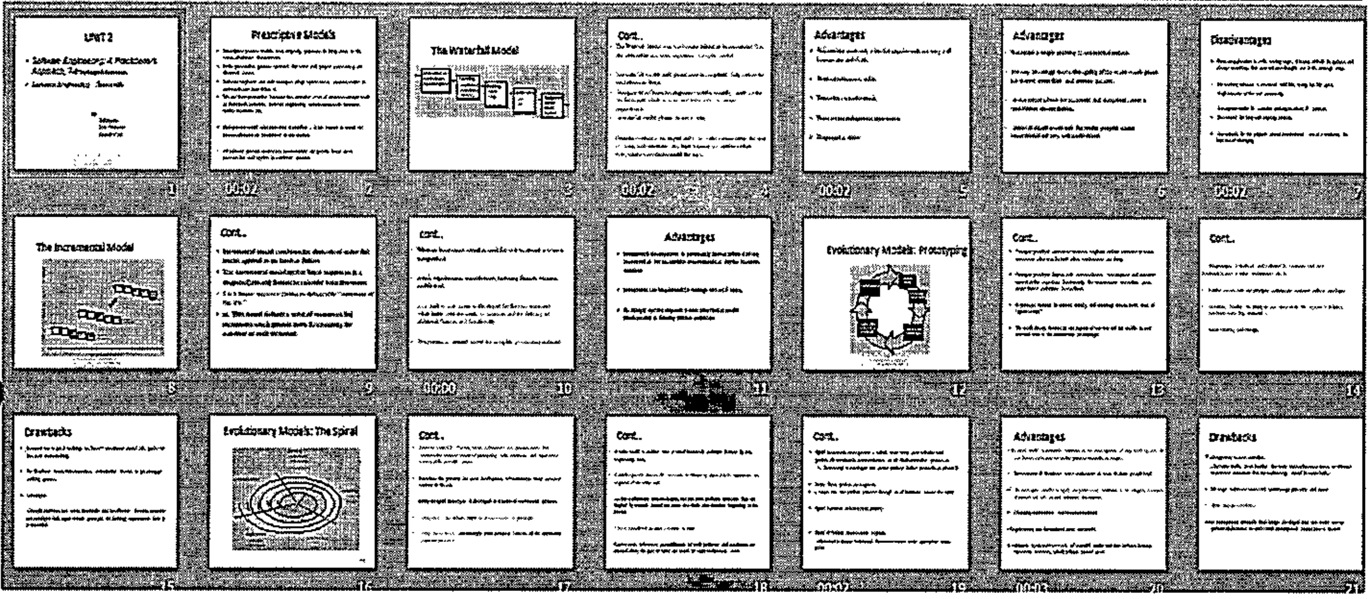
Definition of Software?

What is Software Engineering ?

Software characteristics

Software Vs Hardware

Software	Hardware
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]



UNIT 2

Prescriptive Models

The Waterfall Model

Cont..

Advantages

Advantages

Disadvantages

The Incremental Model

Cont..

Cont..

Advantages

Evolutionary Models: Prototyping

Cont..

Cont..

Drawbacks

Evolutionary Models: The Spiral

Cont..

Cont..

Cont..

Advantages

Drawbacks

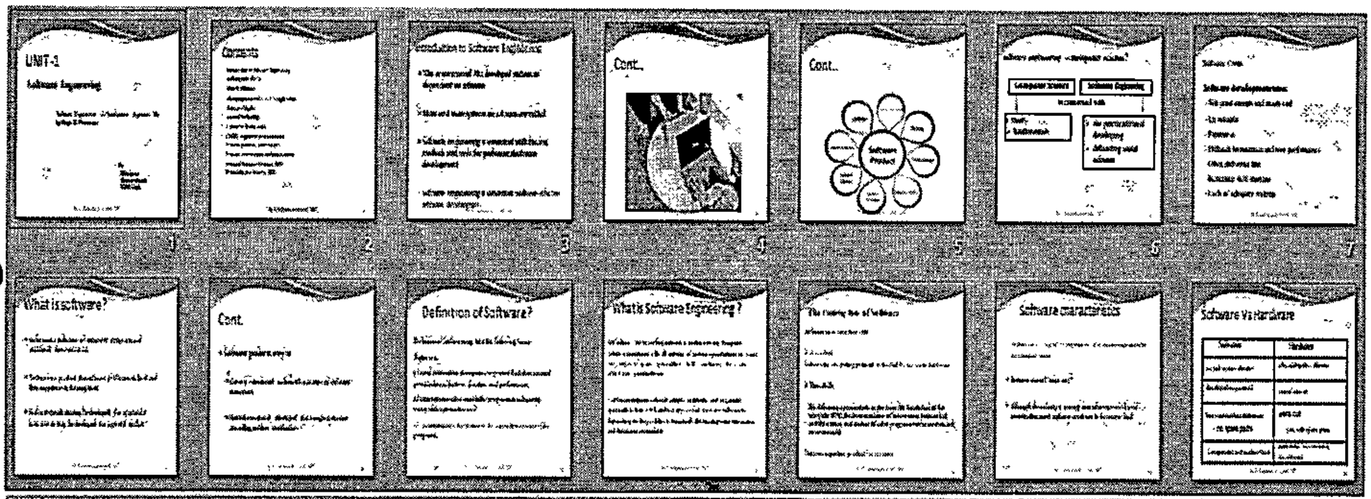
Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



UNIT-1 Software Engineering

Contents

Introduction to Software Engineering

Cont..

Cont..

[illegible]

Software Crisis

What is software?

Cont.

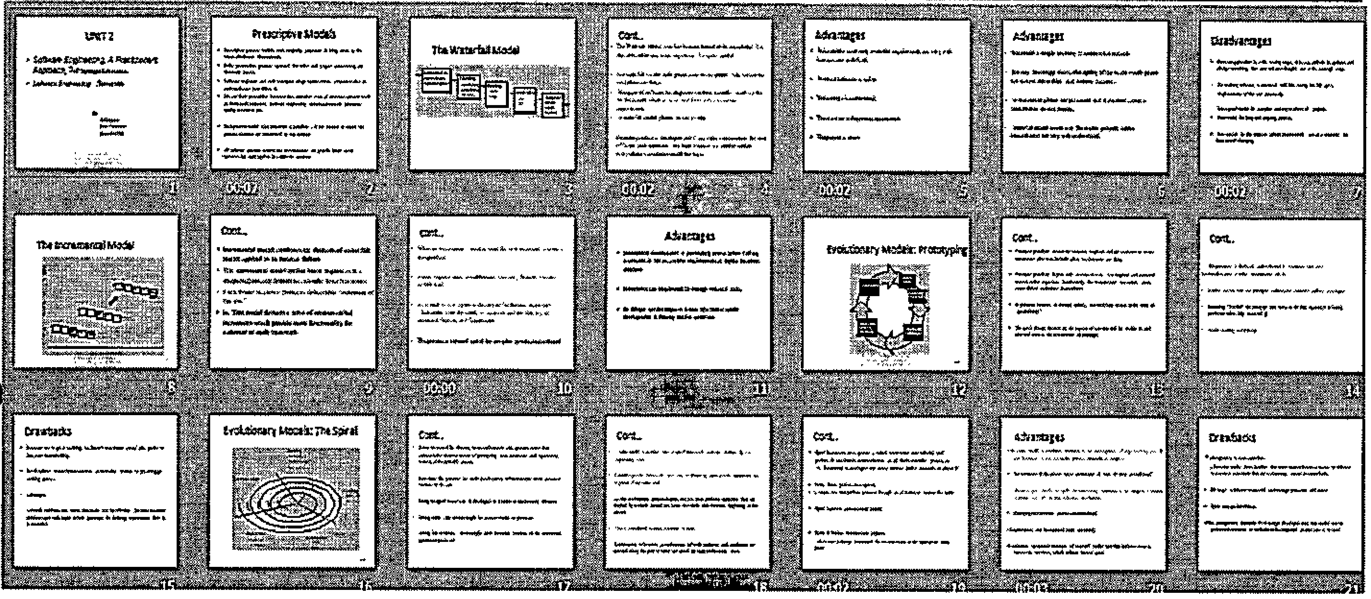
Definition of Software?

What is Software Engineering ?

The Changing Nature of Software

Software characteristics

Software Vs Hardware



UNIT 2

Prescriptive Models

The Waterfall Model

Cont..

Advantages

Advantages

Disadvantages

The Incremental Model

Cont..

Cont..

Advantages

Evolutionary Models: Prototyping

Cont..

Cont..

Drawbacks

Evolutionary Models: The Spiral

Cont..

Cont..

Cont..

