ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MOODBIDRI - 574 225

Affiliated to VTU, Belgaum and Approved by A.I.C.T.E., New Delhi

COURSE BOOK

(ACD - 08, ACD - 09, ACD - 10, ACD - 12, ACD - 13)

Period of the Semester (Odd / Even) : From 2017 to .....

Semester : 8th 'B' 10IS81

Subject with Code : Software Architectures

TIME SLOT

Mon :	9:00 - 9:55 am	Tue :	9:00 - 9:55 am
Wed :	9:55 - 10:50 am	Thu :	12:05 - 1:00 am
Fri. :	9:00 - 9:55 am	Sat :	—

Name of the Teacher : Vineetha Pais

Department : CSE

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY

(A Unit of Alva's Education Foundation)
Shobhavana Campus, Mijar-574225, Moodbidri, D.K
Phone: 08258-262725, Fax: 08258-262726

CALENDAR OF EVENTS (EVEN SEMESTER - 2017) BE, M.TECH & MBA

Week	Month	Days							# of working Days	Activities
		Mon	Tue	Wed	Thu	Fri	Sat	Sun		
1	JAN	30	31						02	
2	FEB			1	2	3	4	5	04	4th : Commencement of Sports for staff/students
3		6	7	8	9	10	11	12	05	11th : Sports Day
4		13	14	15	16	17	18	19	06	15th : Commencement of Literary, Fine Arts and cultural activates
5		20	21	22	23	24	25	26	05	24th: Shivaratri
6		27	28						02	27th, 28th :Project Evaluation – Phase-II
6	MARCH			1	2	3	4	5	04	
7		6	7	8	9	10	11	12	06	
8		13	14	15	16	17	18	19	06	
9		20	21	22	23	24	25	26	06	23th, 24th, 25th : I-IA Test
10		27	28	29	30	31			04	29th: Ugadi 30th :Submission of Activities Results 31th :Project Evaluation–Phase-III
10	APRIL						1	2	01	1st :Project Evaluation–Phase-III
11		3	4	5	6	7	8	9	05	7th : Traditional Day
12		10	11	12	13	14	15	16	05	14th: Good Friday
13		17	18	19	20	21	22	23	06	
14		24	25	26	27	28	29	30	06	27th, 28th, 29th : II-IA Test
15	MAY	1	2	3	4	5	6	7	05	1st : May Day 4th : Last date for Project Report Submission 5th : Talents Day, 6th : College Day
16		8	9	10	11	12	13	14	06	13th : Final Year Project Exhibition
17		15	16	17	18	19	20	21	06	
18		22	23	24	25	26	27	28	06	22nd,23rd, 24th : III-IA Test
19		29	30	31					03	

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY, MIJAR, MOODBIDRI

INDIVIDUAL FACULTY TIME TABLE		FACULTY NAME:	Vineetha Pais (VP)					
Department	Computer Science and Engg.			Academic Year	2017-18 (EVEN SEMESTER)			
Period→	1	2	TEA	3	4		5	6	7
Time →	09.00 – 09 55	09.55 – 10 50		11.10 – 12.05	12.05 – 01.00		02.00 – 03.00	03.00 – 04.00	04.00 – 05.00
MONDAY	SA (B)		BREAK			BREAK	← DAA LAB IV(A) →		
TUESDAY	SA (B)						← MP LAB IV (B) →		
WEDNESDAY		SA (B)					← SEMINAR (B) →		
THURSDAY					SA (B)		← DAA LAB IV(A) →		
FRIDAY	SA (B)			← MLT →			← MP LAB IV (B) →		
SATURDAY	← MLT →								

Other Special Activities: EMS VIII(B), ROSTRUM ,MINI PROJECT (VI A)

UNITS:	Theory: 18	LAB: 12	Others: 02	TOTAL UNITS:32

Time Table Coordinator

25.01.2017

25/1/18

H.O.D.

Dept. Of Computer Science & Engineering
Alva's Institute of Engg. & Technology
Mijar, MOODBIDRI - 574 225

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY, MIJAR, MOODBIDRI										
INDIVIDUAL FACULTY TIME TABLE				FACULTY NAME:			Sushanth M (SM)			
Department		Computer Science and Engg.				Academic Year		2017-18 (EVEN SEMESTER)		
Period→	1	2	TEA	3	4		5	6	7	
Time →	09.00 – 09 55	09.55 – 10 50		11.10 – 12.05	12.05 – 01.00		02.00 – 03.00	03.00 – 04.00	04.00 – 05.00	
MONDAY	SA (A)							← COACHING →		
TUESDAY				SA (A)		BREAK	← USP LAB VI(A) →			
WEDNESDAY	DC (B)		BREAK	SA (A)						
THURSDAY	DC (B)			SA (A)			← USP LAB VI(A) →			
FRIDAY		DC (B)		SA (A)			← PROJECT WORK VIII (A) →			
SATURDAY				DC (B)						

Other Special Activities: HOSTEL/ANTI RAGING, EMS COORDINATOR (CHIEF DEPT)				
UNITS:	Theory: 22	LAB: 06	Others:02	TOTAL UNITS:30

Time Table Coordinator
25.01.2017

H.O.D. 25/1/17

Dept. Of Computer Science & Engineering
Alva's Institute of Engg. & Technology
Mijar, MOODBIDRI - 574 225

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY, MIJAR, MOODBIDRI

CLASS TIMETABLE

Format No.	ACD06
Issue No.	01
Rev. No.	00

Department	Computer Science and Engg.	Semester	VIII Sem B.E.	Section	A
Academic Year	2016-17	Room No.	316		
Class Coordinator	Ms. Remul Pinto	EMS Coordinator	Mr. Hemanth Kumar N P		

Day / Period→ / Time →	1 9:00 – 9:55	2 9:55 – 10:50	TEA BREAK	3 11:10 – 12:05	4 12:05 – 1:00	LUNCH BREAK	5 2:00 – 3:00	6 3:00 – 4:00	7 4:00 – 5:00
MONDAY	SA	INS		SMS	CGC/ST		SEMINAR [HK+CA+RK+PSK+VD+DM+TNHB]		
TUESDAY	SMS	INS		SA	CGC/ST		SEMINAR [HK+CA+RP+MBS+GRG]		
WEDNESDAY	INS	SMS		SA	CGC/ST		PROJECT WORK [CA+HNP]		
THURSDAY	CGC/ST	INS		SA	SMS		PROJECT WORK [CA+HGM+SP]		
FRIDAY	SMS	INS		SA	CGC/ST		PROJECT WORK [HK+SM+AN]		
SATURDAY	PROJECT WORK [DM+AS]			PROJECT WORK [DM+AS]					

Subject Code	Subject Title	Subject Initial	Faculty Name	Faculty Code
10IS81	Software Architectures	SA	Mr. Sushanth M	SM
10CS82	System Modeling and Simulation	SMS	Ms. Remul Pinto	RP
10CS835	Information and Network Security	INS	Mr. Sayeesh	SS
10CS842	Software Testing (Room No. 315)	ST	Mrs. Vidya	VD
10CS845	Cloud & Grid and Clusters	CGC	Mr. Chanchal Antony	CA
10CS85	Project Work	----------	Mr. Chanchal Antony Mr. Harish Kunder Ms. Deeksha M Ms. Ankitha Shetty Mr. Hemanth Kumar N P Mrs. Harshitha G M Mr. Shilpa Mr. Sushanth M Mr. Parikshith Nayaka S K	CA HK DM AS HNP HGM SP SM PSK
10CS86	Seminar	----------	Mr. Harish Kunder Prof. Golsangi G R Dr. S. Mohideen Badhusha Mr. Chanchal Antony Mr. Tahir N H B Ms. Remul Pinto Mr. Parikshith Nayaka S K Mr. Rithesh Kumar Mrs. Vidya Ms.Deeksha M	HK GRG MBS CA TNHB RP PSK RK VD DM

Time Table Coordinator
25.01.2017

HOD
25/1/17
Dept. Of Computer Science & Engineering
Alva's Institute of Engg. & Technology
Mijar, MOODBIDRI - 574 225

Principal
PRINCIPAL
Alva's Institute of Engg. & Technology
Mijar, MOODBIDRI - 574 225, D.K

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY, MIJAR, MOODBIDRI

CLASS TIMETABLE	Format No.	ACD06
	Issue No.	01
	Rev. No.	00

Department	Computer Science and Engg.	Semester	VIII Sem B.E.	Section	B
Academic Year	2016-17	Room No.	102		
Class Coordinator	Mr. Rithesh Kumar	EMS Coordinator	Mrs. Vineetha Pais		

Day	Period→	1	2	TEA BREAK	3	4	LUNCH BREAK	5	6	7
	Time →	9:00 – 9:55	9:55 – 10:50		11:10 – 12:05	12:05 – 1:00		2:00 – 3:00	3:00 – 4:00	4:00 – 5:00
MONDAY		SA	INS		SMS	CGC/ST		PROJECT WORK [SS+MK]		
TUESDAY		SA	INS		SMS	CGC/ST		PROJECT WORK [SS+PSK]		
WEDNESDAY		INS	SA		SMS	CGC/ST		SEMINAR [HK+MBS+VS+DM+VP]		
THURSDAY		CGC/ST	INS		SMS	SA		SEMINAR [HK+ GRG+RP+RK+SS]		
FRIDAY		SA	INS		SMS	CGC/ST		PROJECT WORK [SS+MK]		
SATURDAY		PROJECT WORK [MM+SP]			PROJECT WORK [MM+SP]					

Subject Code	Subject Title	Subject Initial	Faculty Name	Faculty Code
10IS81	Software Architectures	SA	Mrs. Vineetha Pais	VP
10CS82	System Modeling and Simulation	SMS	Mr. Harish Kunder	HK
10CS835	Information and Network Security	INS	Prof. A Manjunath Kotari	AMK
10CS842	Software Testing (Room No. 315)	ST	Mrs. Vidya	VD
10CS845	Cloud & Grid and Clusters	CGC	Mr. Rithesh Kumar	RK
10CS85	Project Work	----------	Mr. Hemanth Kumar N P Mr. Sayeesh Mrs. Merlyn Melita Mathias Ms. Shilpa Ms.Megha D Hegde Ms. Mangala Kini Mr. Parikshith Nayak S K	HNP SS MM SP MDH MK PSK
10CS86	Seminar	----------	Mr. Harish Kunder Prof. Golsangi G R Dr. S. Mohideen Badhusha Mr.Vasudev S Shahapur Ms.Deeksha M Mrs. Vineetha Pais Ms. Remul Pinto Mr. Rithesh Kumar Mr. Sayeesh	HK GRG MBS VS DM VP RP RK SS

Time Table Coordinator
25.01.2017

H.O.D.
HOD
Dept. Of Computer Science & Engineering
Alva's Institute of Engg. & Technology
Mijar, MOODBIDRI - 574 225

Principal
PRINCIPAL
Alva's Institute of Engg. & Technology

CSE-VIIISem -Section A		
SL NO.	USN	NAME OF THE STUDENTS
1.	4AL12CS003	AISHWARYA RAMESH
2.	4AL12CS009	ANOOP NAIK
3.	4AL12CS010	ANUSHA AJITH
4.	4AL12CS036	GIREESH MON .P.M
5.	4AL12CS048	MANISHA K.P
6.	4AL12CS062	PAYAL M.P SHETTY
7.	4AL13CS001	ADARSH AJITH
8.	4AL13CS002	ADHIEL HUSSAIN
9.	4AL13CS003	AISHWARYA K SHETTY
10.	4AL13CS004	AKASH G
11.	4AL13CS005	AKSHATHA BHAT
12.	4AL13CS007	AMINA SAJIAHA
13.	4AL13CS008	ANIKETAN NOEL
14.	4AL13CS010	ANUSHA
15.	4AL13CS011	ANUSHA S ARADHYA
16.	4AL13CS012	ANUSHA V
17.	4AL13CS013	ANVITHA SHUBHAVEER JAIN
18.	4AL13CS015	ARPITA S BHAT
19.	4AL13CS016	ARYA K M
20.	4AL13CS018	ASIF
21.	4AL13CS019	B ASRITHA
22.	4AL13CS020	BHANDARY SHREYAS SHANKAR
23.	4AL13CS021	BHARATH R
24.	4AL13CS022	BINDU S
25.	4AL13CS024	DARSHSANA K
26.	4AL13CS025	DEEKSHA SHETTY B
27.	4AL13CS026	DEEKSHA SHETTY D
28.	4AL13CS027	DEENA DEEPIKA CUTINHA
29.	4AL13CS028	DEEPAK D K
30.	4AL13CS029	DEEPIKA B SHETTY
31.	4AL13CS030	DEVIPRASAD S SHETTY
32.	4AL13CS031	DILEEP KUMAR
33.	4AL13CS032	DRISYA S BABU
34.	4AL13CS034	GAURAV SUVARNA
35.	4AL13CS035	GOUTAMI PRAKASH HOSUR
36.	4AL13CS036	GOWRAVA
37.	4AL13CS037	GURURAJ HIPPARAGI
38.	4AL13CS038	HASMITHA B V
39.	4AL13CS039	HELLEN TREESA JOHN
40.	4AL13CS040	HITESH HARIKRISHNAN
41.	4AL13CS041	IMPANA M S
42.	4AL13CS043	JERIN JOSE
43.	4AL13CS044	JISHA EMMANUEL
44.	4AL13CS045	JISNA JAYARAJ
45.	4AL13CS047	KAVYA
46.	4AL13CS048	KUSUMA J
47.	4AL13CS049	LATHISHMA SUVARNA M
48.	4AL13CS050	LAVANYA R
49.	4AL13CS051	M RAKSHA KUDVA
50.	4AL13CS052	M SANGEETHA JAIN
51.	4AL13CS053	MAHESHAKUMAR SHETTY
52.	4AL13CS054	MANOJ E

Dept. Of Computer Science & Engineering
Alva's Institute of Engg. & Technology
Mijar, MOODBIDRI - 574 225

CSE-VIII Sem -Section B		
SL No.	**USN**	**Name of the Students**
1.	4AL13CS055	MEGHANA V
2.	4AL13CS056	NALINA L N
3.	4AL13CS057	NAVAMI R S
4.	4AL13CS058	NAVYASHREE
5.	4AL13CS059	NIRIKSHA M
6.	4AL13CS062	PAVAN KUMAR S
7.	4AL13CS063	PAVITHRA G
8.	4AL13CS065	PRAJWAL M SHETTY
9.	4AL13CS066	PRARTHANA G M
10.	4AL13CS069	RAJATHA MITHYANTHA
11.	4AL13CS070	RAKSHITHA K SHETTY
12.	4AL13CS071	RAKSHITHA SUVARNA B
13.	4AL13CS072	RANJITHA M
14.	4AL13CS073	RANJITHA NAIK K
15.	4AL13CS075	RENITA D SOUZA
16.	4AL13CS076	S J SHREYAS
17.	4AL13CS078	SAMRUDDHI N R
18.	4AL13CS079	SAMSON IMMANUEL K
19.	4AL13CS080	SANJANA DEVADIGA
20.	4AL13CS081	SENVA DEEKSHA DIWAKAR
21.	4AL13CS083	SHASHIKANTH WAGLE
22.	4AL13CS084	SHEHAAB AHAMAD A
23.	4AL13CS085	SHETTY HARSHITA PURUSHOTTAM
24.	4AL13CS086	SHETTY KAVYA DIVAKAR
25.	4AL13CS087	SHETTY NITHESH SHRIDHAR
26.	4AL13CS089	SHETTY SHREYAS UMESH
27.	4AL13CS092	SHREELAKSHMI D T
28.	4AL13CS093	SHRUTHI SURENDRAN
29.	4AL13CS094	SHWETA B KURIYAVAR
30.	4AL13CS096	SONIYA GEORGE
31.	4AL13CS097	SOORYA K
32.	4AL13CS098	SRIDEVI
33.	4AL13CS099	SRIDHAR T S
34.	4AL13CS100	SRUTHIN R
35.	4AL13CS103	SUKANYA
36.	4AL13CS104	SUMALATHA N
37.	4AL13CS106	SUPRITHA KN
38.	4AL13CS107	SWETHA T P
39.	4AL13CS108	TIBIN K TOMY
40.	4AL13CS110	VANDANA G
41.	4AL13CS117	ROHIT RAJAN BABU
42.	4AL13CS407	VIDITHA KARMARKAR
43.	4AL14CS400	KARTHIK C SHEKAR
44.	4AL14CS404	SHETTY ROHAN SURESH

H.O.D.
Dept. Of Computer Science & Engineering
Alva's Institute of Engineering & Technology
Mijar, MOODBIDRI - 574 225

SOFTWARE ARCHITECTURES

Subject Code: 10IS81

I.A. Marks : 25

Hours/Week : 04 **Exam Hours: 03**

Total Hours : 52 **Exam Marks: 100**

PART – A

UNIT – 1
6 Hours
Introduction: The Architecture Business Cycle: Where do architectures come from? Software processes and the architecture business cycle; What makes a "good" architecture? What software architecture is and what it is not; Other points of view; Architectural patterns, reference models and reference architectures; Importance of software architecture; Architectural structures and views.

