4/4 B.Tech. FIRST SEMESTER

CS7T1 **SOFTWARE ARCHITECTURE** **Credits: 4**

(Large-Scale C++ Software Design)

Required

Lecture: 4 periods/week **Internal assessment: 30 marks**

Tutorial: 1 period /week **Semester end examination: 70 marks**

Course Context and Overview: Software Architecture and Design teaches the principles and concepts involved in theweb analysis and design of large software systems.

Prerequisite: Software Engineering, C++

Objectives:

By the end of this course, you should be able to:

1. Able to explain the process of decomposing large systems into physical hierarchies of smaller, more manageable components.
2. Able to identify the need for following good physical as well as logical design practices,
3. Learn specific techniques designed to eliminate cyclic, compile-time, and link-time dependencies.
4. Able to explain the top-down approach to the logical design of individual components.
5. Understand different functions for input and output, various data types, basic operators, files and functions.

Learning Outcomes

1. Able to explain the process of decomposing large systems into physical hierarchies of smaller, more manageable components.
2. Able to identify the need for following good physical as well as logical design practices.
3. Learn specific techniques designed to eliminate cyclic, compile-time, and link-time dependencies.
4. Able to explain the top – down approach to the logical design of individual components

UNIT I

Introduction

From C to C++

Using C++ to Develop Large Projects:

Cyclic Dependencies, Excessive Link Time Dependencies, Excessive Compile time Dependencies, The global Name Space, and Logical Vs Physical Design

Reuse, **Quality:** Quality Assurance, Quality Ensurance

Software Development Tools.

Preliminaries

Multi-File C++ Programs: Declaration versus Definition, Internal versus External Linkage,Header Files, and Implementation (.c) Files

Typedef Declarations, **Assert Statements**, **A Few Matters of Style: Identifier Names, Class Member Layout**, Iterators**,**

Logical Design Notation: The IsA Relation, The Uses-In the Interface relation, the uses inthe Implementation Relation

Inheritance versus Layering, Minimality

UNIT II

Ground Rules: Overview, Member **Data Access**
The Global Name Space:

Global Data, Free functions, Typedefs and Constant data, preprocess of Macros. Names in header files

Include Guards, **Redundant Include Guards, Documentation, Identifier Naming Conventions**.

UNIT III

Components. Components versus Classes, Physical Design Rules, The Depends On Relation, Implied Dependency, Extracting Actual Dependencies, Friendship:
Long Distance Friendship and Implied Dependency , Friendship and Fraud

UNIT IV

Physical Hierarchy:
A Metaphor for Software Testing, A Complex Subsystem, The Difficulty in Testing "Good" Interfaces, Design for Testability, Testing in Isolation, Acyclic Physical Dependencies, Level Numbers, Hierarchical and Incremental Testing, Testing a Complex Subsystem, Testing versus Tested, Cyclic Physical Dependencies, Cumulative Component Dependency (CCD), Physical Design Quality.

UNIT V

Levelization. Some Causes of Cyclic Physical Dependencies:Enhancements, conveniences, Intrinsic Interdependency

Escalation, Demotion, Opaque Pointers, Dumb Data, Redundancy, Callbacks, Manager Class, Factoring, Escalating Encapsulation

UNIT VI

Insulation:

From Encapsulation to Insulation: The cost of compile time coupling **C++ Constructs and Compile-Time Coupling**:

Inheritance(IsA) and compile time coupling, Layering (hasA/holdsA) and compile time coupling, inline functions and compile time coupling, private members compile time coupling , protected member compile time coupling , compiler generated member functions and compile time coupling , include directive and compile time coupling , default arguments and compile time coupling , enumerations and compile time coupling

UNIT VII

Partial Insulation Techniques:

Removing private inheritance , removing private member functions , removing embedded data members , removing private members functions , removing protected members , removing private member data , removing compile generated functions , removing include directives, removing default arguments

Total Insulation Techniques:
The protocol class, the fully insulating concrete class, the insulating wrapper

UNIT VIII

The procedural interface:

The procedural interface architecture, creating and destroying opaque objects, inheritance and opaque objects

To Insulate or Not to Insulate: the cost of insulation, when not to insulate, how to insulate, how much to insulate.

Learning Resources

TEXT BOOK:

Large-Scale C++ Software Design, John Lakos,1997, Addison-Wesley Professional

REFERENCE BOOKS:

Bjarne Stroustrup, "The C++ Programming Language", Third Edition, Pearson Education.