Advantages

Drawbacks

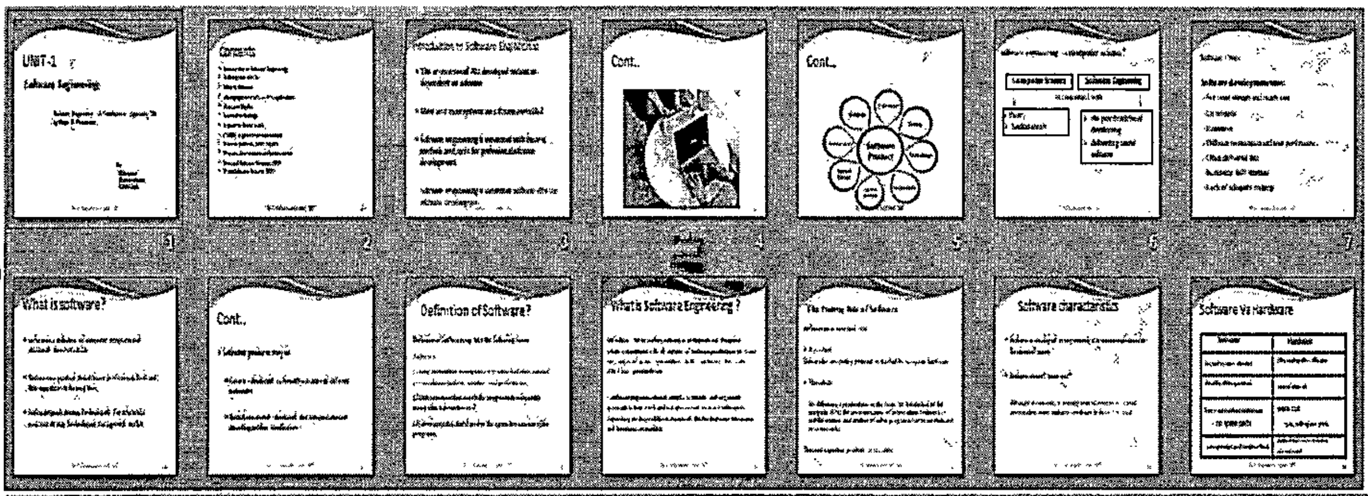
Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



UNIT-1

Software Engineering

Contents

Introduction to Software Engineering

Cont.,

Cont.,

What is software?

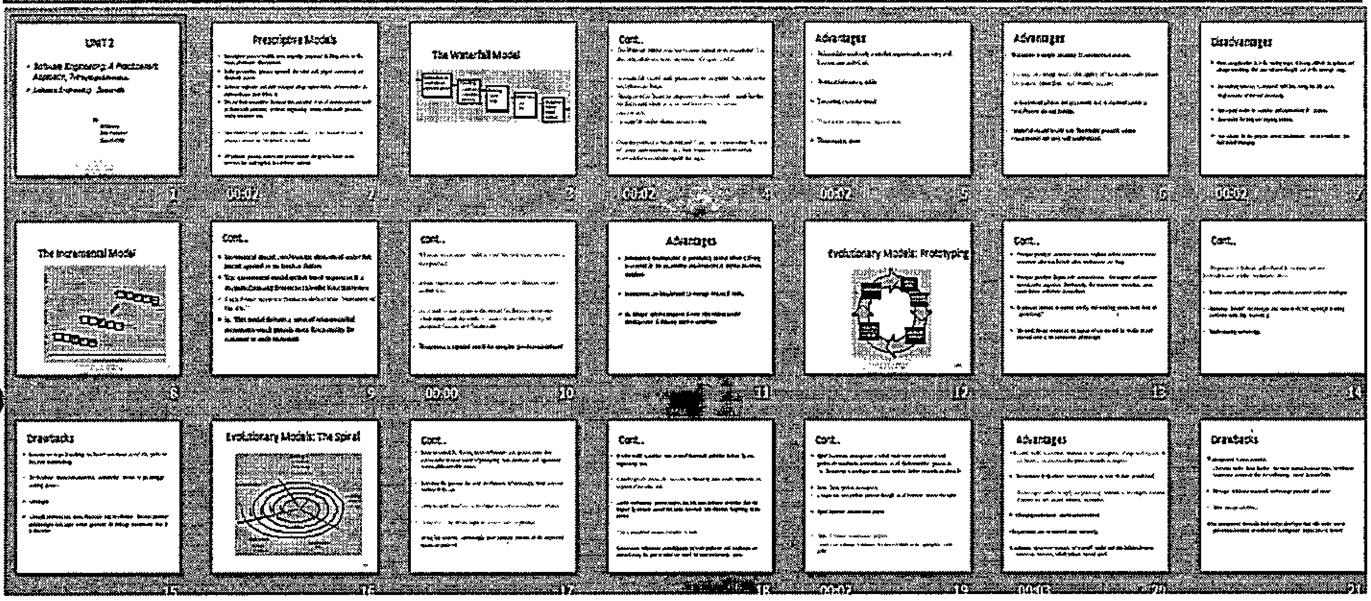
Cont.,

Definition of Software?

What is Software Engineering ?

Software characteristics

Software Vs Hardware



UNIT 2

Prescriptive Models

The Waterfall Model

Cont..

Advantages

Advantages

Disadvantages

The Incremental Model

Cont..

Cont..

Advantages

Evolutionary Models: Prototyping

Cont..

Cont..

Drawbacks

Evolutionary Models: The Spiral

Cont..

Cont..

Cont..

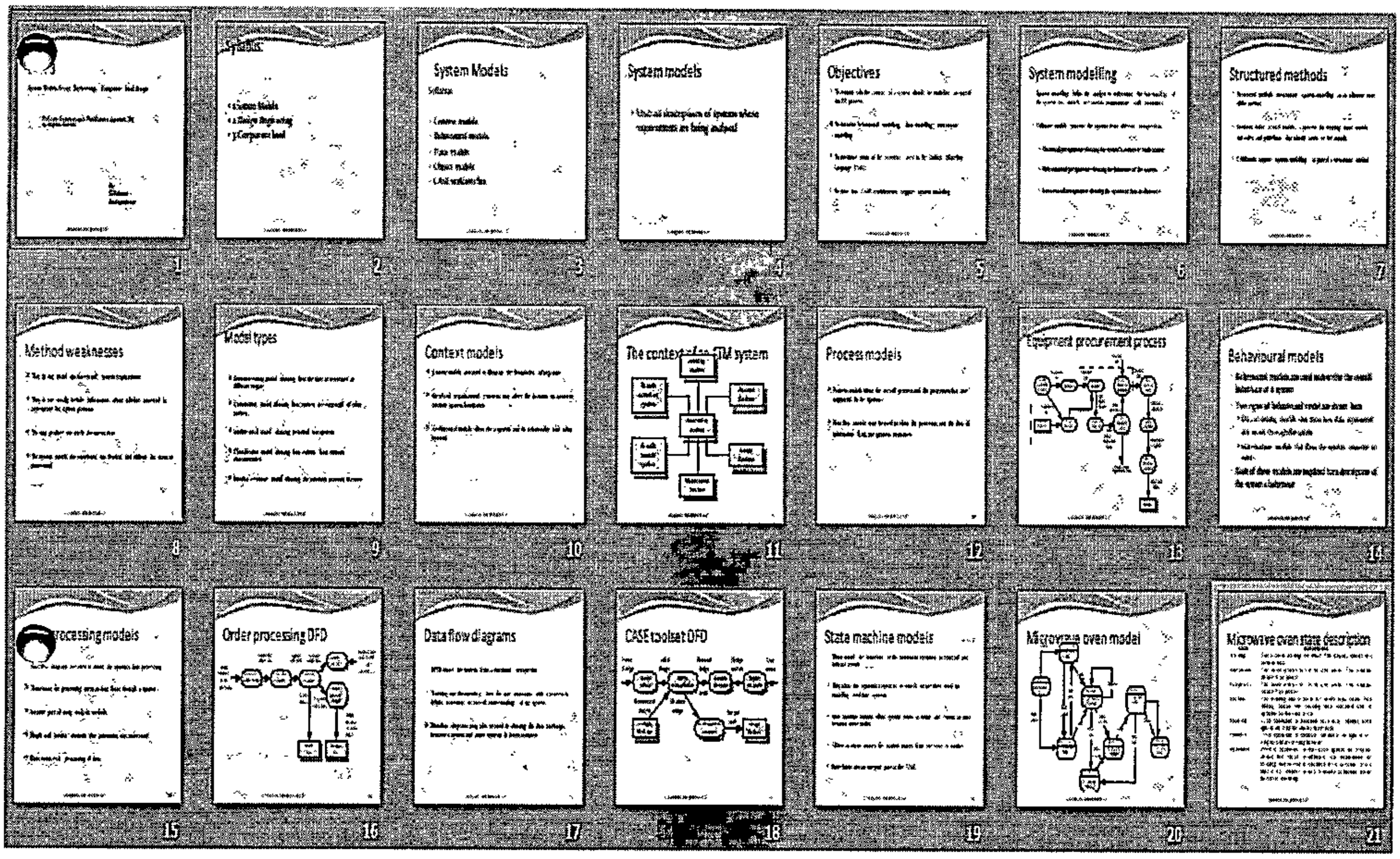
Advantages

Drawbacks

Vidya Jyothi Institute of Technology

An Autonomous Institution
(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



1 [illegible]