UNIT – 2
7 Hours
Architectural Styles and Case Studies: Architectural styles; Pipes and filters; Data abstraction and object-oriented organization; Event-based, implicit invocation; Layered systems; Repositories; Interpreters; Process control; Other familiar architectures; Heterogeneous architectures. Case Studies: Keyword in Context; Instrumentation software; Mobile robotics; Cruise control; Three vignettes in mixed style.

UNIT – 3
6 Hours
Quality: Functionality and architecture; Architecture and quality attributes; System quality attributes; Quality attribute scenarios in practice; Other system quality attributes; Business qualities; Architecture qualities. Achieving Quality: Introducing tactics; Availability tactics; Modifiability tactics; Performance tactics; Security tactics; Testability tactics; Usability tactics; Relationship of tactics to architectural patterns; Architectural patterns
and styles.

UNIT – 4
7 Hours
Architectural Patterns – 1: Introduction; From mud to structure: Layers, Pipes and Filters, Blackboard.

PART – B

UNIT – 5
7 Hours
Architectural Patterns – 2: Distributed Systems: Broker; Interactive Systems: MVC, Presentation-Abstraction-Control.

UNIT – 6
6 Hours
Architectural Patterns – 3: Adaptable Systems: Microkernel; Reflection.

UNIT – 7
6 Hours
Some Design Patterns: Structural decomposition: Whole – Part; Organization of work: Master – Slave; Access Control: Proxy.

UNIT – 8
7 Hours
Designing and Documenting Software Architecture: Architecture in the life cycle; Designing the architecture; Forming the team structure; Creating a skeletal system. Uses of architectural documentation; Views; Choosing the
relevant views; Documenting a view; Documentation across views.

Text Books:

1. Len Bass, Paul Clements, Rick Kazman: Software Architecture in Practice, 2nd Edition, Pearson Education, 2003.
(Chapters 1, 2, 4, 5, 7, 9)
2. Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, Michael Stal: Pattern-Oriented Software Architecture, A System of Patterns, Volume 1, John Wiley and Sons, 2007. (Chapters 2, 3.1 to 3.4)
3. Mary Shaw and David Garlan: Software Architecture- Perspectives on an Emerging Discipline, PHI, 2007.
(Chapters 1.1, 2, 3)

Reference Books:

1. E. Gamma, R. Helm, R. Johnson, J. Vlissides: Design Patterns- Elements of Reusable Object-Oriented Software, Pearson Education, 1995.

Web Reference: http://www.hillside.net/patterns/

USN | | | | | | | | | | |

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY, MOODBIDRI

(A unit of Alva's Education Foundation)

Eighth Semester B.E. (CSE) **I. A Test Examinations-I**

23rd MAR 2017

Subject CODE :10IS81 **SUBJECT NAME: SOFTWARE ARCHITECTURES**

Duration: 1 ½ Hour Max. Marks: 50

*Note: 1) Answer **Two Full** questions from each **Part.***

PART-A

1. a) Why is Software Architecture important? Explain in brief. 6 M
   b) Explain in brief about Pipes and Filters style with neat diagram and also list out advantages and disadvantages of the same. 6½ M

2. a) How the following architecture styles gives solution for Keyword in Context case study and also explain problems of each
      i. Main program/Subroutine with shared data
      ii. Pipes and Filters solution 6 M
   b) Explain with a neat diagram how the Architecture Business Cycle (ABC) works. 6½ M

3. a) Briefly explain what different influences to and from architecture are. 6 M
   b) What is architectural pattern, reference model, reference architecture? 6½ M

PART B

4. a) How the following architecture styles gives solution for Mobile Robotics case study and also explain requirements of each
      i. Layered architecture solution.
      ii. Implicit invocation solution. 6 M
   b) Explain the Layered systems architecture style with a neat diagram and also list the advantages, disadvantages of the same. 6½ M

5. a) Explain in brief about Instrumentation software
      i. With Object Oriented model solution.
      ii. With a Layered model solution. 6 M
   b) Explain security general scenario with a neat diagram. 6½ M

6. a) Write a note on system quality attributes. 6 M
   b) Explain Availability general scenario with a neat diagram. 6½ M

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY, MOODBIDRI

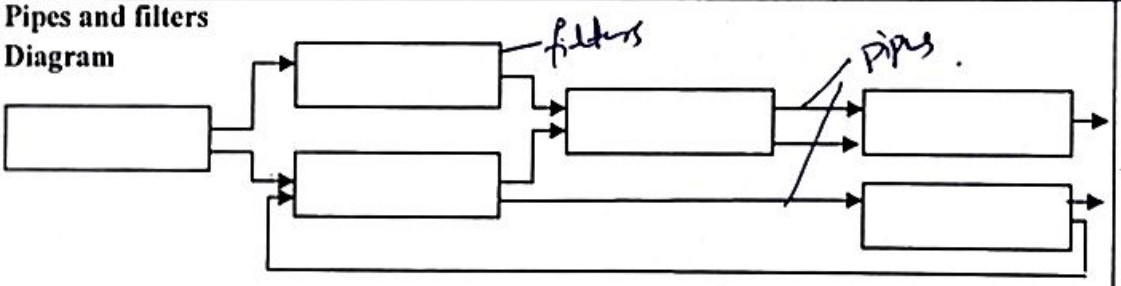
(A unit of Alva's Education Foundation)

Eighth Semester B.E. (CSE) **I. A Test Examinations-I**

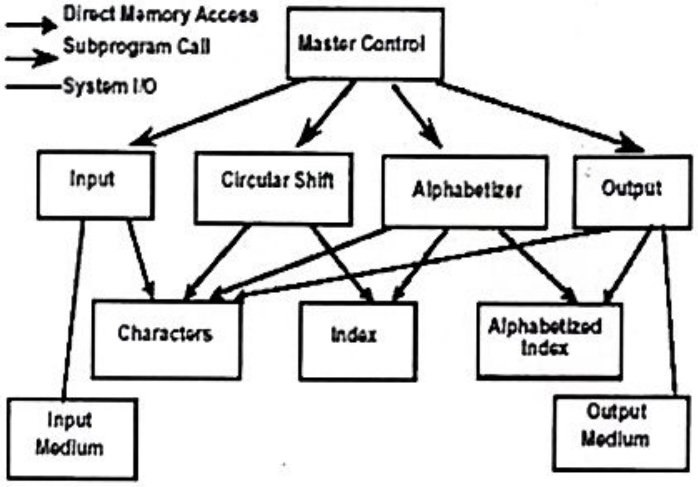
SCEHEME OF EVALUATION

SUB CODE: 10IS81 **SOFTWARE ARCHITECTURE**

Note: Answer any Two full questions from each part.

1.	a)	• **Architecture is a vehicle for stakeholder communication** • **Architecture manifests the earliest set of design decisions** ➢ The architecture defines constraints on implementation ➢ The architecture dictates organizational structure ➢ The architecture inhibits or enables a systems quality attributes ➢ Predicting system qualities by studying the architecture ➢ The architecture makes it easier to reason about and manage change ➢ The architecture helps in evolutionary prototyping ➢ The architecture enables more accurate cost and schedule estimates • **Architecture is a transferable, reusable model** ➢ Software product lines share a common architecture ➢ Systems can be built using large, externally developed elements ➢ Less is more: It pays to restrict the vocabulary of design alternatives ➢ An architecture permits template based development ➢ An architecture can be the basis for training	6M
1	b)	**Pipes and filters** **Diagram**  Pipe-and-filter systems have a number of nice properties. First, they allow the designer to understand the overall input/output behavior of a system as a simple composition of the behaviors of the individual filters. Second, they support reuse: any two filters can be hooked together, provided they agree on the data that are being transmitted between them. Third, systems are easy to maintain and enhance: new filters can be added to existing systems and old filters can be replaced by improved ones. Fourth, they permit certain kinds of specialized analysis, such as throughput and deadlock analysis. Finally, they naturally support concurrent execution. Each filter can be implemented as a separate task and potentially executed in parallel with other filters. But these systems also have their disadvantages.[3] First, pipe-and-filter systems often lead to a batch organization of processing. Although filters can process data incrementally, they are inherently independent, so the designer must think of each filter as providing a complete transformation of input data to output data. In particular, because of their transformational character, pipe-and-filter systems are typically not good at handling interactive applications. This problem is most severe when incremental display updates are required, because the output pattern for incremental updates is radically different from the pattern for filter output. Second, they may be hampered by having to maintain correspondences between two separate but related streams. Third, depending on the implementation, they may force a lowest common denominator on data transmission, resulting in added work for each filter to parse and unparse its data. This, in turn, can lead both to loss of performance and to increased complexity in writing the filters themselves.	3 1½ 2

2 a) Main program/subroutine with shared data



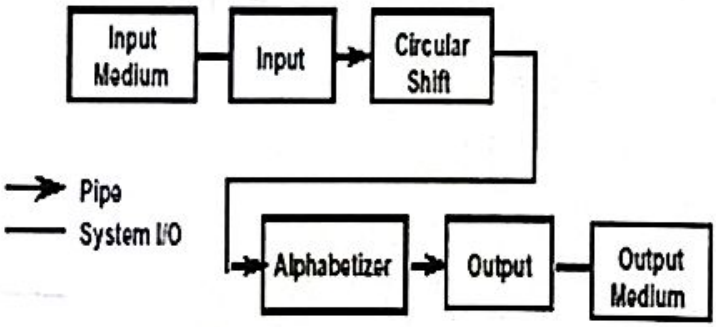
2

Drawbacks:

- Changes in data storage format affects all modules
- Changes in algorithm not well supported
- Enhancements not easily encorporated
- Not supportive of reuse

1

Pipes and filters

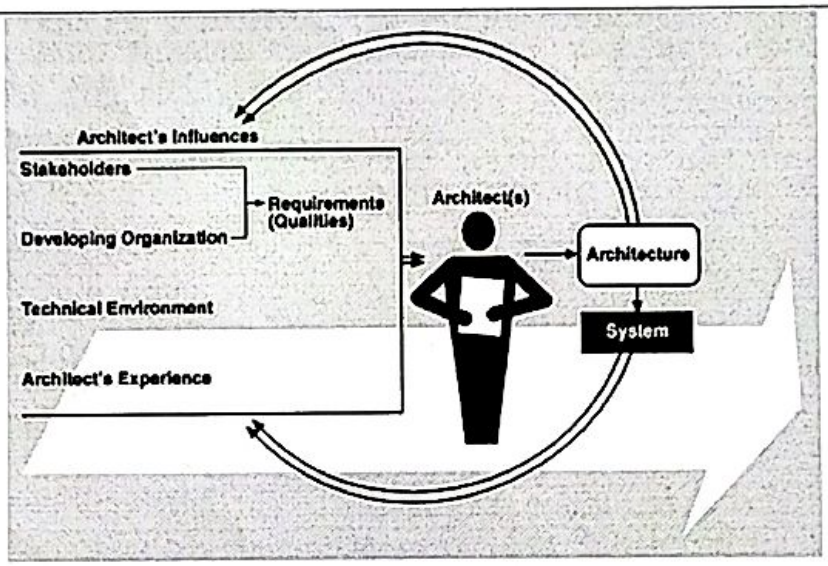
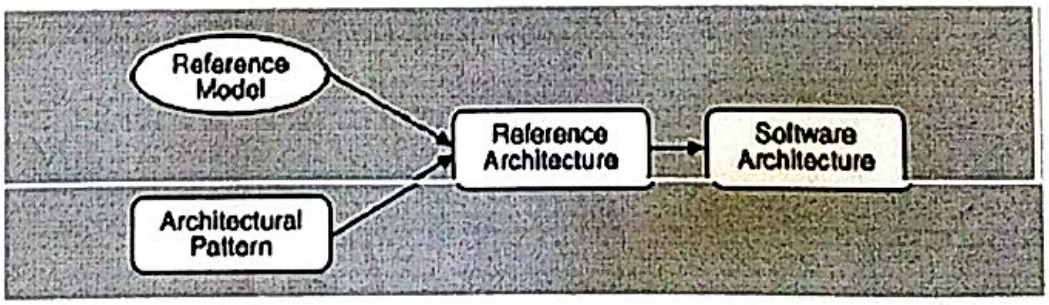


2

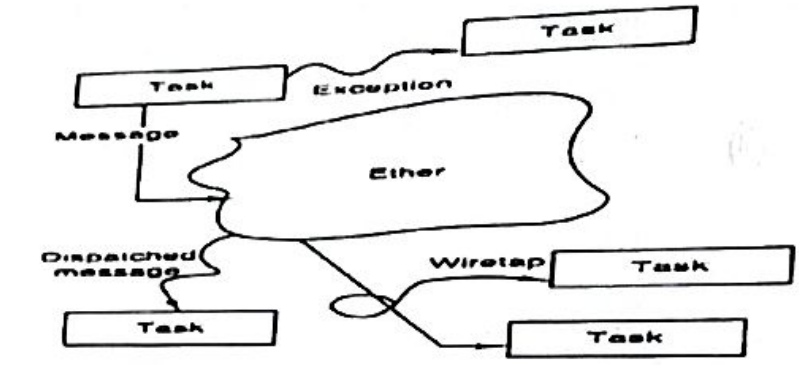
Drawbacks:

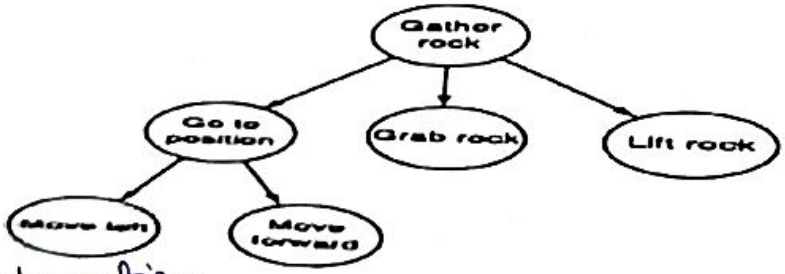
- It is virtually impossible to modify the design to support an interactive system
- The solution is inefficient in terms of its use of space, since each filter must copy all of the data to its output ports
- Changes in processing algorithm- reuse affects
- Overhead: parsing and unparsing the data into pipes
- Extra execution overhead

1

2	b)	 ➢ The architecture affects the structure of the developing organization ➢ The architecture can affect the goals of the developing organization ➢ The architecture can affect customer requirements for the next system by giving the customer the opportunity to receive a system (based on the same architecture) in a more reliable, timely, and economical manner than if the subsequent system were to be built from scratch ➢ The process of system building will affect the architect's experience with subsequent systems by adding to the corporate experience base. A few systems will influence and actually change the software engineering culture, that is, the technical environment in which system builders operate and learn.	4 1/2 9
3	a)	**Building the ABC by identifying the influences to and from architectures:** ➢ Architectures are influenced by system stakeholders ➢ Architectures are influenced by the developing organization ➢ Architectures are influenced by the background and experience of the architects Architectures are influenced by the technical environment	6
3	b)	**Architectural Patterns, Reference Models, and Reference Architectures** 	3 1/2

ii) Implicit Invocation

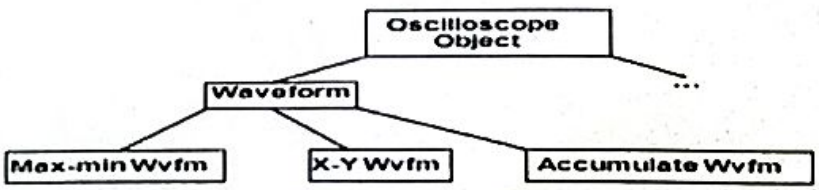




Explanation

1½

1½

4	b)	**Layered systems** **Diagram.....** Advantages: • The implementers can partition a complex problem into a sequence of incremental steps. • In layered systems each layer interact with atmost the layers above and below, changes to the fuction of one layer affect atmost two other layers • They support reuse. Disadvantages: • Performance may require closer coupling between logically high level functions and their lower lavel implementations • It is difficult to find right level of abstraction • It is difficult to map existing protocols into the ISO framework
5	a)	**Instrumentaion software** i) **Object oriented model** **Focused on producing object-oriented model of the domain** **This produced a good model of the data involved** 

2½

2

2

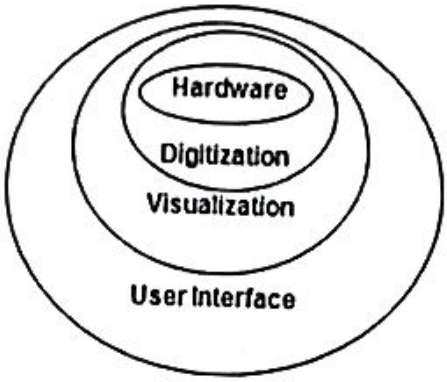
1

No overall model of how the types fit together
Led to confusion about partitioning the functionality

- Should measurements be associated with data type of what is being measured? Or represented externally
- Which objects should the interface interact with.

