

Located on the Tk’emlups te Secwepemc territory within the unceded traditional lands of Secwepmecul’ecw (Secwepemc Nation)

**COMP 2920:**  **Software Architecture & Design (3,1,0)**

**Fall 2017**

Instructor: Mridula Sharma Phone/Voice Mail: (250)8527149

Office: HL 407 E-Mail: msharma@tru.ca

Office Hours: by appointment

**Calendar /Course Description**

Students learn how to establish, define and manage the requirements for a software system. Students gain knowledge of fundamental concepts and methods of software design. Students learn how to use design notations of unified modeling language to develop design of a software product. Students are introduced to the design guidelines, quality, and evaluation criteria of software architecture. Students study how to design, generate, and modify software patterns and their use in software development. **Course/Learning Outcomes**

Upon successful completion of the course, the student will demonstrate the ability to:

* Understand the Concept of Software Design
* Explain the various models of Software development
* Explain the need and role of Software Architecture
* Identify both functional and non-functional requirements of a software system.
* Explain and develop Object Oriented design
* Design a simple software architecture using UML modelling.
* Construct various models of the design of a simple software system.
* Articulate and evaluate overall architecture of a simple system.
* Understand the intellectual property of a software design.

**Pre-requisite: COMP 1230 a minimum of “C”**

**Text Book (Recommended):**

|  |
| --- |
| 1. Title: Software Modeling & Design: UML, Use Cases, Patterns, and Software Architectures   Author: Hassan Gomaa   1. Object-Oriented Modeling and Design with UML (2nd Edition) Author: Michael R. Blaha and James R Rumbaugh |

1. UML Distilled : A Brief Guide to the Standard Object Modeling Language by Martin Fowler
2. UML 2.0 in action – A project based tutorial by Patrick Grassle PACKT publishing
3. Ethics for the Information Age (6th Edition) Michael J. Quinn, Addison-Wesley

**Syllabus:**

**Topics**

* Introduction to software Design & Software development process models
* Need and role of Software Architecture (Architecture Views, Context of software architecture in business, technical, stakeholders, Designing an architecture, Quality attributes of software architecture, Architecture Evaluation)
* Design Guidelines Developing Systems (Addressing design goals, decomposing systems, top - down approach for dynamic systems, bottom - up approach for dynamic systems, flexibility guidelines for behavioral design)
* Requirements Elicitation Concepts & Activities, Requirements Types
* Object Oriented Concepts – using UML to represent different models
* Modeling Requirements (Use Case Diagram) -Relationships between use cases - extend, include, and generalize.
* Static Structural Modeling (Class diagram, diagram notations and modeling, relations among objects and examples, mapping use cases to classes, relationships among classes, Associations, Dependencies., Inheritance - Generalizations, Aggregation)
* Interaction Modeling (Sequence Diagram) (Sequence diagram notations and examples, iterations, conditional messaging, branching, object creation and destruction, time constraints, origin of links, Activations in sequence diagram)
* Interaction Modeling (Collaboration Diagram) (Collaboration diagram notations and examples, iterations, conditional messaging, branching
* Activity Diagram (Action State, Activity State, Object Node, Control and Object Flow, Transition, Guidelines for Creating Activity Diagrams, Action Decomposition, Partition)
* Behavioral modelling using State Diagram
* Component & Deployment Diagram - package design
* Intellectual Property
* Demonstrate 4+1 view of a software

**Practice/Assessment Topics for lab:**

* Working with UML tool - ArgoUML
* Use Case Diagram Exercises
* Implementing extend, include, and generalize relations Exercises
* Activity Diagram Exercises
* Sequence Diagram Exercises
* Collaboration Diagram Exercises
* Class Diagram Exercises
* State Diagram Exercises
* Component, Deployment Diagram Exercises
* 4+1 model development

To aid the student in successfully completing the course, each student:

* requires a TRU network account, to save files on the network and access WebCT;
* is given a personal account on the Computing Science Linux server to complete C programming on UNIX (this is separate from the TRU network)—*students are not required to have Linux installed at home*
* Reference site C- Programming: <http://www2.its.strath.ac.uk/courses/c/>

**Student Evaluation**

Labs, Quizzes & Assignments 20%

Mid-term Exam 25%(18th October)

Final Written Exam 40%

Project 15%

**Notes on Course Success**:

To pass the course the student must achieve a minimum of 50% overall. Also, the student must achieve

* minimum of 50% on overall quizzes & Assignments, and
* minimum of 50% on the programming exam and final written exam

**Notes on Submissions**:

* Submissions are expected up to the due date, but not after. Late submissions are not considered and receive a mark of zero.
* Any copied work will get negative marks. For example, when an assignment is given with ***n*** marks, –***n*** marks will be given to any copied work.
* The plagiarism will be reported according to the Academic Integrity Policy.

Notes on Attendance:

Students are expected to attend every lecture and seminar—material is sometimes presented that is not in the text. Obtaining the content and activities for any missed times are the responsibility of the student.

Notes on Academic Integrity Policy:

All assessments and tests given for computing courses are governed by the Academic Integrity Policy of both the TRU and the Computing Science Department. Please consult the following online document (presented in two forms).

* <http://www.tru.ca/policy/ed/ed05-0.pdf>

***Seminars*:**

The purpose of seminar periods is to either review specific lecture topics, or provide discussion outside regular material. Seminar time may consist of in-class or take-home exercises, research discussion, or preparation for lab activities. During seminar periods, students are given time to put into practise some of the ideas discussed in lecture.

**Special Course Activities**

**Methods for Prior Learning Assessment and Recognition**

Students may receive credit for a similar course, taken at another institution, with the written consent from one of: COMP 2920 Instructor, Computing Science Dept. Chairperson, and BCS Advisor or B.Sc. (Major in Comp. Sc.) Advisor.

**Use of Technology**

The tools and support systems in the course: the Linux operating system and C programming language (with GNU GCC), TRULAN (Local Area Network), and World Wide Web.