2 Syllabus

3 System Models

4 System models

5 Objectives

6 System modelling

7 Structured methods

8 Method weaknesses

9 Model types

10 Context models

11 The context of an ATM system

12 Process models

13 Equipment procurement process

14 Behavioural models

15 Data processing models

16 Order processing DFD

17 Data flow diagrams

18 CASE toolset DFD

19 State machine models

20 Microwave oven model

21 Microwave oven state description

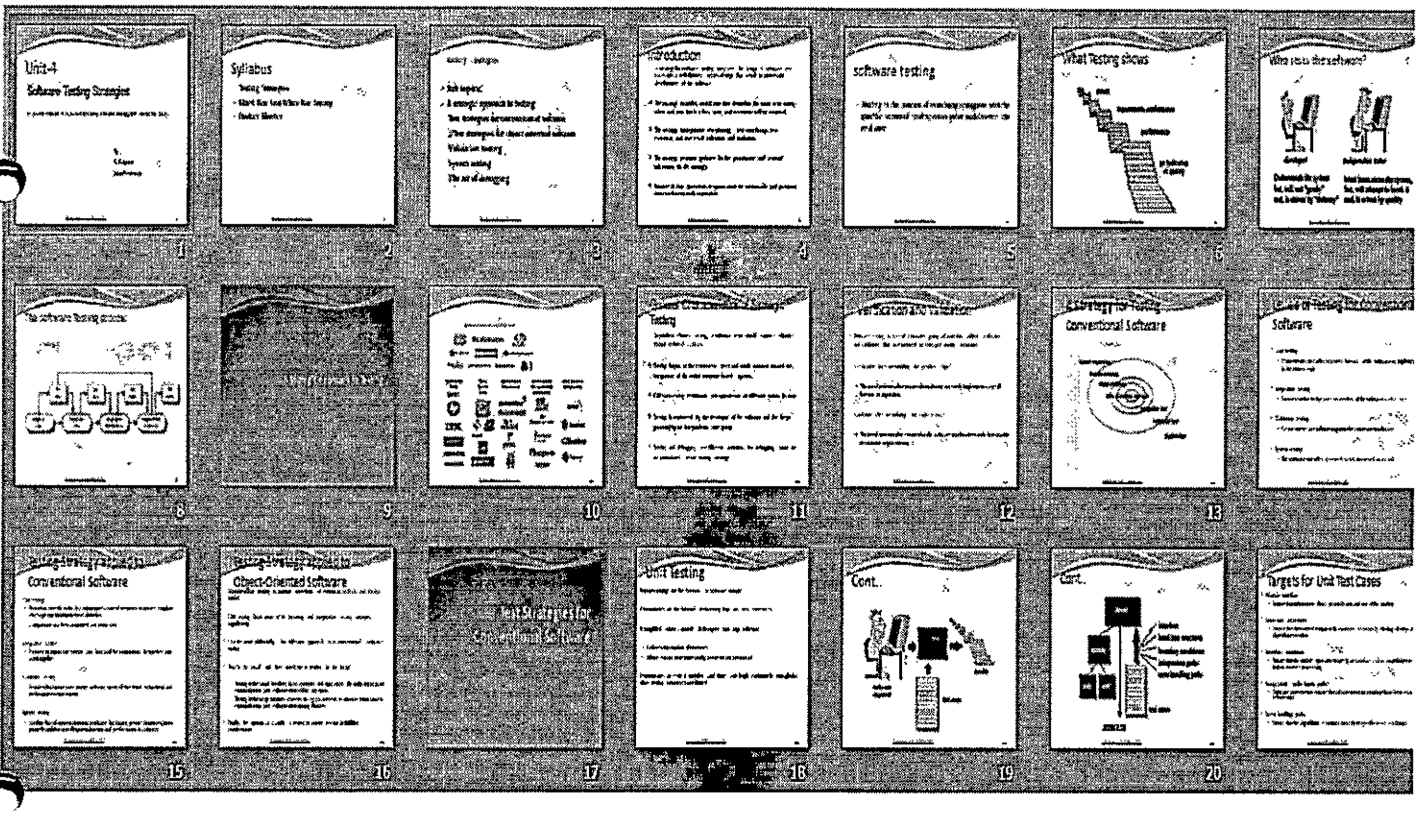
Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Unit-4 Software Testing Strategies

Syllabus

Introduction

software testing

What Testing shows

Who tests the software?

The software Testing process

Verification and Validation

A Strategy for Testing Conventional Software

Levels of Testing for Conventional Software

Testing Strategy applied to Conventional Software

Testing Strategy applied to Object-Oriented Software

Test Strategies for Conventional Software

Unit Testing

Cont..

Cont..

Targets for Unit Test Cases

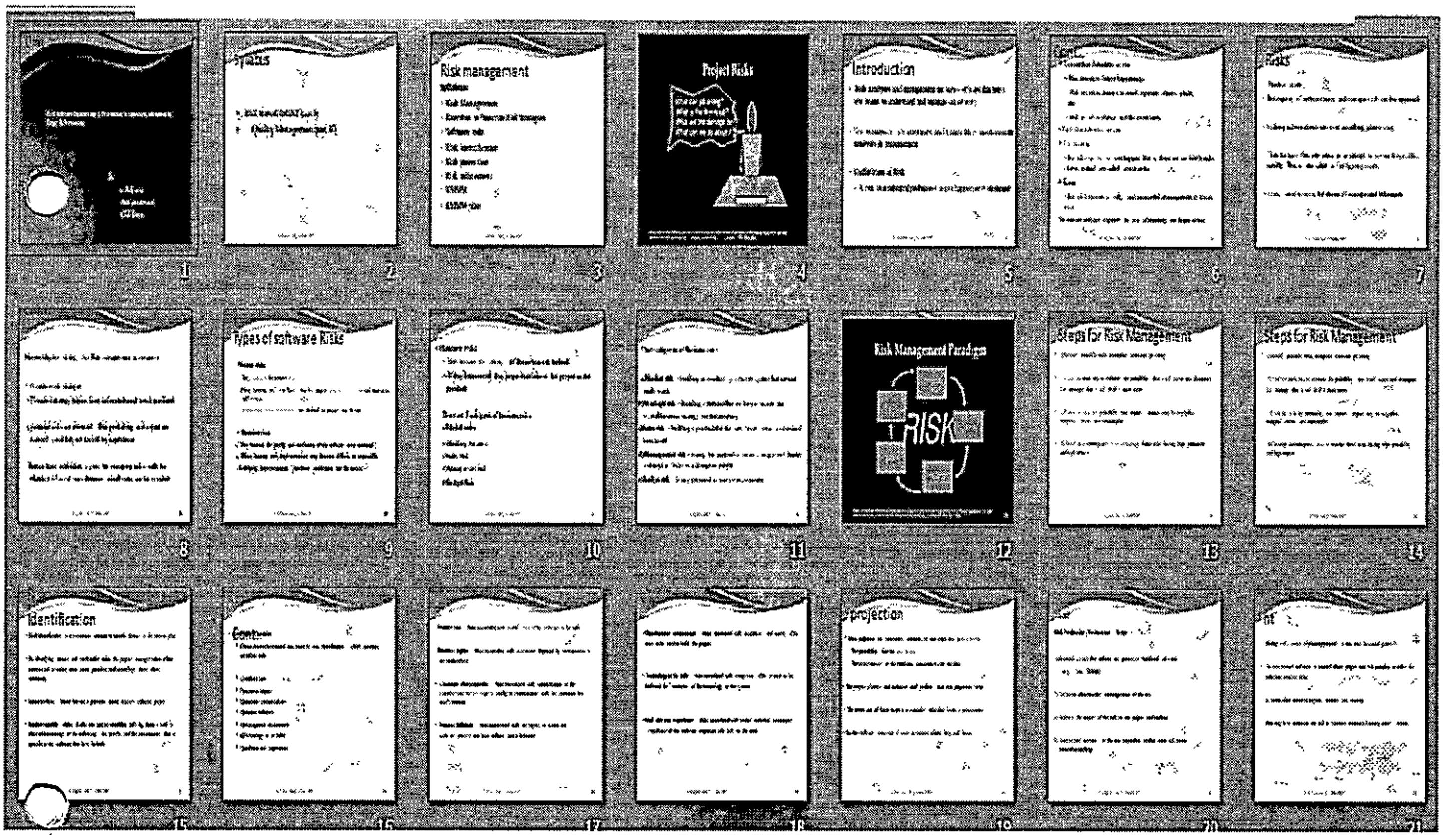
Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



Semester End Question Papers

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

Subject code: A14510

II B. Tech II SEM EXAMINATION(SUPPLY) DECEMBER 2017
SOFTWARE ENGINEERING - CSE & IT

Time: 3hrs **Max.Marks:75**

Note: This question paper contains two PARTS, PART A and B.
PART A is compulsory which carries 25 marks, Answer all questions.
PART B consists of 5 Units. Answer any one full question from each unit.