2

(i) **Layered model**



1

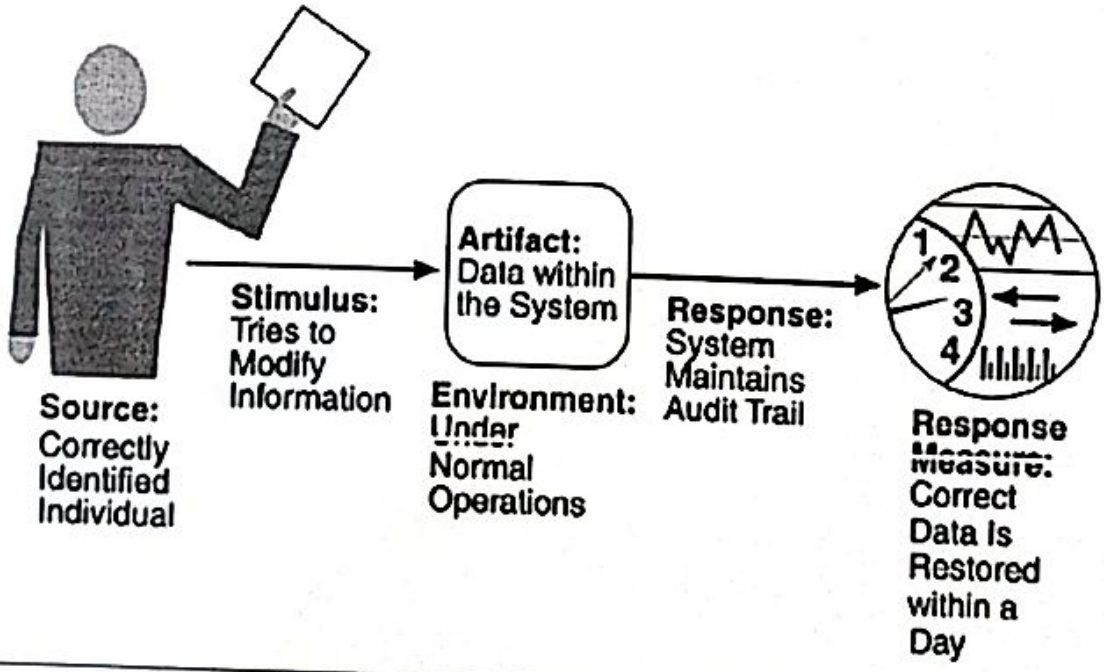
This model was intuitively appealing since it partitioned up the functionality into well defined groups.
However, wrong model:

- main problem was that the boundaries of abstraction enforced by the layers conflict with what was really needed.

2

user interactions with the visualizations but real oscilloscopes must interact at several levels

5 b) **Security general scenario**



3

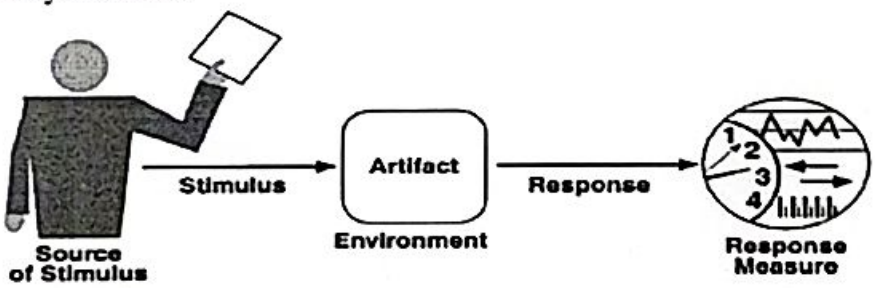
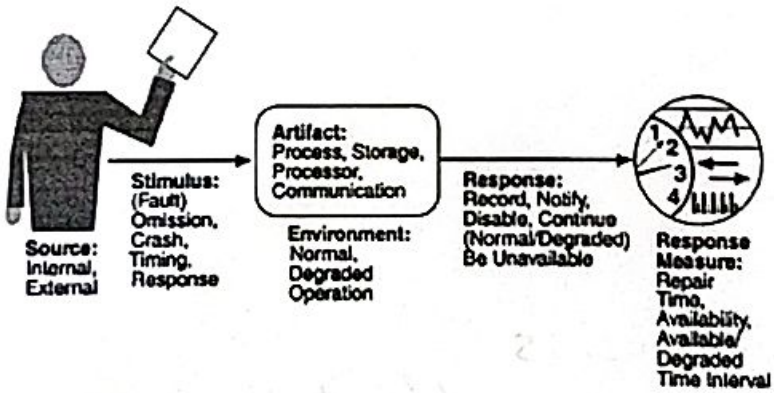
		Portion of Scenario — **Possible Values** Source — Individual or system that is correctly identified, identified incorrectly, of unknown identity who is internal/external, authorized/not authorized with access to limited resources, vast resources Stimulus — Tries to display data, change/delete data, access system services, reduce availability to system services Artifact — System services; data within system Environment — Either online or offline, connected or disconnected, firewalled or open	3½
6	a)	**System quality attributes**  **Each part explaination....**	3 3
6	b)	Availability scenario  The availability of a system is the probability that it will be operational when it is needed. This is typically defined as $\alpha = \frac{\text{mean time to failure}}{\text{mean time to failure + mean time to repair}}$	3 1

Table 4.1. Availability General Scenario Generation

Portion of Scenario	Possible Values
Source	Internal to the system; external to the system
Stimulus	Fault: omission, crash, timing, response
Artifact	System's processors, communication channels, persistent storage, processes
Environment	Normal operation;
	degraded mode (i.e., fewer features, a fall back solution)
Response	System should detect event and do one or more of the following:
	record it
	notify appropriate parties, including the user and other systems
	disable sources of events that cause fault or failure according to defined rules
	be unavailable for a prespecified interval, where interval depends on criticality of system
	continue to operate in normal or degraded mode
Response Measure	Time interval when the system must be available
	Availability time
	Time interval in which system can be in degraded mode
	Repair time

2½

SUBJECT TEACHERS

1. Mr. Sushant mangasuli 20/3/17
2. Mrs. Vineetha Pais 21/3/17.

H.O.D 21/3/17

USN | | | | | | | | | | |

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY, MOODBIDRI

(A unit of Alva's Education Foundation)

Eighth Semester B.E. (CSE) **I. A Test Examinations-II**

27th April – 2017

10IS81-SOFTWARE ARCHITECTURE

Duration: 1 ½ Hour Max. Marks: 50

*Note: 1) Answer **Two Full** questions from each Part.*

PART-A

1. a) Define tactics. Explain availability tactics with a neat diagram — 6 M
   b) Explain the scenario of MVC which shows how user input that results in changes to the model triggers the change propagation mechanism — 6½ M

2. a) Explain the implementation of PAC — 6 M
   b) Explain CRC card of Broker architectural pattern — 6½ M

3. a) List and explain the components involved in microkernel architecture. — 6M
   b) Explain Testability tactics with a neat diagram — 6½ M

PART B

4. a) Explain the dynamic scenario of microkernel which shows the behavior of Microkernel architecture when an external server requests a service that is provided b internal server — 6 M
   b) Explain security tactics with a neat diagram — 6½ M

5. a) What are the steps involved in implementing broker architectural pattern — 6 M
   b) Explain Performance tactics with a neat diagram — 6½ M

6. a) Explain the CRC card of PAC architectural pattern. — 6 M
   b) Explain the benefits and liabilities of MVC architectural pattern — 6½ M

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY, MOODBIDRI

(A unit of Alva's Education Foundation)

Eighth Semester B.E. (CSE) **I. A Test Examinations-II**

SCEHEME OF EVALUATION

SUB CODE: 10IS81 **SOFTWARE ARCHITECTURES**

Note: Answer any Two full questions.

1 a) **Availability Tactics:** 6

FAULT DETECTION

Three widely used tactics for recognizing faults are ping/echo, heartbeat, and exceptions.

- *Ping/echo.* One component issues a ping and expects to receive back an echo, within a predefined time, from the component under scrutiny. This can be used within a group of components mutually responsible for one task. It can also used be used by clients to ensure that a server object and the communication path to the server are operating within the expected performance bounds. "Ping/echo" fault detectors can be organized in a hierarchy, in which a lowest-level detector pings the software processes with which it shares a processor, and the higher-level fault detectors ping lower-level ones. This uses less communication bandwidth than a remote fault detector that pings all processes.
- *Heartbeat (dead man timer).* In this case one component emits a heartbeat message periodically and another component listens for it. If the heartbeat fails, the originating component is assumed to have failed and a fault correction component is notified. The heartbeat can also carry data. For example, an automated teller machine can periodically send the log of the last transaction to a server. This message not only acts as a heartbeat but also carries data to be processed.
- *Exceptions.* One method for recognizing faults is to encounter an exception, which is raised when one of the fault classes we discussed in Chapter 4 is recognized. The exception handler typically executes in the same process that introduced the exception.

FAULT PREVENTION

The following are some fault prevention tactics

- *Removal from service.* This tactic removes a component of the system from operation to undergo some activities to prevent anticipated failures. One example is rebooting a component to prevent memory leaks from causing a failure. If this removal from service is automatic, an architectural strategy can be designed to support it. If it is manual, the system must be designed to support it.
- *Transactions.* A transaction is the bundling of several sequential steps such that the entire bundle can be undone at once. Transactions are used to prevent any data from being affected if one step in a process fails and also to prevent collisions among several simultaneous threads accessing the same data.
- *Process monitor.* Once a fault in a process has been detected, a monitoring process can delete the nonperforming process and create a new instance of it, initialized to some appropriate state as in the spare tactic.

FAULT RECOVERY

Fault recovery consists of preparing for recovery and making the system repair. Some preparation and repair tactics follow.

- *Voting*. Processes running on redundant processors each take equivalent input and compute a simple output value that is sent to a voter. If the voter detects deviant behavior from a single processor, it fails it. The voting algorithm can be "majority rules" or "preferred component" or some other algorithm. This method is used to correct faulty operation of algorithms or failure of a processor and is often used in control systems. If all of the processors utilize the same algorithms, the redundancy detects only a processor fault and not an algorithm fault. Thus, if the consequence of a failure is extreme, such as potential loss of life, the redundant components can be diverse.

  One extreme of diversity is that the software for each redundant component is developed by different teams and executes on dissimilar platforms. Less extreme is to develop a single software component on dissimilar platforms. Diversity is expensive to develop and maintain and is used only in exceptional circumstances, such as the control of surfaces on aircraft. It is usually used for control systems in which the outputs to the voter are straightforward and easy to classify as equivalent or deviant, the computations are cyclic, and all redundant components receive equivalent inputs from sensors. Diversity has no downtime when a failure occurs since the voter continues to operate. Variations on this approach include the Simplex approach, which uses the results of a "preferred" component unless they deviate from those of a "trusted" component, to which it defers. Synchronization among the redundant components is automatic since they are all assumed to be computing on the same set of inputs in parallel.

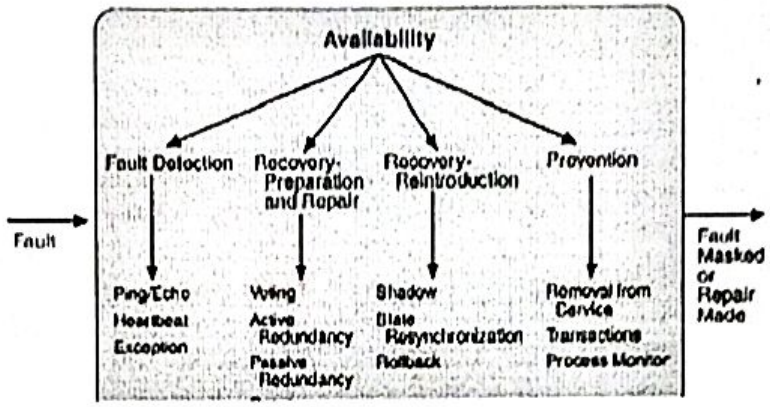
- *Active redundancy (hot restart)*. All redundant components respond to events in parallel. Consequently, they are all in the same state. The response from only one component is used (usually the first to respond), and the rest are discarded. When a fault occurs, the downtime of systems using this tactic is usually milliseconds since the backup is current and the only time to recover is the switching time. Active redundancy is often used in a client/server configuration, such as database management systems, where quick responses are necessary even when a fault occurs. In a highly available distributed system, the redundancy may be in the communication paths. For example, it may be desirable to use a LAN with a number of parallel paths and place each redundant component in a separate path. In this case, a single bridge or path failure will not make all of the system's components unavailable.

  Synchronization is performed by ensuring that all messages to any redundant component are sent to all redundant components. If communication has a possibility of being lost (because of noisy or overloaded communication lines), a reliable transmission protocol can be used to recover. A reliable transmission protocol requires all recipients to acknowledge receipt together with some integrity indication such as a checksum. If the sender cannot verify that all recipients have received the message, it will resend the message to those components not acknowledging receipt. The resending of unreceived messages (possibly over different communication paths) continues until the sender marks the recipient as out of service.

- *Passive redundancy (warm restart/dual redundancy/triple redundancy)*. One component (the primary) responds to events and informs the other components (the standbys) of state updates they must make. When a fault occurs, the system must first ensure that the backup state is sufficiently fresh before resuming services. This approach is also used in control systems, often when the inputs come over communication channels or from sensors and have to be switched from the primary to the backup on failure. Chapter 6, describing an air traffic control example, shows a system using it. In the air traffic control system, the secondary decides when to take over from the primary, but in other systems this decision can be done in other components. This tactic depends on the standby components taking over reliably. Forcing switchovers periodically—for example, once a day or once a week—increases the availability of the system. Some database systems force a switch with storage of every new data item. The new data item is stored in a shadow page and the old page becomes a backup for recovery. In this case, the downtime can usually be limited to seconds.

  Synchronization is the responsibility of the primary component, which may use atomic broadcasts to the secondaries to guarantee synchronization.

- *Spare*. A standby spare computing platform is configured to replace many different failed components. It must be rebooted to the appropriate software configuration and have its state initialized when a failure occurs. Making a checkpoint of the system state to a persistent device periodically and logging all state changes to a persistent device allows for the spare to be set to the appropriate state. This is often used as the standby client workstation, where the user can move when a failure occurs. The downtime for this tactic is usually minutes.



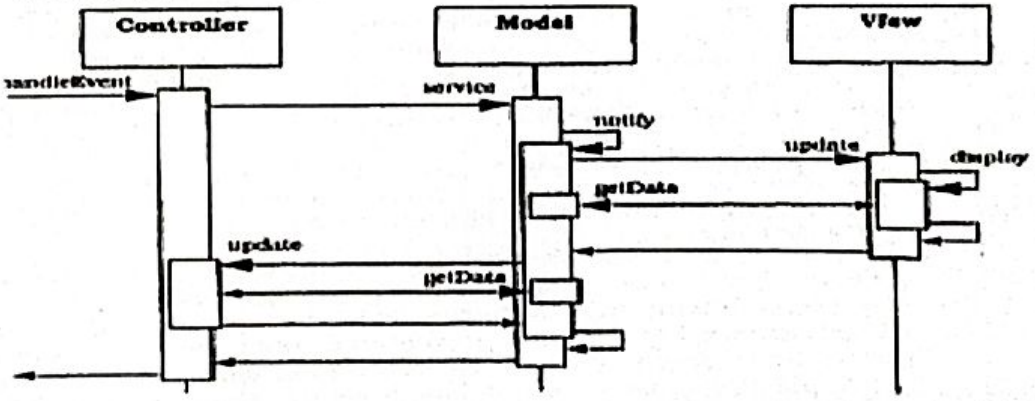
after it has been corrected. Such tactics are shadow operation, state resynchronization, and rollback.