PART – A

ANSWER ALL THE QUESTIONS	**25M**
1. Explain evolving role of software?	2M
2. Define software. What are different types of software?	3M
3. Explain Functional requirements in detail?	2M
4. Explain Requirement Management?	3M
5. Discuss context model?	2M
6. Discuss User Interface design?	3M
7. Explain system Testing?	2M
8. Discuss about Risk Projection?	3M
9. Explain about software risks?	2M
10. Explain about Software Reliability?	3M

PART-B

ANSWER ALL THE QUESTIONS **5 x 10= 50 M**

11 i) a) Explain Software Engineering – A layered Technology?
b) Explain in detail about Software Applications?

OR

ii) a) Discuss about CMMI levels?
b) Explain software framework activities?

12 i) a) What is feasibility study?
b) Discuss about Evolutionary process models?

OR

ii) a) Explain in detail about user and system requirements?
b) Discuss Agile Process Model?

13 i) a) What is Design Engineering. Discuss about design concepts?
b) Explain object models?

OR

ii) a) Explain component model design?
b) Write short notes on A) user interface design B) software Architecture

14 i) a) Explain conventional approaches for software testing with an example?
b) Discuss Unit Testing and Integration Testing?

OR

ii) a) Explain about Metrics for Source Code?
b) What is software Quality?

15 i) a) Write about ISO 9000 quality standards
b) Write short notes on formal Technical reviews

OR

ii) a) Explain Metrics for software Quality?
b) Differentiate between Reactive vs. Proactive Risk

Vidya Jyothi Institute of Technology (Autonomous)

Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTUH, Hyderabad

(Aziz Nagar, C.B.Post, Hyderabad-500075) Sub Code:A14510 R15

II B.Tech II Semester Supplementary Examination, October/November-2020

Subjects Name: SOFTWARE ENGINEERING **BRANCH: CSE & IT**

Time: 2 Hours **Max Marks:75**

Note: This question paper contains EIGHT questions and the students are asked to answer any FIVE questions. Each question carries 15 marks.

Bloom's Level:

Remember	L1	Analyze	L4
Understand	L2	Evaluate	L5
Apply	L3	Create	L6

	ANSWER ANY FIVE QUESTIONS (5Qx15M=75M)	**Bloom's Level**	**Marks**
1	a) Discuss different software myths and true aspects of these myths. b) Demonstrate the process framework in detail.	L2,L2	15M
2	a) Analyze various levels of Capability Maturity Model Integration (CMMI). b) Describe the evolving role of software.	L2,L3	15M
3	a) Explain different checks to be carried out during requirement validation process. b) Illustrate incremental process model. State benefits and problems associated with it.	L2,L2	15M
4	a) Why the understanding of requirements from stake holders is difficult? Explain. b) Describe various activities performed in requirements elicitation phase with an example.	L2,L3	15M
5	a) Explain data models and object models. b) Describe the procedure for modeling component level design.	L2,L2	15M
6	a) Discuss about the design concepts in a software development process. b) Assess the user interface design of a software with an example and neat sketch.	L2,L3	15M
7	a) Explain the strategic approach to software testing. b) Differentiate black-box and white-box testing.	L2,L2	15M
8	a) Differentiate black-box and white-box testing. b) Explain in detail about the risk management in software development life cycle.	L2,L3	15M

VJIT(A)

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

Subject code: A14510

II B. Tech II SEM EXAMINATION(SUPPLY) DECEMBER 2017
SOFTWARE ENGINEERING - CSE & IT

Time: 3hrs **Max.Marks:75**

Note: This question paper contains two PARTS, PART A and B.
PART A is compulsory which carries 25 marks, Answer all questions.
PART B consists of 5 Units. Answer any one full question from each unit.

PART – A

ANSWER ALL THE QUESTIONS **25M**

1. Explain evolving role of software? 2M
2. Define software. What are different types of software? 3M
3. Explain Functional requirements in detail? 2M
4. Explain Requirement Management? 3M
5. Discuss context model? 2M
6. Discuss User Interface design? 3M
7. Explain system Testing? 2M
8. Discuss about Risk Projection? 3M
9. Explain about software risks? 2M
10. Explain about Software Reliability? 3M

PART-B

ANSWER ALL THE QUESTIONS **5 x 10= 50 M**

11 i) a) Explain Software Engineering – A layered Technology?
b) Explain in detail about Software Applications?

OR

ii) a) Discuss about CMMI levels?
b) Explain software framework activities?

12 i) a) What is feasibility study?
b) Discuss about Evolutionary process models?

OR

ii) a) Explain in detail about user and system requirements?
b) Discuss Agile Process Model?

13 i) a) What is Design Engineering. Discuss about design concepts?
b) Explain object models?

OR

ii) a) Explain component model design?
b) Write short notes on A) user interface design B) software Architecture

14 i) a) Explain conventional approaches for software testing with an example?
b) Discuss Unit Testing and Integration Testing?

OR

ii) a) Explain about Metrics for Source Code?
b) What is software Quality?

15 i) a) Write about ISO 9000 quality standards
b) Write short notes on formal Technical reviews

OR

ii) a) Explain Metrics for software Quality?
b) Differentiate between Reactive vs. Proactive Risk

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY (AUTONOMOUS) **R18**

Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTUH, Hyderabad)
(Aziz Nagar, C.B. Post, Hyderabad-500075)

Subject Code: A24510

B.Tech. II Year II Semester Regular Examinations, OCTOBER/NOVEMBER-2020

SUBJECT : SOFTWARE ENGINEERING BRANCH : CSE&IT

Time: 2 Hours Max. Marks:75

Note: This question paper contains EIGHT questions and answer any FIVE questions. Each question carries 15 marks.

Bloom's Level:

Remember	L1
Understand	L2
Apply	L3
Analyze	L4
Evaluate	L5
Create	L6

ANSWER ANY FIVE QUESTIONS	5QX15M = 75 M	Bloom's Level	Marks
1.a)	What is legacy software? Explain briefly its impact in software engineering.	L3	7M
b)	What are the five generic process framework activities? Explain.	L3	8M
2.a)	Discuss the architecture of Layered technology.	L6	7M
b)	Discuss capability maturity models.	L6	8M
3	"The functional requirements specification of a system should be both complete and consistent". Substantiate this statement with relevant examples.	L4	15M
4.a)	Why is traceability an important aspect of requirements management? Why context system models are useful for requirements validation?	L3	7M
b)	Compare iterative development with incremental delivery approach.	L2	8M
5.a)	What is data design? Explain how it is done with examples.	L3	7M
b)	Discuss the characteristics of good design.	L6	8M
6	Draw a sequence diagram that shows a possible sequence of actions that occur when a new article is catalogued by the LIBSYS system.	L6	15M
7.a)	Explain about the test strategies for conventional software.	L5	7M
b)	Explain the metrics for software quality.	L5	8M
8.a)	Discuss about software reviews.	L6	7M
b)	Discuss clearly about risk projection.	L6	8M

VJIT(A)

Extra Topics Delivered

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Extra Topics Added

SNO.	Extra Topic Added
1	Unified process Model
2	UML concepts and 9 diagrams
3	Extreme Programming
4	Adaptive Software Development
5	Scrum

Innovation in Teaching Learning

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

1)Interactive Learning : case based learning

Faculty Name: G.kalpana

Subject : Software Engineering

Topic: SRS preparation

Participants: Students of B.Tech IIyr /II Sem , CSE- D section

Content: THE REQUIREMENTS DOCUMENT

The requirements document is the official statement of what is required of the system developers Should include both a definition and a specification of requirements
It is NOT a design document.
As far as possible, it should set of WHAT the system should do rather than HOW it should do it.

Requirements document requirements!

Specify external system behaviour
Specify implementation constraints
Easy to change
Serve as reference tool for maintenance
Record forethought about the life cycle of the system i.e. predict changes
Characterise responses to unexpected events

REQUIREMENTS DOCUMENT STRUCTURE

Introduction
Glossary
User requirements definition
System architecture
System requirements specification
System models
System evolution
Appendices
Index

Implementation: I made the students into groups and i gave different case studies like Bank application, Hospital management, Point of sales, Library Management system etc. and i asked them to identify various types of requirements and prepare system Requirement Document.