- *Shadow operation.* A previously failed component may be run in "shadow mode" for a short time to make sure that it mimics the behavior of the working components before restoring it to service.
- *State resynchronization.* The passive and active redundancy tactics require the component being restored to have its state upgraded before its return to service. The updating approach will depend on the downtime that can be sustained, the size of the update, and the number of messages required for the update. A single message containing the state is preferable, if possible. Incremental state upgrades, with periods of service between increments, lead to complicated software.
- *Checkpoint/rollback.* A checkpoint is a recording of a consistent state created either periodically or in response to specific events. Sometimes a system fails in an unusual manner, with a detectably inconsistent state. In this case, the system should be restored using a previous checkpoint of a consistent state and a log of the transactions that occurred since the snapshot was taken.

1 b) **Scenario 1** shows how user input that results in changes to the model triggers the change-propagation mechanism:

- The controller accepts user input as the event-handling procedure, interprets the event, and activates a service procedure of the model.
- The model performs the requested service. This results in a change to its internal data.
- The model notifies all views and controllers registered with the change-propagation mechanism of the change by calling their update procedures.
- Each view requests the changed data from the model and re-displays itself on the screen.
- Each registered controller retrieves data from the model to enable or disable certain user functions. For example, enabling the menu entry for saving data can be a consequence of modifications to the data of the model.
- The original controller regains control and returns from its event-handling procedure.

6½



2.	a)	**Implementation of PAC** 1 *Define a model of the application.* Analyze the problem domain and map it onto an appropriate software structure. Do not consider the distribution of components to PAC agents when performing this step. Concentrate on finding a proper decomposition and organization of the application domain. To this end, answer the following questions: • Which services should the system provide? • Which components can fulfill these services? • What are the relationships between components? • How do the components collaborate? • What data do the components operate on? • How will the user interact with the software? Follow an appropriate analysis method when specifying the model. 2 *Define a general strategy for organizing the PAC hierarchy.* At this point we have not yet defined individual agents, but can specify general guidelines for organizing the hierarchy of cooperating agents. 3 *Specify the top-level PAC agent.* Identify those parts of the analysis model that represent the functional core of the system. These are mostly components that maintain the global data model of the system, and components directly operating on this data. Identify also all user interface elements that are common to the whole application, such as menu bars or dialogs with information about the system. All components identified in this step will be part of the top-level agent. 4 *Specify the bottom-level PAC agents.* Identify those components of the analysis model that represent the smallest self-contained units of the system on which the user can perform operations or view presentations. In our example system, these units are the various diagrams and charts presenting election data, and the spreadsheet for entering this data. 5 *Specify bottom-level PAC agents for system services.* Often an application includes additional services that are not directly related to its primary subject. In our example system we define an error handler. Other systems may provide services for communicating with other systems or for configuration purposes. Each of these services, including their human-computer interaction, can be implemented as a separate bottom-level agent [BaCo91]. 6 *Specify intermediate-level PAC agents to compose lower-level PAC agents.* Often, several lower-level agents together form a higher-level semantic concept on which users can operate. 7 *Specify intermediate-level PAC agents to coordinate lower-level PAC agents.* Many systems offer multiple views of the same semantic concept. For example, in text editors you find 'layout' and 'edit' views of a text document. When the data in one view changes, all other views must be updated. Such coordination components, which you may have identified when modeling the analysis model, provide their own human-computer interaction; for example, menu entries and associated callback functions. The view coordinator agent of our example system is such an intermediate-level agent. To implement agents that coordinate multiple views you may apply the View Handler pattern (291). 8 *Separate core functionality from human-computer interaction.* For every PAC agent, introduce presentation and abstraction components. All components that provide the user interface of the agent, such as graphical images presented to the user, presentation-specific data like screen coordinates, or menus, windows, and dialogs form the presentation part. All components that maintain core data or operate on them form the abstraction.	6

2 b) Broker CRC cards: 6½

Class: Client	Collaborators
Responsibility • Implements user functionality. • Sends requests to servers through a client-side proxy.	• Client-side Proxy • Broker

Class: Server	Collaborators
Responsibility • Implements services. • Registers itself with the local broker. • Sends responses and exceptions back to the client through a server-side proxy.	• Server-side Proxy • Broker

Class: Broker	Collaborators
Responsibility • (Un)registers servers. • Offers APIs. • Transfers messages. • Error recovery. • Interoperates with other brokers through bridges. • Locates servers.	• Client • Server • Client-side Proxy • Server-side Proxy • Bridge

Class: Client-side Proxy	Collaborators
Responsibility • Encapsulates system-specific functionality. • Mediates between the client and the broker.	• Client • Broker

Class: Server-side Proxy	Collaborators
Responsibility • Calls services within the server. • Encapsulates system-specific functionality. • Mediates between the server and the broker.	• Server • Broker

3 a) Components involved in microkernel architecture: 6

Class: Microkernel	Collaborators
Responsibility • Provides core mechanisms. • Offers communication facilities. • Encapsulates system dependencies. • Manages and controls resources.	• Internal Server

Class: Internal Server	Collaborators
Responsibility • Implements additional services. • Encapsulates some system specifics.	• Microkernel

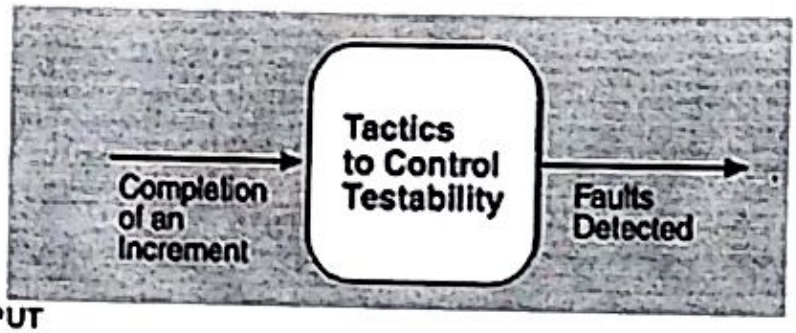
Class: External Server	Collaborators: • Microkernel
Responsibility • Provides programming interfaces for its clients.	

Class: Client	Collaborators: • Adapter
Responsibility • Represents an application.	

Class: Adapter	Collaborators: • External Server • Microkernel
Responsibility • Hides system dependencies such as communication facilities from the client. • Invokes methods of external servers on behalf of clients.	

3 b) **Testability Tactics:**

6½



INPUT/OUTPUT

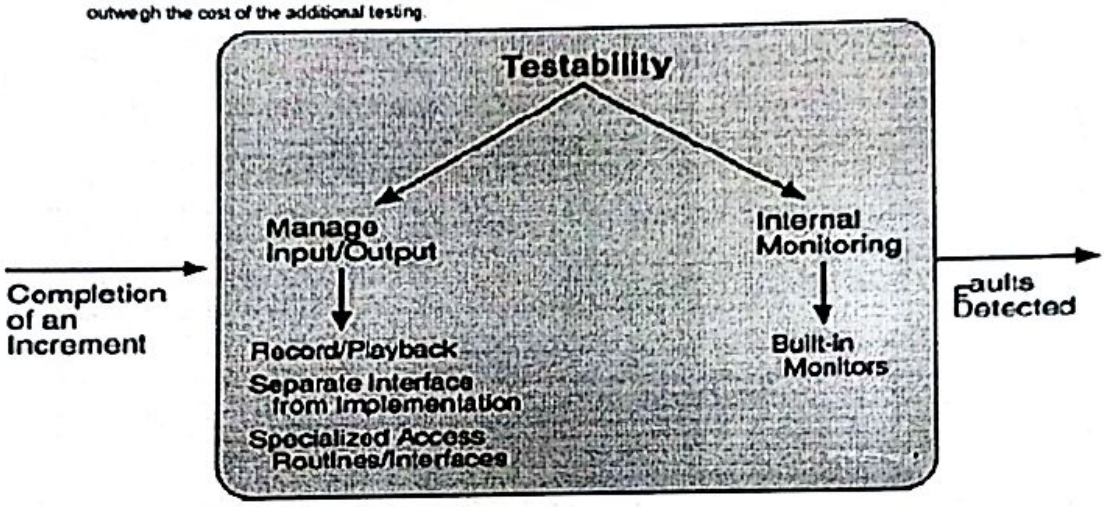
There are three tactics for managing input and output for testing.

- *Record/playback.* Record/playback refers to both capturing information crossing an interface and using it as input into the test harness. The information crossing an interface during normal operation is saved in some repository and represents output from one component and input to another. Recording this information allows test input for one of the components to be generated and test output for later comparison to be saved.
- *Separate interface from implementation.* Separating the interface from the implementation allows substitution of implementations for various testing purposes. Stubbing implementations allows the remainder of the system to be tested in the absence of the component being stubbed. Substituting a specialized component allows the component being replaced to act as a test harness for the remainder of the system.
- *Specialize access routes/interfaces.* Having specialized testing interfaces allows the capturing or specification of variable values for a component through a test harness as well as independently from its normal execution. For example, metadata might be made available through a specialized interface that a test harness would use to drive its activities. Specialized access routes and interfaces should be kept separate from the access routes and interfaces for required functionality. Having a

INTERNAL MONITORING

A component can implement tactics based on internal state to support the testing process.

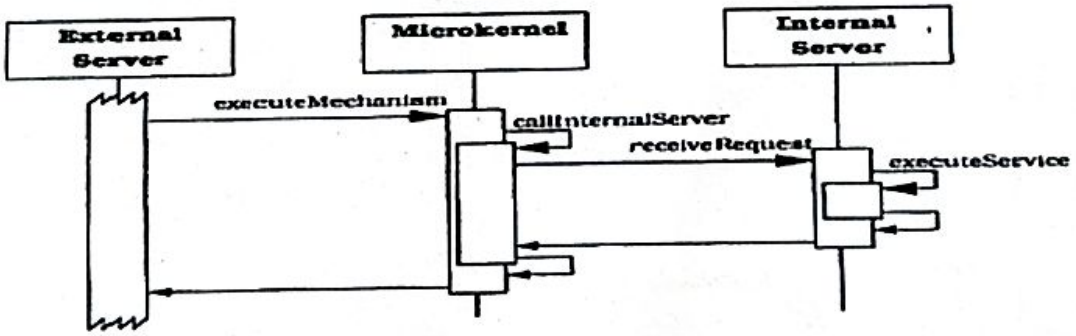
- Built-in monitors. The component can maintain state, performance load, capacity, security, or other information accessible through an interface. This interface can be a permanent interface of the component or it can be introduced temporarily via an instrumentation technique such as aspect-oriented programming or preprocessor macros. A common technique is to record events when monitoring states have been activated. Monitoring states can actually increase the testing effort since tests may have to be repeated with the monitoring turned off. Increased visibility into the activities of the component usually more than outweigh the cost of the additional testing.



4 a) **Dynamic Scenario of Microkernel:** 6

Scenario II illustrates the behavior of a Microkernel architecture when an external server requests a service that is provided by an internal server. In this scenario we assume that the internal server is implemented as a separate process. It could alternatively be implemented as a shared library that is dynamically linked to the microkernel

- The external server sends a service request to the microkernel.
- A procedure of the programming interface of the microkernel is called to handle the service request. During method execution the microkernel sends a request to an internal server.
- After receiving the request, the internal server executes the requested service and sends all results back to the microkernel.
- The microkernel returns the results back to the external server.
- Finally, the external server retrieves the results and continues with its control flow.



4 b) **Security tactics:**

DETECTING ATTACKS

The detection of an attack is usually through an intrusion detection system. Such systems work by comparing network traffic patterns to a database. In the case of misuse detection, the traffic pattern is compared to historic patterns of known attacks. In the case of anomaly detection, the traffic pattern is compared to a historical baseline of itself. Frequently, the packets must be filtered in order to make comparisons. Filtering can be on the basis of protocol, TCP flags, payload sizes, source or destination address, or port number.

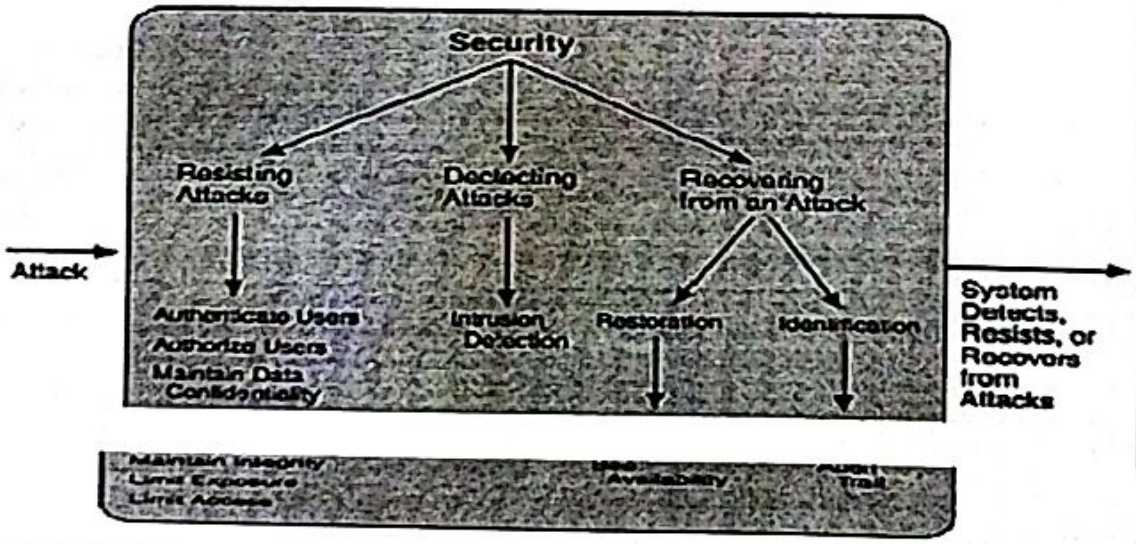
Intrusion detectors must have some sort of sensor to detect attacks, managers to do sensor fusion, databases for storing events for later analysis, tools for offline reporting and analysis, and a control console so that the analyst can modify intrusion detection actions.

RECOVERING FROM ATTACKS

Tactics involved in recovering from an attack can be divided into those concerned with restoring state and those concerned with attacker identification (for either preventive or punitive purposes).

The tactics used in restoring the system or data to a correct state overlap with those used for availability since they are both concerned with recovering a consistent state from an inconsistent state. One difference is that special attention is paid to maintaining redundant copies of system administrative data such as passwords, access control lists, domain name services, and user profile data.

The tactic for identifying an attacker is to maintain *an audit trail*. An audit trail is a copy of each transaction applied to the data in the system together with identifying information. Audit information can be used to trace the actions of an attacker, support nonrepudiation (it provides evidence that a particular request was made), and support system recovery. Audit trails are often attack targets themselves and therefore should be maintained in a trusted fashion.