Outcome: All the students are actively participated in this activity and i gave few suggestions for those who are facing difficulty in preparing SRS with that they understood the topic easily and completely.

[illegible]

(Faculty Incharge)

[illegible]

(CSE-HOD)

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-50075.

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Interactive Learning: Case Based Activity

Faculty Name: G.kalpana

Subject : Software Engineering

Topic: ***Object behaviour modelling***
Participants: II yr/II sem B.Tech CSE students – D section

Content:

Object behaviour modelling

A behavioural model shows the interactions between objects to produce some particular system behaviour that is specified as a use-case.

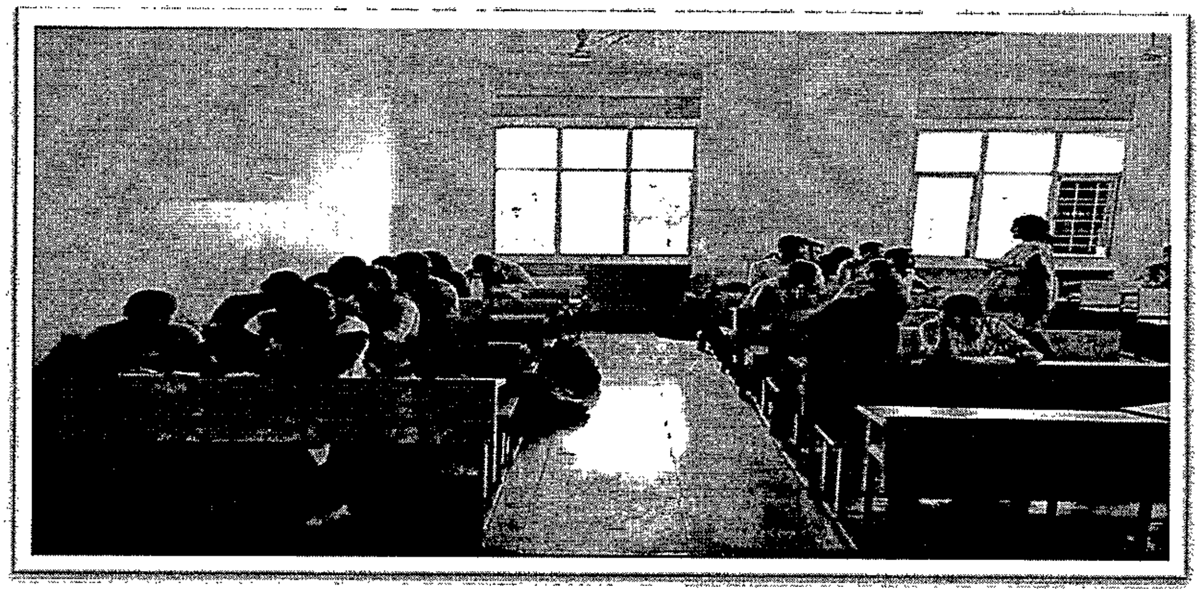
Sequence diagrams (or collaboration diagrams) in the UML are used to model interaction between objects.

- Sequence models show the sequence of object interactions that take place
  - Objects are arranged horizontally across the top;
  - Time is represented vertically so models are read top to bottom;
  - Interactions are represented by labelled arrows, Different styles of arrow represent different types of interaction;
  - A thin rectangle in an object lifeline represents the time when the object is the controlling object in the system.

Implementation: I made the students into some set of groups and also gave few case studies and asked them to draw sequence diagram as part of behavioral modeling. I gave a Bank case study to the students and asked them to identify objects and draw Sequence diagram as part part of behavioral modeling using Unified Modeling Language .finally they have explained what they have drawn and understood the concepts. Similarly two more case studies given ie POS and Hospital Management system.

Outcome:

With this activity all students are actively participated and have improved their knowledge in the topic. For those who have not followed i have explained with few more case studies.



[illegible]

(Faculty Incharge)

[illegible]

(CSE-HOD)

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-500075.

5) Interactive Learning : Think Pair share

Subject : Software Engineering

Topic: Function Point Calculation

Participants: Students of II/II D section

Think-pair-share (TPS) is a collaborative learning strategy where students work together to solve a problem or answer a question about an assigned reading. This strategy requires students to (1) think individually about a topic or answer to a question; and (2) share ideas with classmates. Discussing with a partner maximizes participation, focuses attention and engages students in comprehending the reading material.

Content:

- *Function-based metrics:* use the function point as a normalizing factor or as a measure of the "size" of the specification
- *Specification metrics:* used as an indication of quality by measuring number of requirements by type

Function-Based Metrics

- The *function point metric* (FP), first proposed by Albrecht, can be used effectively as a means for measuring the functionality delivered by a system.
- Function points are derived using an empirical relationship based on countable (direct) measures of software's information domain and assessments of software complexity
- Information domain values are defined in the following manner:
  - *Number of external inputs (EIs)*
  - *Number of external outputs (EOs)*
  - *Number of external inquiries (EQs)*
  - *Number of internal logical files (ILFs)*
  - *Number of external interface files (EIFs)*

- **Function Points**
  - To compute function points (FP), the following relationship is used:

$$FP = \text{count total} \times [0.65 + 0.01 \times \sum (F_i)] \quad (1)$$

Where count total is the sum of all FP entries obtained from figure.

Information Domain Value	Count		Weighting factor simple	average	complex		
External Inputs (EIs)		x	3	4	6	=	
External Outputs (EOs)			4	5	7	=	
External Inquiries (EQs)			3	4	6	=	
Internal Logical Files (ILFs)		x	7	10	15	=	
External Interface Files (EIFs)			5	7	10	=	
Count total ⟶							

- The F_i(i= 1 to 14) are *value adjustment factors* based on responses to the following questions:

Implementation: As part of this activity students are asked to find out the function point for given sample values Using the above formula .

1. **Compute the function point value for a project with the following information domain characteristics:**
   (1) No. of user inputs = 24
   (2) No. of user outputs = 65
   (3) No. of user inquiries = 12
   (4) No. of files = 12
   (5) No. of external interfaces = 4
   Assume all complexity adjustment values are moderate and 14 algorithms have been counted.

   Solution:

Measurement Parameter	*Count*		*Weighing factor* *Simple Average Complex*		
1. Number of external inputs (EI)	24	*	4	=	96
2. Number of external outputs (EO)	65	*	5	=	325
3. Number of external inquiries (EQ)	12	*	4	=	48
4. Number of internal files (ILF)	12	*	10	=	120
5. Number of external interfaces (EIF)	4	*	7	=	28
Count-total ⟶					617

Now Fi for moderate case = 2.

So sum of all Fi (i ˝ 1 to 14) = 14 * 2 = 28

FP = Count-total * [0.65 + 0.01 * S (Fi)] = 617 * [0.65 + 0.01 * 28] = 617 * [0.65 + 0.28] =

617 * 1.23 = 758.91 = **759**

Outcome: All the students groups are actively participated in this activity and i gave few suggestions for those who are facing difficulty solving the above problem with that they understood the topic easily and completely.

Some other aids like

1. Seminar by students for specific topic - total 2 seminars conducted
   - During the 1st unit I asked the students to give seminar on changing nature of software and frame work activities
   - During the second unit I asked the students to give 1 seminar on various process models
2. NPTEL Lectures and other Video are also projected

[illegible]

(Faculty Incharge)

[illegible]

(CSE-HOD)