5	a)	**Broker Pattern** **Implementation:** 1 *Define an object model, or use an existing model.* Your choice of object model has a major impact on all other parts of the system under development. Each object model must specify entities such as object names, objects, requests, values, exceptions, supported types, type extensions, interfaces and operations. In this first step you should only consider semantic issues. If the object model has to be extensible, prepare the system for future enhancements. For example, specify a basic object model and how it can be refined systematically using extensions. More information on this topic is available in [OMG92]. 2 *Decide which kind of component-interoperability the system should offer.* You can design for interoperability either by specifying a binary standard or by introducing a high-level interface definition language (IDL). An IDL file contains a textual description of the interfaces a server offers to its clients. The binary approach needs support from your programming language. For example, binary method tables are available in Microsoft Object Linking and Embedding (OLE) [Bro94]. These tables consist of pointers to method implementations, and enable clients to call methods indirectly using pointers. Access to OLE objects is only supported by compilers or interpreters that know the physical structure of these tables. 3 *Specify the APIs the broker component provides for collaborating with clients and servers.* On the client side, functionality must be available for constructing requests, passing them to the broker and receiving responses. Decide whether clients should only be able to invoke server operations statically, allowing clients to bind the invocations at compile-time. If you want to allow dynamic invocations[15] of servers as well, this has a direct impact on the size or number of APIs. For example, you need some way of asking the broker about existing server objects. You can implement this with the help of a meta-level schema, as described in the Reflection pattern (193). 4 *Use proxy objects to hide implementation details from clients and servers.* On the client side, a local proxy (263) object represents the remote server object called by the client. On the server side, a proxy 5 *Design the broker component* in parallel with steps 3 and 4. In this step we describe how to develop a broker component that acts as a messenger for every message passed from a client to a server and vice-versa. To increase the performance of the whole system, some implementations do not transmit messages via the broker. In these systems most of the work is done by the proxies, while the broker is still responsible for establishing the initial communication link between clients and servers. A direct communication between client and server is only possible when both of them can use the same protocol. We call such systems *Direct Communication Broker systems*	6

5.1 Specify a detailed *on-the-wire* protocol for interacting with client-side proxies and server-side proxies. Plan the mapping of requests, responses, and exceptions to your internal message protocol. In an on-the-wire protocol, the internal message protocol handles the mapping of higher-level structures such as parameter values, method names and return values to corresponding structures specified by the underlying inter-process communication mechanism.

5.2 A local broker must be available for every participating machine in the network. If requests, responses or exceptions are transferred from one network node to another, the corresponding local brokers must communicate with each other using an on-the-wire protocol. Use

5.3 When a client invokes a method of a server, the Broker system is responsible for returning all results and exceptions back to the original client. In other words, the system must remember which client has sent the request. In the *Direct Communication variant* (see the Variants section) there is no need to remember the originator of an invocation, because the client and the server are directly connected through a communication channel. In Indirect Broker systems you can choose between different means of remembering the sender of a request. For example, you may specify the client's address as an additional, invisible parameter of the request or message.

5.4 If the proxies (see step 4) do not provide mechanisms for marshaling and unmarshaling parameters and results, you must include that functionality in the broker component.

5.5 If your system supports asynchronous communication between clients and servers, you need to provide *message buffers* within the broker or within the proxies for the temporary storage of messages.

5.6 Include a *directory service* for associating local server identifiers with the physical location of the corresponding servers in the broker. For example, if the underlying inter-process communication protocol is based on TCP/IP, you could use an Internet port number as the physical server location.

5.7 When your architecture requires system-unique identifiers to be generated dynamically during server registration, the broker must offer a *name service* for instantiating such names.

5.8 If your system supports *dynamic method invocation* (see step 3), the broker needs some means for maintaining type information about existing servers. A client may access this information using the broker APIs to construct a request dynamically. You can implement such type information by instantiating the Reflection pattern (193). In this, metaobjects maintain type information that is accessible by a metaobject protocol.

6 *Develop IDL compilers.* Whenever you implement interoperability by providing an interface definition language, you need to build an *IDL compiler* for every programming language you support. An IDL compiler translates the server interface definitions to programming language code. When many programming languages are in use, it is best to develop the compiler as a *framework* that allows the developer to add his own code generators.

5 b) **Performance Tactics:** 6½

RESOURCE DEMAND

Event streams are the source of resource demand. Two characteristics of demand are the time between events in a resource stream (how often a request is made in a stream) and how much of a resource is consumed by each request.

One tactic for reducing latency is to reduce the resources required for processing an event stream. Ways to do this include the following

- Increase *computational efficiency*. One step in the processing of an event or a message is applying some algorithm. Improving the algorithms used in critical areas will decrease latency. Sometimes one resource can be traded for another. For example, intermediate data may be kept in a repository or it may be regenerated depending on time and space resource availability. This tactic is usually applied to the processor but is also effective when applied to other resources such as a disk.
- Reduce *computational overhead*. If there is no request for a resource, processing needs are reduced. In Chapter 17, we will see an example of using Java classes rather than Remote Method Invocation (RMI) because the former reduces communication requirements. The use of intermediaries (so important for modifiability) increases the resources consumed in processing an event stream, and so removing them improves latency. This is a classic modifiability/performance tradeoff

Another tactic for reducing latency is to reduce the number of events processed. This can be done in one of two fashions.

- *Manage event rate*. If it is possible to reduce the sampling frequency at which environmental variables are monitored, demand can be reduced. Sometimes this is possible if the system was overengineered. Other times an unnecessarily high sampling rate is used to establish harmonic periods between multiple streams. That is, some stream or streams of events are oversampled so that they can be synchronized.
- *Control frequency of sampling*. If there is no control over the arrival of externally generated events, queued requests can be sampled at a lower frequency, possibly resulting in the loss of requests.

Other tactics for reducing or managing demand involve controlling the use of resources.

- *Bound execution times*. Place a limit on how much execution time is used to respond to an event. Sometimes this makes sense and sometimes it does not. For iterative, data-dependent algorithms, limiting the number of iterations is a method for bounding execution times.
- *Bound queue sizes*. This controls the maximum number of queued arrivals and consequently the resources used to process the arrivals.

RESOURCE MANAGEMENT

Even though the demand for resources may not be controllable, the management of these resources affects response times. Some resource management tactics are:

- *Introduce concurrency*. If requests can be processed in parallel, the blocked time can be reduced. Concurrency can be introduced by processing different streams of events on different threads or by creating additional threads to process different sets of activities. Once concurrency has been introduced, appropriately allocating the threads to resources (load balancing) is important in order to maximally exploit the concurrency.
- *Maintain multiple copies of either data or computations*. Clients in a client-server pattern are replicas of the computation. The purpose of replicas is to reduce the contention that would occur if all computations took place on a central server. Caching is a tactic in which data is replicated, either on different speed repositories or on separate repositories, to reduce contention. Since the data being cached is usually a copy of existing data, keeping the copies consistent and synchronized becomes a responsibility that the system must assume.

RESOURCE ARBITRATION

Whenever there is contention for a resource, the resource must be scheduled. Processors are scheduled, buffers are scheduled, and networks are scheduled. The architect's goal is to understand the characteristics of each resource's use and choose the scheduling strategy that is compatible with it.

A scheduling policy conceptually has two parts: a priority assignment and dispatching. All scheduling policies assign priorities. In some cases the assignment is as simple as first-in/first-out. In other cases, it can be tied to the deadline of the request or its semantic importance. Competing criteria for scheduling include optimal resource usage, request importance, minimizing the number of resources used, minimizing latency, maximizing throughput, preventing starvation to ensure fairness, and so forth. The architect needs to be aware of these possibly conflicting criteria and the effect that the chosen tactic has on meeting them.

A high-priority event stream can be dispatched only if the resource to which it is being assigned is available. Sometimes this depends on pre-empting the current user of the resource. Possible preemption options are as follows: can occur anytime; can occur only at specific pre-emption points; and executing processes cannot be pre-empted. Some common scheduling policies are:

1. First-in/First-out. FIFO queues treat all requests for resources as equals and satisfy them in turn. One possibility with a FIFO queue is that one request will be stuck behind another one that takes a long time to generate a response. As long as all of the requests are truly equal, this is not a problem, but if some requests are of higher priority than others, it is problematic.

2. *Fixed-priority scheduling.* Fixed-priority scheduling assigns each source of resource requests a particular priority and assigns the resources in that priority order. This strategy insures better service for higher-priority requests but admits the possibility of a low-priority, but important, request taking an arbitrarily long time to be serviced because it is stuck behind a series of higher-priority requests. Three common prioritization strategies are

   - *semantic importance.* Each stream is assigned a priority statically according to some domain characteristic of the task that generates it. This type of scheduling is used in mainframe systems where the domain characteristic is the time of task initiation.

   - *deadline monotonic.* Deadline monotonic is a static priority assignment that assigns higher priority to streams with shorter deadlines. This scheduling policy is used when streams of different priorities with real-time deadlines are to be scheduled.

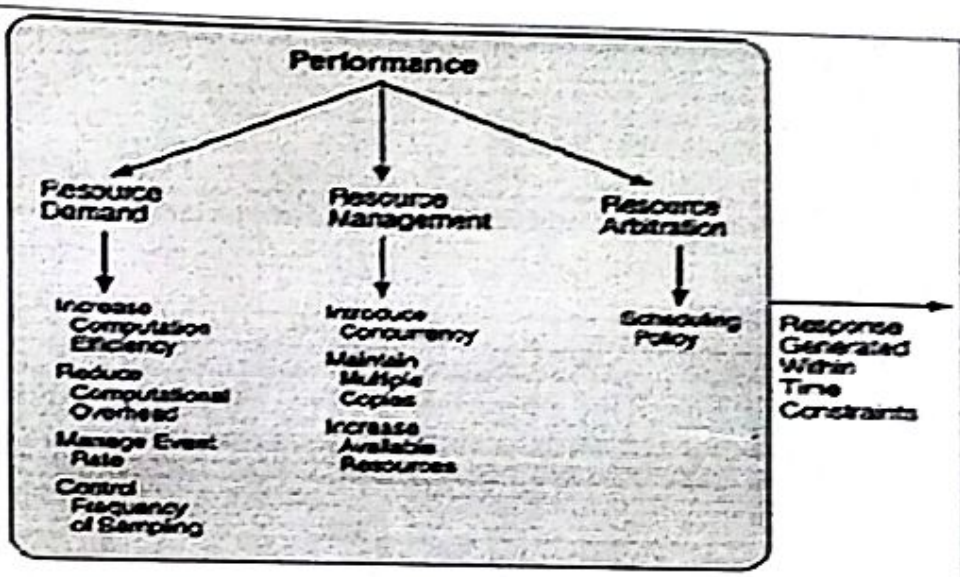
   - *rate monotonic.* Rate monotonic is a static priority assignment for periodic streams that assigns higher priority to streams with shorter periods. This scheduling policy is a special case of deadline monotonic but is better known and more likely to be supported by the operating system.

3. Dynamic *priority scheduling:*

   - *round robin.* Round robin is a scheduling strategy that orders the requests and then, at every assignment possibility, assigns the resource to the next request in that order. A special form of round robin is a cyclic executive where assignment possibilities are at fixed time intervals.

   - *earliest deadline first.* Earliest deadline first assigns priorities based on the pending requests with the earliest deadline.

4. Static *scheduling.* A cyclic executive schedule is a scheduling strategy where the pre-emption points and the sequence of assignment to the resource are determined offline.



6 a) CRC card of PAC:

6

Class: Top-level Agent	Collaborators
Responsibility • Provides the functional core of the system. • Controls the PAC hierarchy.	• Intermediate-level Agent • Bottom-level Agent

Class: Interm. -level Agent	Collaborators
Responsibility • Coordinates lower-level PAC agents. • Composes lower-level PAC agents to a single unit of higher abstraction.	• Top-level Agent • Intermediate-level Agent • Bottom-level Agent

Class: Bottom-level Agent	Collaborators
Responsibility • Provides a specific view of the software or a system service, including its associated human-computer interaction.	• Top-level Agent • Intermediate-level Agent

6 b) MVC

6½

Benefits:

Multiple views of the same model. MVC strictly separates the model from the user-interface components. Multiple views can therefore be implemented and used with a single model. At run-time, multiple views may be open at the same time, and views can be opened and closed dynamically.

Synchronized views. The change-propagation mechanism of the model ensures that all attached observers are notified of changes to the application's data at the correct time. This synchronizes all dependent views and controllers.

'Pluggable' views and controllers. The conceptual separation of MVC allows you to exchange the view and controller objects of a model. User interface objects can even be substituted at run-time.

Exchangeability of 'look and feel'. Because the model is independent of all user-interface code, a port of an MVC application to a new platform does not affect the functional core of the application. You only need suitable implementations of view and controller components for each platform.

Framework potential. It is possible to have an application framework on this pattern, as sketched in implementation steps 7 through 10. The various Smalltalk development environments have proven this approach.

Liabilities:

Increased complexity. Following the Model-View-Controller structure strictly is not always the best way to build an interactive application. Gamma [Gam91] argues that using separate model, view and controller components for menus and simple text elements increases complexity without gaining much flexibility.

Potential for excessive number of updates. If a single user action results in many updates, the model should skip unnecessary change notifications. It may be that not all views are interested in every change-propagated by the model. For example, a view with an iconized window may not need an update until the window is restored to its normal size.

Intimate connection between view and controller. Controller and view are separate but closely-related components, which hinders their individual reuse. It is unlikely that a view would be used without its controller, or vice-versa, with the exception of read-only views that share a controller that ignores all input.

Close coupling of views and controllers to a model. Both view and controller components make direct calls to the model. This implies that changes to the model's interface are likely to break the code of both view and controller. This problem is magnified if the system uses a multitude of views and controllers. You can address this problem by applying the Command Processor pattern (277), as described in the Implementation section, or some other means of indirection.

Inefficiency of data access in view. Depending on the interface of the model, a view may need to make multiple calls to obtain all its display data. Unnecessarily requesting unchanged data from the model weakens performance if updates are frequent. Caching of data within the view improves responsiveness.

Inevitability of change to view and controller when porting. All dependencies on the user-interface platform are encapsulated within view and controller. However, both components also contain code that is independent of a specific platform. A port of an MVC system thus requires the separation of platform-dependent code before rewriting. In the case of an MVC framework or a large composed application, an additional encapsulation of platform dependencies may be required.

Difficulty of using MVC with modern user-interface tools. If portability is not an issue, using high-level toolkits or user interface builders can rule out the use of MVC. It is usually expensive to retrofit toolkit components or the output of user interface layout tools to MVC. Additional wrapping would be the minimum requirement. In addition, many high-level tools or toolkits define their own flow of control and handle some events internally, such as displaying a pop-up menu or scrolling a window. Finally, a high-level user interface platform may already interpret events and offer callbacks for each kind of user activity. Most controller functionality is therefore already provided by the toolkit, and a separate component is not needed.

28/4/17

Signature of Subject Teachers 28/4/17

H.O.D 28/4

USN | | | | | | | | | | |

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY, MOODBIDRI

(A unit of Alva's Education Foundation)

Eighth Semester B.E. (CSE) **I. A Test Examinations-III**

29th May – 2017

10CS81 – SOFTWARE ARCHITECTURES

Duration: 1 ½ Hour Max. Marks: 50

*Note: 1) Answer **Two Full** questions each **Part.***

PART-A

1.	a)	Explain the structure of Reflection Pattern	6
	b)	Explain the dynamic scenario which illustrates the collaboration between base level and meta level when reading the objects stored in a disk file	6 ½
2.	a)	Explain the dynamic scenario of whole part which illustrates the behavior of whole part structure	6
	b)	List out the steps to implement whole part structure	6 ½
3.	a)	Explain the dynamic structure of master slave pattern	6
	b)	Explain the benefits and liabilities of master slave pattern	6 ½

PART-B

4.	a)	Explain the structure of proxy pattern	6
	b)	List out the variants of proxy pattern	6 ½
5.	a)	Explain the dynamic scenario of proxy pattern	6
	b)	Explain the structure of blackboard pattern	6 ½
6.	a)	Explain the dynamic scenario of blackboard pattern	6
	b)	Explain the benefits and liabilities of blackboard pattern	6 ½

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY, MOODBIDRI

(A unit of Alva's Education Foundation)

Eighth Semester B.E. (CSE-A & B Section) **I. A Test -III**

SCEHEME OF EVALUATION

29th May - *2017*

10CS81 – SOFTWARE ARCHITECTURE

PART A

1. a) The meta level consists of a set of metaobjects. Each metaobject encapsulates selected information about a single aspect of the structure, behavior, or state of the base level. There are three sources for such information: It can be provided by the run-time environment of the system, such as C++ type identification objects. It can be user-defined, such as the function call mechanism in the previous section. It can be retrieved from the base level at run-time, for example information about the current state of computation

The base level models and implements the application logic of the software. Its components represent the various services the system offers as well as their underlying data model. The base level also specifies the fundamental collaboration and structural relationships between the components it includes. If the software includes a user interface, this is also part of the base level.

Class Base Level	Collaborators • Meta Level
Responsibility • Implements the application logic. • Uses information provided by the meta level.	