5) Interactive Learning : Role play

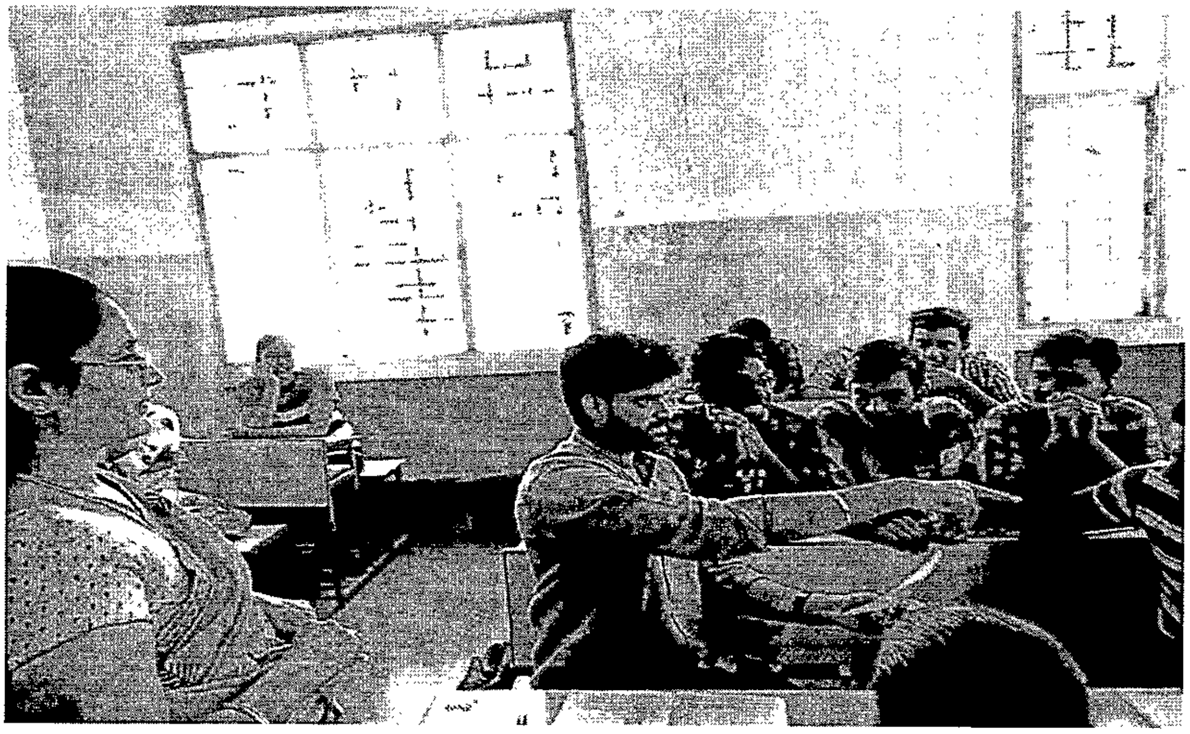
Subject : Software Engineering

Topic: RMMM plan preparation

Participants: Students of II/II D section

Implementation: Initially i have explained about RMMM and how RMMM plan has to prepared. As part of this activity i asked the students to prepare RMMM plan for any sample application and play various roles involved in preparing and keep tracking the same.

Outcome: All students actively participated and played different roles.in preparing the RMM plan. For those who have not followed i was explained..Student's interaction skills are improved along with subjective knowledge.



[illegible]

(Faculty Incharge)

[illegible]

(CSE-HOD)

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-500075.

Assessment Sheet – Co Wise (Direct Attainment)

Vidya Jyothi Institute of Technology

(An Autonomous Institution)

(Accredited by NAAC, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

Department of Computer Science & Engineering

(Accredited by NBA)

Academic Year: 2019-20 **BATCH: 2018-22**

II B.Tech- II Sem

Course: SE

Branch:CSE

S.No	Reg.No	MID I Threshold 60%							MID II Threshold 60%						Threshold
		ASM - I (5)	PART-A			PART-B			ASM - II (5)	PART-A	PART-B				
			Q1(2M)(CO1)	Q2(2M)(CO2)	Q3(2M)(CO3)	Q4(5M)(CO1)	Q5(5M)(CO2)	Q6(4M)(CO3)		Q1(6M)(CO3)	Q2(7M)(CO4)	Q3(7M)(CO4)	Q4(7M)(CO5)	Q5(7M)(CO5)	End Exam (75M)
1	17911A0568	5	1	1	1	2	2	1	5	4	7			7	7
2	17911A0593	5	2	2	2	5	5	3	5	5		7		7	45
3	18911A0501	5	0	0	0	0	0	0	5	0	0	0			49
4	18911A0502	5	1	1	1	3	3	2	5	4	7			7	52
5	18911A0503	5	0	0	0	0	0	0	5	1		5	5		47
6	18911A0504	5	1	1	2	5	5	2	5	2	5	5			61
7	18911A0505	5	1	1	1	5	5	1	5	2		6	6		52
8	18911A0506	5	1	1	1	5	5	1	5	1	7		7		48
9	18911A0507	5	2	2	2	5	5	3	5	2		7	7		58
10	18911A0508	5	2	2	2	5	5	2	5	5		7		7	18
11	18911A0509	5	2	2	2	5	5	3	5	5		7		7	51
12	18911A0510	5	2	2	2	5	5	3	5	5		7		7	45
13	18911A0511	5	0	0	0	0	0	0	5	0	0	0			49
14	18911A0512	5	1	1	1	3	3	2	5	4	7			7	10
15	18911A0513	5	0	0	0	0	0	0	5	1		5	5		52
16	18911A0515	5	1	1	2	5	5	2	5	2	5	5			49

17	18911A0516	5	1	1	1	5	5	1	5	2		6	6		2
18	18911A0518	5	1	1	1	5	5	1	5	1	7		7		51
19	18911A0519	5	2	2	2	5	5	3	5	2		7	7		52
20	18911A0521	5	2	2	2	5	5	2	5	5		7		7	49
21	18911A0522	5	2	2	2	5	5	3	5	5		7		7	60
22	18911A0523	5	2	2	2	5	5	3	5	5		7		7	45
23	18911A0524	5	0	0	0	0	0	0	5	0	0	0			58
24	18911A0525	5	1	1	1	3	3	2	5	4	7			7	54
25	18911A0527	5	0	0	0	0	0	0	5	1		5	5		50
26	18911A0528	5	1	1	2	5	5	2	5	2	5	5			49
27	18911A0529	5	2	2	2	5	5	2	5	5		7		7	53
28	18911A0530	5	2	2	2	5	5	3	5	5		7		7	47
29	18911A0531	5	2	2	2	5	5	3	5	5		7		7	15
30	18911A0532	5	0	0	0	0	0	0	5	0	0	0			49
31	18911A0533	5	1	1	1	3	3	2	5	4	7			7	56
32	18911A0534	5	0	0	0	0	0	0	5	1		5	5		56
33	18911A0535	5	1	1	2	5	5	2	5	2	5	5			45
34	18911A0536	5	1	1	1	3	3	2	5	4	7			7	45
35	18911A0537	5	0	0	0	0	0	0	5	1		5	5		60
36	18911A0538	5	1	1	2	5	5	2	5	2	5	5			8
37	18911A0539	5	1	1	1	5	5	1	5	2		6	6		54
38	18911A0540	5	1	1	1	5	5	1	5	1	7		7		49
39	18911A0541	5	2	2	2	5	5	3	5	2		7	7		55
40	18911A0542	5	2	2	2	5	5	2	5	5		7		7	49
41	18911A0543	5	2	2	2	5	5	3	5	5		7		7	53
42	18911A0544	5	2	2	2	5	5	3	5	5		7		7	65
43	18911A0546	5	0	0	0	0	0	0	5	0	0	0			52
44	18911A0547	5	1	1	1	3	3	2	5	4	7			7	59
45	18911A0548	5	0	0	0	0	0	0	5	1		5	5		48
46	18911A0549	5	1	1	2	5	5	2	5	2	5	5			54
47	18911A0550	5	1	1	1	5	5	1	5	2		6	6		49
48	18911A0551	5	1	1	1	5	5	1	5	1	7		7		52
49	18911A0552	5	2	2	2	5	5	3	5	2		7	7		60

50	18911A0553	5	2	2	2	5	5	2	5	5		7		7	50
51	18911A0554	5	2	2	2	5	5	3	5	5		7		7	49
52	18911A0555	5	2	2	2	5	5	3	5	5		7		7	45
53	18911A0556	5	0	0	0	0	0	0	5	0	0	0			52
54	18911A0557	5	1	1	1	3	3	2	5	4	7			7	51
55	18911A0559	5	0	0	0	0	0	0	5	1		5	5		49
56	18911A0560	5	1	1	2	5	5	2	5	2	5	5			56
57	18911A0561	5	1	1	1	5	5	1	5	2		6	6		66
58	18911A0562	5	1	1	1	5	5	1	5	1	7		7		14
59	18911A0563	5	2	2	2	5	5	3	5	2		7	7		54
60	18911A0564	5	2	2	2	5	5	2	5	5		7		7	10
61	18911A0565	5	2	2	2	5	5	3	5	5		7		7	49
62	18911A0566	5	2	2	2	5	5	3	5	5		7		7	18
63	18911A0567	5	0	0	0	0	0	0	5	0	0	0			60
64	18911A0568	5	1	1	1	3	3	2	5	4	7			7	58
65	18911A0569	5	0	0	0	0	0	0	5	1		5	5		45
66	18911A0570	5	1	1	2	5	5	2	5	2	5	5			34
67	18911A0571	5	1	1	1	5	5	1	5	2		6	6		36
68	18911A0572	5	1	1	1	5	5	1	5	1	7		7		47
69	18911A0573	5	2	2	2	5	5	3	5	2		7	7		38
70	18911A0574	5	2	2	2	5	5	2	5	5		7		7	26
71	18911A0575	5	2	2	2	5	5	3	5	5		7		7	31
72	18911A0576	5	2	2	2	5	5	3	5	5		7		7	38
73	18911A0577	5	0	0	0	0	0	0	5	0	0	0			27
74	18911A0578	5	1	1	1	3	3	2	5	4	7			7	33
75	18911A0579	5	0	0	0	0	0	0	5	1		5	5		30
76	18911A0580	5	1	1	2	5	5	2	5	2	5	5			34
77	18911A0581	5	1	1	1	5	5	1	5	2		6	6		26
78	18911A0582	5	1	1	1	5	5	1	5	1	7		7		31
79	18911A0583	5	2	2	2	5	5	3	5	2		7	7		33
80	18911A0584	5	2	2	2	5	5	2	5	5		7		7	26
81	18911A0585	5	2	2	2	5	5	3	5	5		7		7	12
82	18911A0586	5	2	2	2	5	5	4	5	6		7	7		24