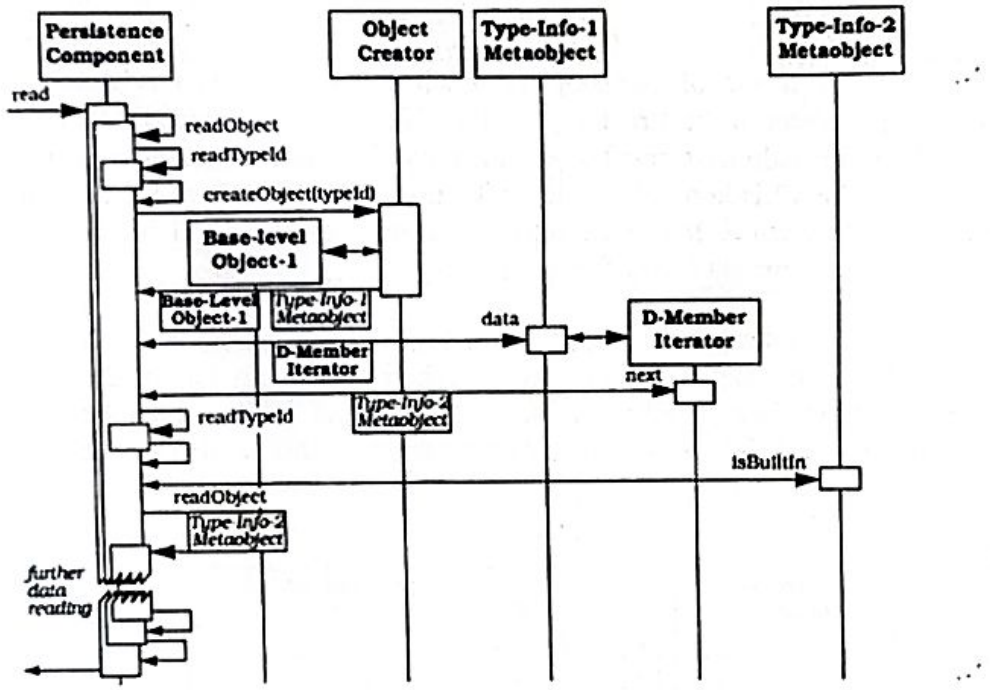
Class Meta Level	Collaborators • Base Level
Responsibility • Encapsulates system internals that may change. • Provides an interface to facilitate modifications to the meta-level.	

Class Metaobject Protocol	Collaborators • Meta Level • Base Level
Responsibility • Offers an interface for specifying changes to the meta level. • Performs specified changes	

1.b)

The scenario is divided into six phases: The user wants to read stored objects. The request is forwarded to the read (l procedure of the persistence component, together with the name of the data file in which the objects are stored. Procedure read 0 opens the data file and calls an internal readobject (l procedure which reads the first type identifier. Procedure readObj ect (calls the metaobject that is responsible for the creation of objects. The 'object creator' metaobject instantiates an 'empty' object of the previously-determined type

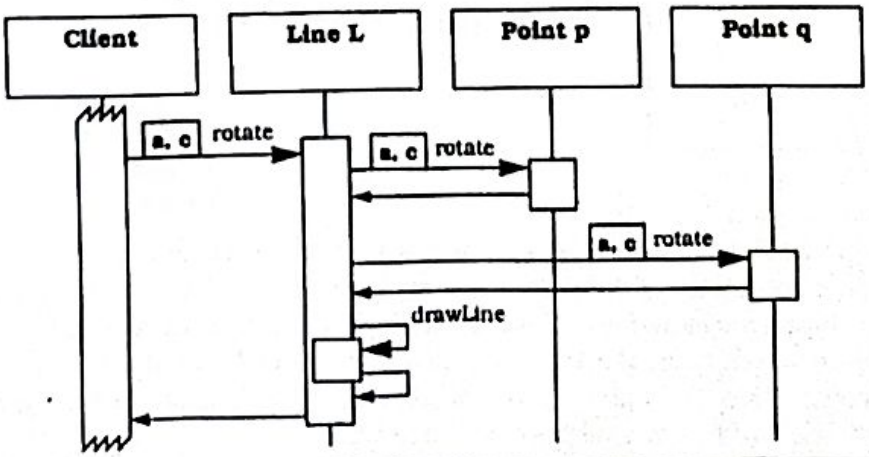
Procedure readobject () requests an iterator over the data members of the object to be read from its corresponding metaobject. The procedure iterates over the data members of the object. Procedure readobject ( ) reads the type identifier for the next data member. If the type identifier denotes a built-in type--a case we do not illustrate--the readobj ect ( ) procedure directly assigns the next data item from the file to the data member, based on the data member's size and offset within the object.



2.a)

The scenario consists of four phases:

- A client invokes the `rotate` method of the line `L` and passes the angle `a` and the rotation center `c` as arguments.
- The line `L` calls the `rotate` method of the point `p`.
- The line `L` calls the `rotate` method of the point `q`.
- The line `L` redraws itself using the new positions of `p'` and `q'` as endpoints.



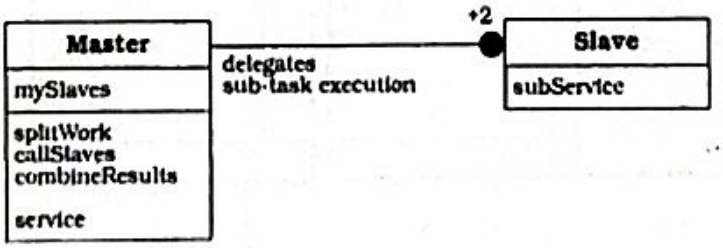
2.b) To implement a Whole-Part structure, apply the following steps: 1 Design the public interface of the Whole. Analyze the functionality the Whole must offer to its clients. Only consider the client's viewpoint in this step. Think of the Whole as an atomic component that is not structured into Parts, and compile a list of methods that together comprise the public interface of the Whole.
2 Separate the Whole into Parts, or synthesize it #om existing ones. There are two approaches to assembling the Parts you need--either assemble a Whole 'bottom-up' from existing Parts, or decompose it 'top-down' into smaller Parts
3 Ifyou follow a bottom-up approach, use existing Parts from component libraries or class libraries and specify their collaboration.
4 If you follow a top-down approach, partition the Mirhole's services into smaller collaborating services and map these collaborating services to separate Parts.
5 SpeczA the services of the Whole in terms of services of the Parts.
6. Implement the Parts. If the Parts are Whole-Part structures themselves, design them recursively starting with step 1.
7. Implement the Whole. Implement the Whole's services based on the structure you developed in the preceding steps

3.a) The master component provides a service that can be solved by applying the 'divide and conquer' principle. It offers an interface that allows clients to access this service. Internally, the master implements functions for partitioning work into several equal sub-tasks, starting and controlling their processing, and computing a final result from all the results obtained. The master also maintains references to all slave instances to which it delegates the processing of sub-tasks. 6

Class Master	**Collaborators** • Slave
Responsibility • Partitions work among several slave components • Starts the execution of slaves • Computes a result from the sub-results the slaves return.	

Class Slave	**Collaborators** -
Responsibility • Implements the sub-service used by the master.	

The structure defined by the Master-Slave pattern is illustrated by the following OMT diagram.



3. b) The Master-Slave design pattern provides several benefits:
Exchangeability and extensibility
Separation of concerns
Efficiency
The Master-Slave pattern suffers from three liabilities:
Feasibility
Machine dependency.
Hard to implement
Portability

PART B

4.a) The proxy offers the same interface as the original, and ensures correct access to the original. To achieve this the proxy maintains a reference to the original it represents. Usually there is a one-to-one relationship between the proxy and the original, though there are exceptions to this rule for Remote and Firewall proxies, two variants of this general pattern. See the Variants section for more information. The abstract original provides the interface implemented by the proxy and the original. In a language like C++, with no notable difference between subtyping and inheritance, both the proxy and the original inherit from the abstract original. Clients code agdnst this interface when accessing the original.

Class Client	Collaborators • Proxy
Responsibilities • Uses the interface provided by the proxy to request a particular service. • Fulfills its own task.	

Class AbstractOriginal	Collaborators -
Responsibilities • Serves as an abstract base class for the proxy and the original.	

Class Proxy	Collaborator • Original
Responsibilities • Provides the interface of the original to clients. • Ensures a safe, efficient and correct access to the original.	

Class Original	Collaborators -
Responsibilities • Implements a particular service.	

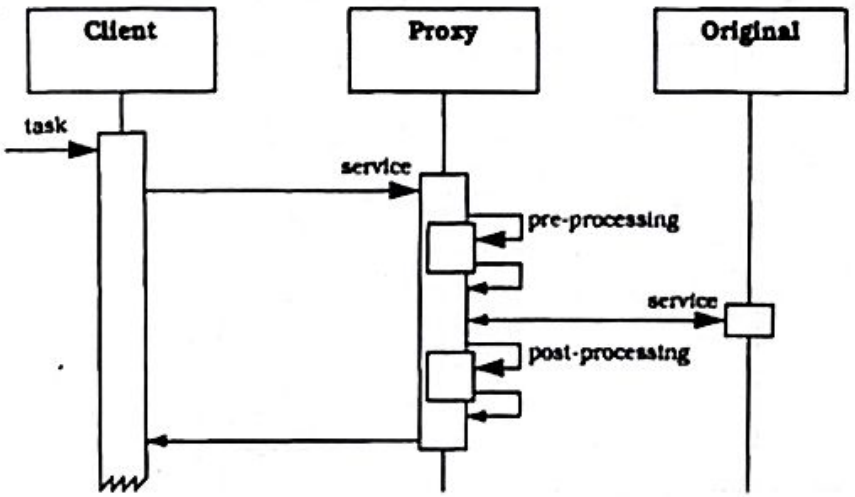
4.b) **Remote Proxy.** Clients of remote components should be shielded from network addresses and inter-process communication protocols.
Protection Proxy. Components must be protected from unauthorized access. **Cache Proxy.** Multiple local clients can share results from remote components. **Synchronization Proxy.** Multiple simultaneous accesses to a component must be synchronized.
Counting Proxy. Accidental deletion of components must be prevented or usage statistics

collected.

Virtual Proxy. Processing or loading a component is costly, while partial information about the component may be sufficient.

Firewall Proxy. Local clients should be protected from the outside world.

5.a) While working on its task the client asks the proxy to carry out a service. r The proxy receives the incoming service request and pre-processes it. This pre-processing involves actions such as looking up the address of the original, or checking a local cache to see if the requested information is already available. If the proxy has to consult the original to fulfill the request, it forwards the request to the original using the proper communication protocols and security measures. The original accepts the request and fulfills it. It sends the response back to the proxy. The proxy receives the response. Before or after transferring it to the client it may carry out additional post-processing actions such as caching the result, calling the destructor of the original or releasing a lock on a resource.



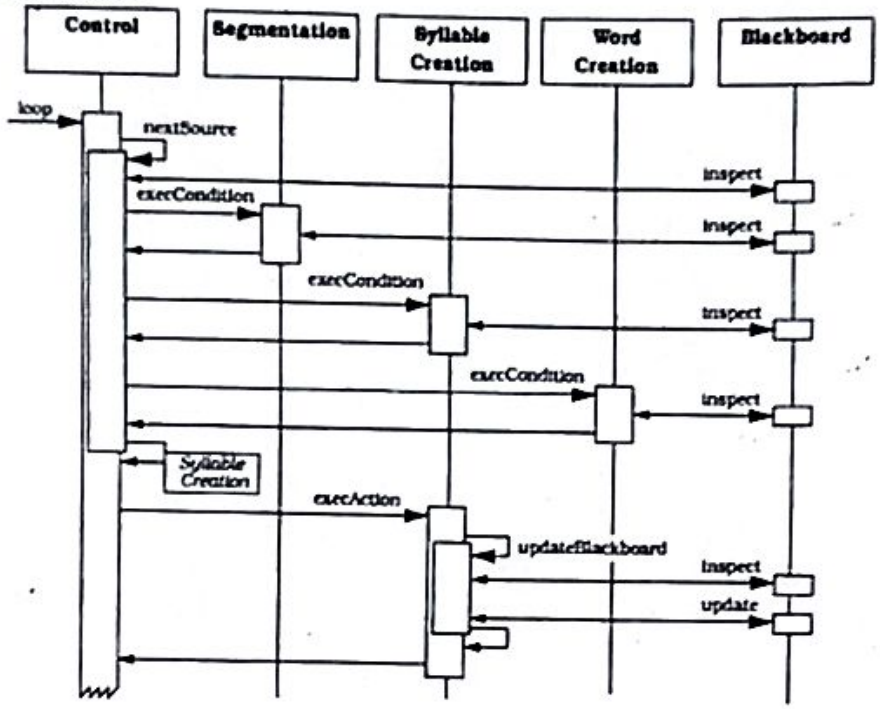
5.b) Divide your system into a component called blackboard, a collection of knowledge sources, and a control component.

Class Blackboard	Collaborators -
Responsibility • Manages central data	

Class Knowledge Source	Collaborator • Blackboard
Responsibility • Evaluates its own applicability • Computes a result • Updates Blackboard	

Class Control	Collaborators • Blackboard • Knowledge Source
Responsibility • Monitors Blackboard • Schedules Knowledge Source activations	

6. a) The main loop of the Control component is started.
Control calls the nextsource () procedure to select the next knowledge source. operates on nextsource () first determines which knowledge sources are potential contributors by observing the blackboard. In this example we assume the candidate lmowledge sources are Segmentation. Syllable Creation and Word Creation. nextsource0 invokes the condition-part of each candidate knowledge source. In the example, the condition-parts of Segrnentation. Syllable Creation and Word Creation inspect the blackboard to determine if and how they can contribute to the current state of the solution. Knowledge Source The Control component chooses a lmowledge source to invoke, and a hypothesis or a set of hypotheses to be worked on.



6.b) **Consequences:**
Experimentation
Support for changeability and maintainability.
Reusable knowledge sources
Support for fault tolerance and robustness.
Liabilities
Difficulty of testing
No good solution is guaranteed
Difficulty of establishing a good control strategy.