83	18911A0587	5	2	2	2	5	5	3	5	5		7		7	16
84	18911A0588	5	1	1	1	3	3	1	5	2		7	7		28
85	18911A0589	5	2	2	2	5	5	3	5	5		7		7	30
86	18911A0590	5	1	1	2	5	5	2	5	1	7		7		29
87	18911A0591	5	2	2	2	5	5	2	5	6		7	7		26
88	18911A0592	5	1	1	1	5	5	1	5	3		7		7	26
89	18911A0593	5	2	2	2	5	5	4	5	3		7		7	29
90	18911A0594	5	1	1	1	5	5	1	5	4	7			7	26
91	18911A0595	5	1	1	1	3	3	2	5	3		7		7	29
92	18911A0596	5	1	1	1	4	4	2	5	2		7	7		36
93	18911A0597	5	1	1	1	2	2	2	5	3		7		7	26
94	18911A0598	5	1	1	2	5	5	2	5	2	4			4	32
95	18911A0599	5	1	1	2	5	5	2	5	2	5	5			28
96	18911A05A0	5	1	1	1	3	3	1	5	5		7		7	32
97	18911A05A1	5	2	2	2	5	5	1	5	4	7			7	29
98	18911A05A2	5	2	2	2	5	5	1	5	2	5	5			41
99	18911A05A3	5	1	1	1	3	3	1	5	2		6	5		15
100	18911A05A4	5	2	2	2	5	5	1	5	1		5	5		16
101	18911A05A5	5	2	2	2	5	5	1	5	4	7			7	28
102	18911A05A6	5	1	1	1	4	4	1	5	2	4			4	39
103	18911A05A7	5	1	1	2	5	5	2	5	2	4			4	35
104	18911A05A8	5	1	1	1	3	3	2	5	4	7			7	8
105	18911A05A9	5	1	1	1	4	4	2	5	2		7	7		31
106	18911A05B0	5	2	2	2	5	5	2	5	6		7	7		27
107	18911A05B1	5	1	1	2	5	5	2	5	6		7	7		37
108	18911A05B2	5	1	1	1	5	5	1	5	3		7		7	26
109	18911A05B3	5	1	1	1	3	3	2	5	5		7		7	32
110	18911A05B4	5	1	1	1	4	4	2	5	6		7	7		26
111	18911A05B6	5	2	2	2	5	5	1	5	6		7	7		30
112	18911A05B7	5	1	1	1	4	4	1	5	5		7		7	43
113	18911A05B8	5	2	2	2	5	5	2	5	0	0	0			33
114	18911A05B9	5	1	1	1	4	4	2	5	2		6	6		26
115	18911A05C1	5	1	1	1	5	5	1	5	6		7	7		34

116	18911A05C2	5	1	1	1	5	5	2	5	2		6	6		30
117	18911A05C3	5	1	1	1	5	5	1	5	5		7		7	39
118	18911A05C4	5	2	2	2	5	5	1	5	6		7	7		32
119	18911A05C5	5	1	1	1	5	5	2	5	4	7			7	28
120	18911A05C6	5	1	1	2	5	5	2	5	2		6	5		33
121	18911A05C7	5	1	1	1	2	2	2	5	2		6	5		28
122	18911A05C8	5	1	1	1	5	5	1	5	2		6	5		9
123	18911A05C9	5	2	2	2	5	5	2	5	6		7	7		40
124	18911A05D0	5	1	1	1	2	2	2	5	5		7		7	42
125	18911A05D1	5	2	2	2	5	5	1	5	1		5	5		29
126	18911A05D2	5	1	1	2	1	5	2	5	6		7	7		29
127	18911A05D4	5	1	1	1	1	5	2	5	3		7		7	26
128	18911A05D5	5	1	1	1	1	5	1	5	2		6	6		26
129	18911A05D7	5	1	1	1	2	5	1	5	4	7			7	26
130	18911A05D8	5	2	2	2	2	5	1	5	6		7	7		39
131	18911A05D9	5	1	1	2	2	5	2	5	4	7			7	36
132	18911A05E0	5	1	1	2	2	5	2	5	2		2	2		4
133	18911A05E1	5	1	1	2	2	5	2	5	5		7		7	28
134	18911A05E2	5	2	2	2	1	5	1	5	4	7			7	27
135	18911A05E3	5	1	1	1	2	3	2	5	1	7		7		26
136	18911A05E4	5	2	2	2	2	5	4	5	6		7	7		45
137	18911A05E5	5	2	2	2	2	5	2	5	6		7	7		14
138	18911A05E6	5	2	2	2	1	5	4	5	4	7			7	54
139	18911A05E7	5	1	1	1	1	5	2	5	0		0		0	45
140	18911A05E8	5	1	1	1	1	4	2	5	2	5	5			14
141	18911A05E9	5	1	1	1	2	2	2	5	2	5	5			60
142	18911A05F0	5	2	2	2	5	5	1	5	6		7	7		49
143	18911A05F1	5	1	1	1	4	4	2	5	6		7	7		13
144	18911A05F2	5	1	1	1	1	5	2	5	6		7		7	54
145	18911A05F3	5	2	2	2	1	5	1	5	6		7		7	12
146	18911A05F4	5	1	1	2	1	5	2	5	6		7		7	46
147	18911A05F5	5	2	2	2	2	5	1	5	6		7	7		52
148	18911A05F6	5	1	1	1	2	4	2	5	6	7			7	16

149	18911A05F7	5	2	2	2	2	5	1	5	6	0		0		48
150	18911A05F8	5	2	2	2	2	5	4	5	6		7		7	54
151	18911A05F9	5	2	2	2	2	5	1	5	6	7			7	45
152	18911A05G0	5	2	2	2	1	5	3	5	6		7	7		52
153	18911A05G1	5	1	1	2	2	5	2	5	6	7			7	49
154	18911A05G2	5	2	2	2	2	5	1	5	6		7		7	54
155	18911A05G3	5	1	1	1	2	4	1	5	6		7		7	49
156	18911A05G4	5	2	2	2	1	5	2	5	6		7	7		48
157	18911A05G5	5	2	2	2	1	5	3	5	6		6	6		52
158	18911A05G6	5	2	2	2	1	5	3	5	6	7			7	22
159	18911A05G7	5	1	1	1	1	1	0	5	6		4	4		48
160	18911A05G8	5	1	1	1	4	4	1	5	6	5	5			51
161	18911A05G9	5	2	2	2	5	5	3	5	6		7		7	47
162	18911A05H0	5	2	2	2	5	5	4	5	6		7	7		59
163	18911A05H1	5	2	2	2	5	5	4	5	6		7		7	54
164	18911A05H2	5	2	2	2	5	5	1	5	6		7	7		52
165	18911A05H3	5	2	2	2	5	5	3	5	6		7		7	49
166	18911A05H5	5	1	1	1	2	2	1	5	6		7	7		54
167	18911A05H6	5	2	2	2	5	5	4	5	6		7	7		55
168	18911A05H7	5	2	2	2	5	5	3	5	6	7			7	49
169	18911A05H8	5	1	1	2	5	5	2	5	6		6	5		14
170	18911A05H9	5	1	1	1	1	5	1	5	6	7			7	52
171	18911A05J0	5	1	1	1	1	3	2	5	6		5	5		52
172	18911A05J1	5	1	1	1	1	5	2	5	6	7			7	45
173	18911A05J2	5	1	1	1	2	3	2	5	6		7		7	66
174	18911A05J3	5	1	1	1	2	5	1	5	6		7		7	4
175	18911A05J4	5	2	2	2	2	5	3	5	6	7			7	49
176	18911A05J5	5	2	2	2	2	5	4	5	6		6	5		46
177	18911A05J6	5	2	2	2	2	5	2	5	6		7		7	57
178	18911A05J7	5	2	2	2	1	5	2	5	6		7	7		10
179	18911A05J8	5	1	1	1	2	5	1	5	6		5	5		45
180	18911A05J9	5	2	2	2	2	5	2	5	6	7			7	54
181	18911A05K0	5	1	1	1	2	4	2	5	6	4			4	59

182	18911A05K1	5	2	2	2	1	5	3	5	6		7		7	51
183	18911A05K2	5	2	2	2	1	5	4	5	6		7		7	53
184	18911A05K4	5	1	1	2	1	5	2	5	6		7	7		49
185	18911A05K5	5	2	2	2	5	5	4	5	6	7			7	52
186	18911A05K6	5	2	2	2	5	5	4	5	6		7		7	45
187	18911A05K7	5	1	1	1	1	1	0	5	6		6	6		54
188	18911A05K8	5	1	1	1	4	4	1	5	6	3		3		52
189	18911A05K9	5	2	2	2	5	5	4	5	6	7			7	49
190	18911A05L0	5	2	2	2	5	5	4	5	6		7		7	55
191	18911A05L1	5	1	1	1	5	5	1	5	6	7			7	45
192	18911A05L2	5	2	2	2	5	5	1	5	6		7	7		54
193	18911A05L3	5	1	1	1	5	5	2	5	6		7		7	49
194	18911A05L4	5	1	1	1	3	3	2	5	6		7	7		54
195	18911A05L5	5	2	2	2	5	5	1	5	6		6	6		49
196	18911A05L7	5	1	1	1	5	5	2	5	6		5	5		10
197	18911A05L8	5	1	1	1	5	5	1	5	6	5	5			10
198	18911A05L9	5	2	2	2	5	5	3	5	6		7	7		66
199	18911A05M0	5	1	1	1	5	5	2	5	6	7			7	48
200	18911A05M1	5	1	1	1	5	5	1	5	6		6	5		54
201	18911A05M2	5	2	2	2	5	5	3	5	6		7		7	45
202	18911A05M3	5	2	2	2	5	5	1	5	6		6	6		60
203	18911A05M4	5	1	1	1	2	2	2	5	6	3		3		49
204	18911A05M5	5	2	2	2	5	5	1	5	6	7			7	54
205	18911A05M6	5	2	2	2	5	5	1	5	6	7			7	A
206	18911A05M7	5	1	1	2	5	5	2	5	6		6	6		47
207	18911A05M8	5	2	2	2	5	5	2	5	6	7		7		50
208	18911A05M9	5	2	2	2	5	5	3	5	6	7			7	49
209	18911A05N0	5	1	1	1	5	5	2	5	6	7			7	52
210	18911A05N1	5	2	2	2	5	5	4	5	6		7	7		53
211	18911A05N2	5	2	2	2	5	5	4	5	6	7			7	60
212	18911A05N3	5	1	1	2	5	5	4	5	6	5	5			48
213	18911A05N5	5	1	1	1	3	3	4	5	6	4			4	45
214	18911A05N6	5	1	1	1	3	3	4	5	6		6	5		52

215	18911A05N7	5	1	1	1	5	5	4	5	6		7	7		49
216	18911A05N8	5	1	1	2	5	5	4	5	6		7	7		52
217	18911A05N9	5	1	1	2	5	5	4	5	6		7	7		58
218	18911A05P0	5	1	1	1	4	4	4	5	6	7			7	46
219	18911A05P1	5	2	2	2	5	5	4	5	6	7			7	18
220	18911A05P2	5	1	1	1	4	4	4	5	6		6	5		55
221	18911A05P3	5	1	1	1	3	3	4	5	6	7			7	54
222	18911A05P4	5	1	1	1	2	2	4	5	6		6	5		60
223	18911A05P5	5	1	1	1	4	4	4	5	6		7	7		53
224	18911A05P6	5	2	2	2	5	5	4	5	6	7			7	49
225	18911A05P7	5	1	1	1	5	5	4	5	6	7			7	58
226	18911A05P8	5	1	1	1	5	5	4	5	6	7			7	14
227	18911A05P9	5	2	2	2	5	5	4	5	6	7			7	2
228	18911A05Q0	5	2	2	2	5	5	4	5	6	7			7	49
229	19915A0501	5	1	1	1	5	5	4	5	6	7		7		60
230	19915A0502	5	1	1	1	4	4	4	5	6	7			7	48
231	19915A0503	5	2	2	2	5	5	4	5	6	7			7	45
232	19915A0504	5	2	2	2	5	5	4	5	6		6	6		52
233	19915A0505	5	1	1	2	5	5	4	5	6	7			7	55
234	19915A0506	5	1	1	1	0	0	4	5	6		3	3		50
235	19915A0507	5	1	1	1	3	3	4	5	6		6	5		45
236	19915A0508	5	1	1	1	4	4	4	5	6		6	6		60
237	19915A0509	5	2	2	2	5	5	4	5	6		7		7	61
238	19915A0510	5	1	1	1	5	5	4	5	6		2	2		46
239	19915A0511	5	1	1	1	3	3	4	5	6	4			4	58
240	19915A0512	5	2	2	2	5	5	4	5	6	7			7	55
241	19915A0513	5	2	2	2	5	5	4	5	6		7		7	54
242	19915A0514	5	1	1	1	5	5	4	5	6		4	4		49
243	19915A0515	5	2	2	2	5	5	4	5	6		7	7		52
244	19915A0516	5	1	1	1	5	5	4	5	6		7	7		49
245	19915A0517	5	1	1	1	4	4	4	5	6	3		3		14
246	19915A0518	5	1	1	2	5	5	4	5	6		7		7	52
247	19915A0519	5	2	2	2	5	5	4	5	6	7		7		62

248	19915A0520	5	1	1	1	4	4	4	5	6		7		7	62
249	19915A0521	5	1	1	2	5	5	4	5	6	7			7	49
250	19915A0522	5	1	1	1	5	5	4	5	6		7		7	54
251	19915A0523	5	2	2	2	5	5	4	5	6	7			7	8
252	19915A0524	5	2	2	2	5	5	4	5	6	7			7	52
Average marks		5	1.345238095	1.345238095	1.464285714	3.666666667	4.246031746	2.21031746	5	4.365079365	5.666666667	6	6.009345794	6.791666667	41.68924
No of students attemped		252	252	252	252	252	252	252	252	252	99	178	107	120	251
%of students scored 60% and		100	93.25396825	93.25396825	93.25396825	71.03174603	88.88888889	69.04761905	100	71.82539683	89.8989899	93.25842697	97.19626168	99.16666667	63.75
CO ATTAINMENT LEVEL		3	3.0	3.0	3.0	3.0	3.0	2.0	3	3.0	3.0	3.0	3.0	3.0	2.0

ASSESSMENT OF COs FOR THE COURSE

CO	Method			value		CO Attainment (Internal)		CO Attainment (End Exam)	Overall CO Attainment
CO 1	ASM I			3	3.00	2.96		2.00	2.24
	MID I - PART A-Q1			3.0					
	MID I - PART B-Q4			3.0					
CO 2	ASM I			3	3.00				
	MID I - PART A-Q2			3.0					
	MID I - PART B-Q5			3.0					
CO 3	ASM I			3	2.8				
	ASM II			3.0					
	MID I - PART A-Q3			3.0					
	MID I - PART B-Q6			2.0					
	MID II - PART A-Q1			3.0					
CO 4	ASM II			3	3				
	MID II - PART B-Q2			3.0					
	MID II - PART B-Q3			3.0					
CO 5	ASM II			3	3				
	MID II- PART B-Q4			3.0					
	MID II - PART B-Q5			3.0					

[illegible]
COURSE-COORDINATOR

HOD CSE
[illegible]
Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-50075.

Vidya Jyothi Institute of Technology

An Autonomous Institution

(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Course End Survey Form

Name of the student		**Year &sem**	**II – II**
Roll number		**Regulations**	**R 15**

Dear Student,

We need your help in evaluating the courses offered, by responding the short survey below.

Your feedback is very valuable for us in order to continually improve our program. The aim of this survey is to evaluate how well each of the courses has prepared you to have necessary skills.

Your responses will be kept confidential and will not be revealed to anyone outside the department without your permission.

Please indicate (√) the level to which you agree with the following criterion:

(3: Strongly agree 2: Agree 1: Strongly disagree)

Name of The Course: SOFTWARE ENGINEERING		**RATING**		
After completing this course the student must demonstrate the knowledge and ability to		3	2	1
CO 1	Choose a process model to apply for given project requirements			
CO 2	Analyze and apply the framework activities for a given project			
CO 3	Design various system models for a given scenario			
CO 4	Design and apply various testing techniques			
CO 5	Understand metrics for Process and Products			

Any other comments / suggestions: ________________________________

____

____

Signature

Course End Survey Form