16/5/17
Signature of Faculty

Signature of HOD

Eighth Semester B.E. Degree Examination, June/July 2014

Software Architecture

Time: 3 hrs. Max. Marks:100

Note: *Answer FIVE full questions, selecting atleast TWO question from each part.*

PART – A

1 a. With the help of next block diagram of ABC (architecture business cycle). Explain in detail the different activities which are involved in creating a software architecture. (10 Marks)

b. Enumerate and explain in detail, the different groups of software architectures structure are categorized into, with the help of appropriate pictorial description. (10 Marks)

2 a. Explain in brief about KWIC (keyword in context) with shared data solution. (10 Marks)

b. Explain in brief about pipes and filters style, with diagram. (10 Marks)

3 a. Explain what is availability? Explain general scenario for availability? (10 Marks)

b. What do you mean by tactics? Explain availability tactics with neat diagram. (10 Marks)

4 a. What do you mean by architectural pattern? How it is categorized? Explain the structure part of the solution for ISO layered architecture. (10 Marks)

b. Define blackboard architectural pattern? Briefly explain steps used to implement the black board pattern. (10 Marks)

PART – B

5 a. What do you mean by broker architecture? What are the steps involved in implementing distributed broker architecture patterns? (10 Marks)

b. Explain with neat diagram, the dynamic scenarios of model view controller (MVC). (10 Marks)

6 a. What are the steps involved in implementing the microkernel system? (12 Marks)

b. What are the benefits and liabilities of "Reflection architecture". Patterns? (08 Marks)

7 a. Discuss the five steps implementation of Master – slave – pattern. (10 Marks)

b. Explain in brief about variants of whole – part – design pattern, in brief. (10 Marks)

8 a. Briefly explain the different steps performed while designing an architecture using the ADT method. (10 Marks)

b. Write short notes any two of following :

i) Forming team structures

ii) Documenting across views

iii) Documenting interfaces. (10 Marks)

* * * * *

USN | | | | | | | | | | 10IS81

Eighth Semester B.E. Degree Examination, Dec.2014/Jan.2015
Software Architectures

Time: 3 hrs. Max. Marks:100

Note: *Answer FIVE full questions, selecting at least TWO questions from each part.*

PART – A

1 a. What are the factors which affect the influence on software architecture? Explain ABC. (08 Marks)
b. Explain Hybertsson's three views for software architecture. (07 Marks)
c. Explain briefly the properties of a good software architecture design. (05 Marks)

2 a. Define architectural style. Mention any four commonly used types. (05 Marks)
b. Explain the advantages and disadvantages of pipes and filters in architectural style. (08 Marks)
c. Explain the basic requirements for a mobile robot's architecture. (07 Marks)

3 a. What is functionality? Give examples. (04 Marks)
b. Explain the quality attributes scenarios. (09 Marks)
c. Explain how the faults are detected and prevented. (07 Marks)

4 a. Explain the list of components of a pipe and filters and write the problems based on blackboard problem. (08 Marks)
b. Discuss the steps involved in the implementation of pipes and filters architecture. (12 Marks)

PART – B

5 a. What is the need of proxies and bridge components in a broker system? Explain it. (06 Marks)
b. What is broker architecture? Write down the steps involved in implementing distributed broker architecture patterns. (10 Marks)
c. What are the limitations of PAC patterns? (04 Marks)

6 a. List out and explain the components of a microkernel pattern. (10 Marks)
b. Explain the advantages and disadvantages of a reflective architectural pattern. (06 Marks)
c. Mention the liabilities of reflection architecture patterns. (04 Marks)

7 a. Explain the five steps implementation of master-slave pattern. (10 Marks)
b. What are the benefits and liabilities of proxy design patterns? Define proxy design pattern. (10 Marks)

8 a. Draw a neat diagram and explain the evolutionary delivering life cycle model. (10 Marks)
b. What are the steps in ADD? (04 Marks)
c. Write the uses of SA documentation. (06 Marks)

* * * * *

AIET	Lesson Plan & Execution	Format No.	ACD 08
		Issue No.	01
		Rev. No.	00

Name of the faculty	Vineetha Pais
Semester and Section	8th 'B' section
Date of Commencement	13-2-2017
Last Working Day of the Semester	3/6/17
Source Materials List	
1. Len Bass, Paul Clements, Rick Kazman, Software Architecture in Practice	3rd edition
2. Frank Buschmann, Regine Meunier, Hans Rohnert Peter Sommerlad, Michael Stal	Pattern oriented Software Architecture
3. Mary Shaw and David Garlan: Software Perspectives on an emerging	Architecture discipline, PHI 2007
4.	
5.	
Subject Name	

Period	Plan			Execution		
	Date	Topics to be covered	Source Material needed	Topics Covered	Date	Source Material Referred
1	13/2	Unit 1: The Architecture Business Cycle: where do architectures come from?	1	Unit 1: The Architecture Business Cycle where do architectures come from?	13/2	1
2.	14/2	Software process and the architecture business cycle.	1	Software Process and the architecture business cycle	13/2	1
3.	15/2	what make a "good architecture	1	what makes a good architecture	14/2	1
4.	16/2	What Software Architecture is and what it is not	1	What Software Architecture is and what it is not	15/2	1

Period	Plan			Execution		
	Date	Topics to be covered	Source Material needed	Topics Covered	Date	Source Material Referred
5.	17/2	other points of view	1	other points of view	16/2	1
6.	20/2	Architectural patterns reference models	1	Architectural Patterns	17/2	1
7.	21/2	Importance of architecture Architecture Structure and views.	1	reference models	20/2	1
8.	22/2	Unit 2: Architectural Styles Pipe and filters.	3	Importance of Architecture	21/2	1
9.	23/2	Data abstraction and object oriented organization.	3	Architecture Structure and views	22/2	1
10.	27/2	Event based implicit invocation layered Systems	3	Unit 2: Architecture Styles. Pipes & Filters.	23/2	3
11.	28/2	Repositories Interpreters	3	Data abstraction object oriented organisation	28/2	3
12.	1/3	Process control other familiar architectures	3	Event based implicit invocation layered Systems	1/3	3
13.	2/3	Heterogeneous architectures Keyword in Context	3	Repositories Interpreters	2/3	3
14.	3/3	Instrumentation Software Three vignettes in mixed style	3	Process control other familiar architectures	6/3	3
15.	6/3	Unit 3: Functionality & Architecture.	1	Heterogeneous architectures	7/3	3
16.	7/3	Architecture and quality attributes	1	Case Studies Keyword in Context	8/3	3
17.	8/3	Quality attributes Scenarios in practice Business qualities	1	Instrumentation Software	13/3	3

Period	Plan			Execution		
	Date	Topics to be covered	Source Material needed	Topics Covered	Date	Source Material Referred
18.	9/3	Introducing tactics Availability tactics	1	Mobile robotics	14/3	3
19.	10/3	Modifiability tactics Performance tactics Testability tactics	1	Cruise control Mixed style	15/3	3
20.	13/3	Relationship of tactics to architectural tactics	1	Unit 3: Quality: Functionality & architecture	16/3	1
21.	14/3	Unit 4: Architectural Patterns from mud Structure Layers	2	Architecture and quality attributes	20/3	1
22.	15/3	Introduction layered Structure	2	System quality attributes Scenario	21/3	1
23.	16/3	Pipes and filters	2	Business qualities Architecture qualities	22/3	1
24.	17/3	Pipes and filters Continued	2	Introducing tactics Availability tactics	27/3	1
25.	20/3	Black board	2	Modifiability tactics Performance tactics	28/3	1
26.	21/3	black board Continued.	2	Security tactics Testability tactics	30/3.	1
27.	22/3	Unit 5: Architectural Patterns - 2	2	Usability tactics.	31/3	1
28.	27/3	Distributed Systems: Broker	2	Relationship of tactics to architectural pattern	3/4	1
29.	28/3	Continuation: Broker	2	Architectural Patterns and Styles.	4/4	1
30.	30/3	Interactive Systems: MVC.	2	Unit 5: Architecture Patterns - 2	5/4	2

Period	Plan			Execution		
	Date	Topics to be covered	Source Material needed	Topics Covered	Date	Source Material Referred
31.	31/3	Interactive Systm : MVC	2	Distributed Systms	6/4	2
32.	3/4	Presentation	2	Distributed Systm – broker	10/4	2
33.	4/4	Abstraction Control..	2	Interactive Systms – MVC	11/4	2
34.	5/4	Unit 6 : Architectural Patterns – 3	2	MVC Continued	12/4	2
35.	6/4	Adaptable Systms	2	Presentation abstraction Control.	13/4	2
36.	10/4	Adaptable Systms Continued	2	PAC Continued.	17/4	2
37.	11/4	Microkernel Systms	2	Unit 6 : Architectural Patterns – 3	19/4	2
38.	12/4	Microkernel Systms Continued.	2	Adaptable Systms	20/4	2
39.	13/4	Reflection.	2	Microkernal pattern	21/4	2
40.	17/4	Unit 7 : Some Design Patterns	2	Microkernal Continued	24/4	2
41.	18/4	Structural decomposition	2	Reflection Pattern	24/4	2
42.	19/4	Whole – Part Systm	2	Reflection Continued	25/4	2
43.	20/4	Organisation of Work	2	Revision of Reflection	3/5	2

Period	Plan			Execution		
	Date	Topics to be covered	Source Material needed	Topics Covered	Date	Source Material Referred
44.	21/4	Master - Slave	2	Unit 4: Design Patterns	4/5	2
45.	24/4	Access Control: Proxy	2	Structural decomposition	4/5	2
46.	25/4	Unit 3: Architecture in the life cycle.	1	Whole-Part Pattern	5/5	2
47.	26/4	Designing the architecture	1	Organisation of work	8/5	2
48	2/5	Forming the team structure	1	Master slave Pattern	10/5	2
49.	3/5	Creating a Skeletal System	1	Access Control Proxy	11/5	2
50.	8/5	Uses of architectural documentation Views.	1	Proxy Continued	12/5	2
51.	9/5	Choosing the relevant Views	1	Unit 3: Architecture in life cycle.	15/5	1
52.	10/5	Documenting a View. and access Views.	1	Designing the architecture	16/5	1
53.				Forming the team Structure	17/5	1
54.				Creating a Skeletal System	18/5	1
55.				System Skeleton Continued.	19/5	1
56.				Uses of architectural documentation Views	22/5	1

Period	Plan			Execution		
	Date	Topics to be covered	Source Material needed	Topics Covered	Date	Source Material Referred
57.				Choosing the Relevant Views	22/5	1
58.				documenting a View and accross Views	23/5	1
59.				Revision of Patterns.	24/5	1
60.				Revision of Patterns.	25/5	1

Others	Planned	Actual	Remarks :
Special Classes	3	4	
Tutorials	1	1	
Assignments	2	2	
Seminars	2	2	
IA Tests	03	3	
Portions Covered in the entire Semester	8 Units (100%)		

Course Effectiveness

Students Feedback	Good		
Students Responce	Very Good.		
Result	No. of Students AP	No. of Students Passed	% of Result
	43	42	98%

Faculty in Charge

Signature of Principal (& Remarks if any)

HOD's Signature

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI - 574 225

Class : 8th 'B'

Subject : Software Architecture

ATTENDANCE CUM INTERNAL

Subject : Software Architecture

No. of Classes held :

Sl. No.	U.S.N.	Name	Date / Month	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	4AL13CS055	Meghana. Y		1	2	3	4	A	5	6	7	8	9	10	11	12	A	13
2	56	Nalina L N		1	2	3	4	A	5	(A)	A	6	7	8	9	10	11	12
3	57	Naramel R S		1	2	3	4	5	A	(A)	6	A	7	8	9	A	10	11
4	58	Navyashree		1	2	3	4	5	6	7	A	8	9	10	11	12	13	14
5	59	Nerieksha		1	2	3	4	5	6	7	A	(A)	8	9	10	11	12	13
6	62	Pavan Kumar S		1	2	3	4	5	A	(A)	(A)	6	7	8	9	A	(A)	10
7	63	Pavithra		1	2	3	4	5	6	7	8	9	10	11	A	A	12	13
8	65	Prajwal H Shetty		1	2	3	4	5	6	(A)	(A)	(A)	7	8	9	(A)	(A)	10
9	66	Prarthana G M		1	2	3	4	5	6	(A)	(A)	7	8	9	10	11	(A)	12
10	69	Rajatha Mithyantha		1	2	3	4	5	6	7	8	9	10	11	12	13	A	14
11	70	Rakshitha K Shetty		1	2	A	3	4	5	6	(A)	(A)	7	8	9	10	11	12
12	71	Rakshitha Suvarna B		1	2	3	4	5	6	(A)	A	(A)	7	8	9	10	11	12
13	72	Ranjitha		1	2	3	4	5	6	A	7	8	9	10	11	12	13	14
14	73	Ranjitha Naik K		1	2	3	4	5	6	(A)	(A)	7	8	9	10	11	A	12
15	75	Renita D'Souza		1	2	3	4	5	6	A	7	8	9	10	11	12	(A)	13
16	76	SJ Shreyas		1	2	3	4	5	6	7	(A)	(A)	8	9	(A)	(A)	10	11
17	78	Samruddhi N R		1	2	3	(A)	4	5	(A)	(A)	6	7	8	9	10	(A)	11
18	79	Samson Immanuel		1	2	3	4	5	6	7	(A)	(A)	8	9	10	A	11	12
19	80	Sanjana Devadiga		A	A	1	2	3	4	(A)	(A)	5	6	7	8	9	(A)	10
20	81	Senva Deeksha		1	2	3	4	5	6	7	8	A	9	10	11	12	13	14
21	83	Shashikanth		A	A	1	2	3	4	5	6	(A)	7	8	9	10	11	12
22	84	Shehaab		1	2	3	(A)	4	5	(A)	(A)	6	7	8	9	10	(A)	11
23	85	Shetty harshitha		1	2	3	4	5	6	7	8	(A)	9	10	A	11	12	13
24	86	Shetty Kavya		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
25	87	Shetty Nithesh		1	2	3	4	5	(A)	6	(A)	(A)	7	8	9	(A)	10	11
26	89	Shetty Shreyas		1	2	3	4	5	6	7	(A)	8	9	(A)	10	(A)	A	11
27	92	Shree Lakshmi DT		1	2	3	4	5	6	A	A	7	8	9	10	11	A	12
28	93	Shurthi Surendran		1	2	3	4	5	6	(A)	(A)	7	8	9	10	A	A	11
29	94	Shwetha B		1	2	3	4	5	6	(A)	A	7	8	9	10	11	A	12
30	96	Soneya George		1	2	3	4	5	6	(A)	(A)	7	8	9	10	11	12	13
Staff Initials																		

16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
14	15	A	16	A	17	18	19	20	21	22	(A)	23	24	25
13	14	A	A	15	16	17	18	(A)	19	20	21	22	23	24
12	13	14	(A)	15	16	17	18	(A)	19	20	(A)	21	22	A
15	16	17	A	A	18	19	20	21	22	23	(A)	24	25	A
14	15	16	(A)	17	18	19	20	21	22	23	(A)	24	25	26
10	11	(A)	(A)	12	13	14	15	(A)	A	16	(A)	17	18	(A)
14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
11	12	13	(A)	14	A	15	21	(A)	22	23	A	24	25	26
12	13	14	15	16	17	18	19	(A)	20	A	(A)	21	22	(A)
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
12	13	4	(A)	15	16	17	18	(A)	19	20	(A)	21	22	23
12	A	15	16	17	18	19	(A)	20	21	(A)	22	A	A	
15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
13	14	15	16	17	18	19	20	(A)	21	22	(A)	23	24	A
14	15	16	17	18	19	20	21	(A)	(A)	22	23	24	25	26
12	13	14	15	16	17	18	19	(A)	(A)	20	(A)	21	22	23
12	14	(A)	(A)	15	16	17	18	19	20	(A)	21	22	23	(A)
13	14	15	16	17	18	19	20	21	22	23	A	24	25	26
11	12	(A)	13	14	15	16	(A)	17	18	(A)	19	20	(A)	
12	13	14	15	16	17	18	19	20	A	21	22	23	24	25	26	27	(A)
13	14	15	16	(A)	17	18	19	20	21	22	23	24	25	26
12	13	14	(A)	15	16	17	18	(A)	19	20	(A)	21	22	23
12	13	(A)	14	15	(A)	16	17	(A)	(A)	18	19	20	(A)	(A)
13	14	15	A	16	17	18	19	(A)	20	21	(A)	22	23	A
12	13	14	(A)	15	16	17	18	(A)	19	20	21	22	23	(A)
13	14	15	A	16	17	18	19	(A)	20	A	A	21	22	A
14	15	A	16	17	18	19	20	21	22	23	(A)	24	25	26	27

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI - 574 225

Class : 8th 'B'

Subject : Software Architecture

No. of Classes held :

Sl No.	USN	Name	1	2
1	4AU13CS055	Meghana. Y	1	2
2	56	Nalina L N	1	2
3	57	Nayana R S	1	2
4	58	Navyashree	1	2
5	59	Nireksha	1	2
6	62	Pavan Kumar S	1	2
7	63	Parthiba	1	2
8	65	Prajwal H Shetty	1	2
9	66	Preethana G M	1	2
10	69	Rajatha Hithyantha	1	2
11	70	Rakshetha K Shetty	1	2
12	71	Rakshitha Suvarna B	1	2
13	72	Ranjitha	1	2
14	73	Ranjitha Naik K	1	2
15	75	Renita D'Souza	1	2
16	76	SJ Shreyas	1	2
17	78	Samruddhi N R	1	2
18	79	Samson Immanuel	1	2
19	80	Sanjana Devadiga	A	A
20	81	Senva Deeksha	1	2
21	83	Shashikanth	A	A
22	84	Shehaab	1	2
23	85	Shetty harshitha	1	2
24	86	Shetty Kavya	1	2
25	87	Shetty Nithesh	1	2
26	89	Shetty Shreyas	1	2
27	92	Shreelakshmi DT	1	2
28	93	Shruthi Surendran	1	2
29	94	Shwetha B	1	2
30	96	Soneya George	1	2
Staff Initials				

Class : 8th 'B'

Subject : Software Architecture.

Grace attendance upto 2/6/17

Students Attendence Conducted 48	60	No of Class Attended	% of Attendance	Internal Assessment (25) I	II	III	Average Marks
		58	85	AB	17	24	21
		58	85	21	AB	25	23
		59	87	17	15	24	21
				15	18	AB	17
		61	89	13	14	21	18
		58	85	17	01	13	15
		63	92	23	21	25	24
		63	92	9	11	22	17
		65	95	17	AB	16	17
		65	95	21	14	24	23
		67	98	9	9	20	15
		58	85	17	19	24	22
		64	94	20	17	23	22
		64	94	25	AB	25	25
		62	91	17	15	24	21
		67	98	22	14	25	24
		66	97	14	15	24	20
		65	95	18	16	23	21
		59	87	16	AB	25	21
		64	94	18	24	25	25
		61	89	21	14	AB	18
		65	95	06	11	18	15
		66	97	12	18	17	18
		64	94	25	19	25	25
		60	97	19	AB	18	19
		58	85	16	10	24	20
		59	87	9	8	22	16
		64	94	13	AB	16	15
		60	88	23	5	17	15
		65	95	23	24	AB	24

ALVA'S INSTITUTE OF ENGINEERING & TECHNOLOGY

MIJAR, MOODBIDRI - 574 225

ATTENDANCE CUM INTERNAL

Class : 8th 'B'

Subject : Software Architecture

Subject :

No. of Classes Held :

Sl. No.	U.S.N.	Name	Date / Month 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	97	Soorya K	1	2	3	4	5	6	A	A	7	8	9	10	A	A	11	13	14	A	15	16	17	18	L	19	20	21	22	23	A
2	98	Sredevi	1	2	3	A	4	5	A	A	A	6	7	8	9	10	11	13	14	A	15	16	17	18	A	19	20	A	A	A	A
3	99	Sridhar T S	1	2	3	4	5	6	7	8	A	9	10	11	12	13	14	16	17	18	19	20	21	22	23	24	A	25	26	27	28
4	100	Sruthin R	1	2	3	4	5	6	A	A	7	8	9	10	A	A	11	A	A	A	A	13	14	15	A	16	17	A	18	19	20
5	103	Sukanya	1	2	3	4	5	6	A	A	A	7	8	9	10	A	11	13	14	A	15	16	17	18	A	19	20	A	21	22	A
6	104	Sumalatha N	1	2	3	4	5	6	7	A	8	9	10	11	12	13	14	16	17	A	18	19	20	21	22	23	24	A	25	26	A
7	106	Supretha K N	1	2	3	A	A	A	A	A	A	4	5	6	7	A	8	10	11	A	12	13	14	15	A	A	A	16	17	18	A
8	107	Sruthy T P	1	2	3	4	5	6	A	A	A	7	8	9	10	A	A	A	12	A	13	14	15	16	A	A	27	A	28	29	A
9	108	Tiben K Tomy	1	2	3	A	4	5	A	A	6	7	8	9	A	A	10	A	12	A	13	A	14	15	A	16	17	A	18	A	A
10	110	Vandana G	1	2	3	4	5	6	A	A	A	7	A	8	9	A	A	11	12	13	14	A	15	16	A	A	A	A	17	18	A
11	117	Rohet Rajan	1	2	3	4	5	6	7	A	8	9	10	11	12	13	14	16	17	18	19	20	21	A	A	A	A	A	A	22	23
12	401	Videtha	1	2	3	4	5	6	7	A	8	9	10	A	11	A	12	14	15	16	17	18	19	20	21	22	23	24	25	26	A
13	14CS400	Karthik C Shekar	1	2	3	A	4	5	A	A	6	7	8	A	9	A	10	12	13	A	14	15	16	17	A	18	A	A	19	20	A
14	14CS404	Shetty Rohan	1	2	3	4	5	6	7	8	9	10	A	11	12	A	13	15	16	A	17	18	19	20	A	21	22	A	23	24	A
15	18CS064	Pooja																				1	2	3	4	5	6	7	A	8	A
16	14CS402	Rajesh C N																				1	2	3	4	5	6	A	7	8	9
17		Bharath																													
18																															
19																															
20																															
21																															
22																															
23																															
24																															
25																															
26																															
27																															
28																															
29																															
30																															
Staff Initials																															

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI - 574 225

Class : 8th 'B'

Subject : Software Architecture

No. of Classes held :

Sl. No.	U.S.N.	Name	Date / Month 13/2	14/2
			1	2
1	97	Soorya K	1	2
2	98	Sredevi	1	2
3	99	Sridhar T S	1	2
4	100	Sruthin R	1	2
5	103	Sukanya	1	2
6	104	Sumalatha N	1	2
7	106	Supretha K N	1	2
8	107	Srutha T P	1	2
9	108	Tiben K Tomy	1	2
10	110	Vinutana G	1	2
11	117	Rohit Rajan	1	2
12	401	Vidutha	1	2
13	14CS400	Karthik C Shekar	1	2
14	14CS404	Shetty Rohan	1	2
15	13CS064	Pooja		
16	14CS402	Rajesh C N		
17		Bharath		
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
Staff Initials				

Students Attendence		out of 68			Internal Assessment (25)			Average Marks
Conducted		No. of Class Attended	% of Attendance		I	II	III	
48	60							
		62	91		15	AB	14	15
		61	89		11	15	19	17
		63	92		15	7	24	20
		58	85		10	6	20	15
		58	85		22	19	25	24
		63	92		15	13	20	18
		62	91		18	19	15	19
		64	94		17	AB	12	15
		58	85		15	3	14	15
		63	92		16	7	21	19
		65	95		8	24	25	25
		63	92		10	7	19	15
		58	85		15	AB	14	15
		60	88		12	7	21	17
		58	85		24	AB	5	15
		58	85		15	6	17	16
		58	85		20	AB	9	15

AIET	ASSIGNMENTS	Format No.	ACD 10
		Issue No.	01
		Rev. No.	00
Department	CSE	Academic Year	
		Semester	VIII

Sl No.	Title	Books / Journals / Magazines referred	Date of Announcement	Date of Submission	Signature of the faculty
1.	Unit 1 & 2	Software Architecture	28/2/17	11/3/17	
		Len Bass, Paul			
		clements.			
2.	Assignment-	Pattern Oriented	19/4/17	8/5/17	
	Questions Unit-6	Software			
	and Unit 7	Architecture by			
		Frank Buschmann			

AIET	INTERNAL EXAM RESULT ANALYSIS	Format No.	ACD 12
		Issue No.	01
		Rev. No.	00
Department	Computer Science & Engineering	Semester	VIII
		Subject Code	101581
Total No. of Students	46	Academic Year	2016-17

Test	Date	Number of Students				Signature		Remarks
		Attended	0-14	15-20	21-25	Faculty	HOD	
T_1	23.3.17	42	14	20	8			
T_2	27/4/17	36	18	14	4			
T_3	30/5/17	44	7	13	24			
T_4								
T_5								

Signature of Staff in - charge

HOD's Signature

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI - 574 225

Class : VIII th "A" sec C

Subject : Software Architectures

ATTENDANCE CUM INTERNAL

No. of Classes held : 69

Sl. No.	U.S.N.	Name	Date / Month	13/2	14/2	15/2	16/2	17/2	20/2	21/2	22/2	23/2	27/2	28/2	1/3	2/3	3/3
				1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	12CS003	Aishwarya Ramesh		P	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	12CS009	Anoop Naik		P	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3	12CS010	Anusha Ajith		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	12CS036	Gireesh Mon. P.M.		1	2	3	4	5	6	7	8	A	A	A	A	A	A	9
5	12CS048	Manisha K.P		P	2	3	4	5	6	7	8	9	10	A	11	12	13	14
6	12CS062	Payal M.P Shetty		1	2	3	4	5	6	7	8	9	10	A	11	12	13	14
7	13CS001	Adarsh Ajith		P	2	3	4	5	6	7	8	9	10	11	12	A	13	14
8	13CS002	Adhiel Hussain		1	2	3	4	5	6	7	8	9	10	A	11	12	13	14
9	13CS003	Aishwarya K Shetty		1	2	3	4	5	6	7	8	9	10	A	11	12	13	14
10	13CS004	Akash G		P	2	3	4	5	6	7	8	9	10	11	12	13	14	15
11	13CS005	Akshatha Bhat		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	13CS007	Amina Sajiaha		1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
13	13CS008	Aniketan Noel		1	2	3	4	5	6	7	8	9	10	A	11	12	13	14
~~14~~	~~13CS010~~	~~Anusha~~		~~1~~	~~2~~	~~3~~	~~4~~	~~5~~	~~6~~	~~7~~	~~8~~	~~9~~	~~10~~	~~11~~	~~12~~	~~13~~	~~14~~	
15	13CS011	Anusha S Aradhya		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	13CS012	Anusha V		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
17	13CS013	Anvitha Shubhavee Jain		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
18	13CS015	Arpita S Bhat		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
19	13CS016	Arya K M		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
20	13CS018	Asif		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
21	13CS019	B Asritha		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
22	13CS020	Bhandary Shreyas Shaka		A	A	1	2	3	4	5	6	7	A	8	9	10	11	12
23	13CS021	Bharath R		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
24	13CS022	Bindu S		1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
25	13CS024	Darshana K		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
26	13CS025	Deeksha Shetty B		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
27	13CS026	Deeksha Shetty D		1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
28	13CS027	Deena Deepika Cutinha		1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
29	13CS028	Deepak D K		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
30	13CS029	Deepika B Shetty		A	A	1	2	3	4	5	6	7	8	9	10	11	12	13
Staff Initials																		

Class : VIII th "A" sec

Subject : Softwar Arrchitechure

No. of Classes held : 69

Sl. No.	U.S.N. 4AL	Name	1	2	No. of Class Attended	% of Attendance	Internal Assessment (25) I	II	III	Average Marks
1	12CS003	Aishwarya Ramesh	1	2	65	95	06	02	09	08
2	12CS009	Anoop Naik	1	2	64	93	15	07	18	17
3	12CS010	Anusha Ajith	1	2	64	93	11	03	18	15
4	12CS036	Gireesh Mon. P.M.	1	2	62	90	11	03	21	16
5	12CS048	Manisha K.P	1	2	64	93	06	04	17	12
6	12CS062	Payal M.P Shetty	1	2	66	96	13	AB	20	17
7	13CS001	Adarsh Ajith	1	2	64	93	09	06	20	15
8	13CS002	Adhiel Hussain	1	2	62	90	AB	07	22	15
9	13CS003	Aishwarya K Shetty	1	2	66	96	15	09	20	18
10	13CS004	Akash G	1	2	65	95	09	AB	23	16
11	13CS005	Akshatha Bhat	1	2	65	95	13	11	24	19
12	13CS007	Amina Sajiaha	1	2	63	92	21	AB	22	22
13	13CS008	Aniketan Noel	1	2	66	96	13	11	23	18
~~14~~	~~13CS010~~	~~Anusha~~	1	2			AB			
15	13CS011	Anusha S Aradhya	1	2	65	95	15	11	24	20
16	13CS012	Anusha V	1	2	66	96	17	15	25	21
17	13CS013	Anvitha Shuthree Jain	1	2	65	95	19	AB	24	22
18	13CS015	Arpita S Bhat	1	2	64	93	15	18	25	22
19	13CS016	Arya K M	1	2	64	93	18	11	24	21
20	13CS018	Asif	1	2	65	95	10	06	24	17
21	13CS019	B Asritha	1	2	60	87	12	12	24	18
22	13CS020	Bhandary Shreyas Shankar	A	A	60	87	15	03	17	16
23	13CS021	Bharath R	1	2	64	93	07	07	25	16
24	13CS022	Bindu S	1	2	65	95	20	23	25	24
25	13CS024	Darshana K	1	2	66	96	12	AB	17	15
26	13CS025	Deeksha Shetty B	1	2	66	96	18	11	25	22
27	13CS026	Deeksha Shetty D	1	2	63	92	15	03	25	20
28	13CS027	Dena Deepika Cutinha	1	2	64	93	16	11	17	17
29	13CS028	Deepak D K	1	2	68	99	15	03	22	19
30	13CS029	Deepika B Shetty	A	A	62	90	17	AB	25	21
Staff Initials										

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

MIJAR, MOODBIDRI - 574 225

Class : VIII th 'A' Sec

Subject : Software Architectures

ATTENDANCE CUM INTERNAL

No. of Classes held : 69

Sl. No.	USN 4AL	Name	13/1	13/1	14/1	15/1	16/1	17/1	20/1	21/1	22/1	23/1	27/1	28/1	1/3	3/3	3/3
		Date / Month	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
31	13CS030	Devipoosad S Shetty	1	2	3	4	5	6	7	8	9	10	11	A	12	13	14
32	13CS031	Dileep Kumar.	1	2	3	4	5	6	7	8	9	10	11	12	13	A	14
33	13CS032	Divya S Babu	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
34	13CS034	Gaurav Suvarna	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
35	13CS035	Gautami Prakash Hegde	1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
36	13CS036	Gowrava	A	A	1	2	3	4	5	6	7	8	9	10	A	11	12
37	13CS037	Gururaj Hipparagi	A	1	2	3	4	5	6	7	8	9	10	11	12	13	14
38	13CS038	Husmitha B V	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
39	13CS039	Hellen Treesa John	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
40	13CS040	Hitesh Harikrishnan	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
41	13CS041	Impana M.S	1	2	3	4	5	A	6	7	8	9	10	11	12	13	14
42	13CS043	Jeain Jose	A	A	1	2	3	4	5	6	7	8	A	9	10	11	12
43	13CS044	Jisha Emmanuel	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
44	13CS045	Jisha Jayaraj	1	2	3	4	5	6	7	8	9	10	A	11	12	13	14
45	13CS047	Kavya	1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
46	13CS048	Kusuma J	A	A	1	2	3	4	5	6	7	8	9	A	10	11	12
47	13CS049	Lathishma Suvarna M	1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
48	13CS050	Lavanya R	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
49	13CS051	M Raksha Kudva	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
50	13CS052	M Sangeetha Jain	1	2	3	4	5	6	7	8	9	A	10	11	12	13	14
51	13CS053	Maheshakumar Shetty	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
52	13CS054	Manoj E.	1	2	3	4	5	6	7	8	9	10	A	11	12	13	14
53	13CS055	Anjana Jose	← Crash Semester Student →														
24																	
25																	
26																	
27																	
28																	
29																	
30																	
Staff Initials																	

ALVA'S INSTITUTE OF ENGINEERING AND TECHNOLOGY

ALV

MOODBIDRI - 574 225

Class : VIII th 'A' sec

Subject : Software Architect

No. of Classes held : 69

Sl. No.	USN	Name	13/1	14/1	16
	4AL	Date / Month	1	2	3
31	13CS030	Devipoosad S Shetty	1	2	3
32	13CS031	Dileep Kumar.	1	2	3
33	13CS032	Divya S Babu	1	2	3
34	13CS034	Gaurav Suvarna	1	2	3
35	13CS035	Gautami Prakash Hegde	1	2	3
36	13CS036	Gowrava	A	A	1
37	13CS037	Gururaj Hipparagi	A	1	2
38	13CS038	Hasmitha B V	1	2	3
39	13CS039	Hellen Treesa John	1	2	3
40	13CS040	Hitesh Harikrishnan	1	2	3
41	13CS041	Impana M S	1	2	3
42	13CS043	Jesin Jose	A	A	1
43	13CS044	Jisha Emmanuel	1	2	3
44	13CS045	Jisha Jayaraj	1	2	3
45	13CS047	Kavya	1	2	3
46	13CS048	Kusuma J.	A	A	1
47	13CS049	Lathishma Suvarna M	1	2	3
48	13CS050	Lavanya R	1	2	3
49	13CS051	M Raksha Kudva	1	2	3
50	13CS052	M Sangeetha Jain	1	2	3
51	13CS053	Mahabalakumar Shetty	1	2	3
52	13CS054	Mandj E.	1	2	3
53	13CS055	Anjana Jose	← Crct		
24					
25					
26					
27					
28					
29					
30					
Staff Initials					

Students Attendence Inducted 48	60	No. of Class Attended	% of Attendance	Internal Assessment (25) I	II	III	Average Marks
		69					
		63	92	22	19	25	24
		68	99	22	07	17	15
		66	96	13	AB	20	17
		64	93	07	AB	13	10
		65	95	21	20	25	23
		62	90	15	08	22	19
		68	99	06	07	22	15
		65	95	20	AB	24	22
		61	89	09	06	16	13
		65	95	15	04	18	17
		63	92	15	09	21	18
		60	87	06	06	13	10
		67	98	20	15	AB	18
		66	96	12	10	20	16
		67	98	17	09	24	21
		62	90	12	AB	25	19
		67	98	23	16	25	24
		67	98	13	09	23	18
		69	100	25	18	25	25
		67	98	12	08	17	15
		65	95	17	07	24	21
		61	89	10	06	20	15
		37	93	22	07	07	15