THOMPSON RIVERS UNIVERSITY

Course Outline

Department of Computing Science

Faculty of Science

COMP 2920 – 3
Software Architecture & Design (3,1,0)
Fall, 2015

Instructor: Phone/Voice Mail: Office: E-Mail: Office Hours:

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:

1. Identify both functional and non-functional requirements of a software system.
2. Use a design paradigm to design a simple software system.
3. Construct models of the design of a simple software system.
4. Articulate and evaluate software architecture of a simple system.
5. Apply simple examples of patterns in a software design.
6. Understand the intellectual property of a software design.

Pre-requisite: COMP 1230 a minimum of "C"

Text Book:

1. Object-Oriented Modeling and Design with UML, Michael R. Blaha, James R Rumbaugh, ISBN-13: 978-0130159205.
2. Software Architecture in Practice (3rd Edition), Len Bass, Paul Clements, Rick Kazman, ISBN-13: 978-0321815736.
3. Reference Book (R-1): Ethics for the Information Age (6th Edition) Michael J. Quinn, Addison-Wesley

Syllabus

Topics	**Week**	**Book**
Overview of Prominent OO Methodologies (The Rumbaugh OMT, The Booch methodology, Jacobson's OOSE methodologies, Unified Process, Introduction to UML) **Intellectual Property** issues and concerns with software design	1	1
Requirements Elicitation Concepts, Requirements Elicitation Activities, Requirements Types **Modeling Requirements (Use Case Diagram)**	2	2
Relationships between use cases - extend, include, and generalize. **Activity Diagram (**Action State, Activity State, Object Node, Control and Object Flow, Transition, Guidelines for Creating Activity Diagrams, Action Decomposition, Partition)	3	2
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)	4	3
Interaction Modeling (Collaboration Diagram) (Collaboration diagram notations and examples, iterations, conditional messaging, branching, object creation and destruction, time constraints, origin of links, activations in sequence diagram.	5	3
Static Structural Modeling (Class diagram, diagram notations and modeling, relations among objects and examples, mapping use cases to classes, relationships among classes)	6	4
Class Modeling and Design Approaches (System decomposition, Guidelines for designing class diagram, Cohesion, Coupling, Forms of coupling (identity, representational, subclass, inheritance), Associations, Dependencies., Inheritance - Generalizations, Aggregation)	7	4
Behavioral (Dynamic Structural Modeling) State Diagram (State Diagram Notations, events (signal events, change events, Time events). b. State Diagram states (composite states, parallel states, History states), transition and condition, state diagram behavior(activity effect, do-activity, entry and exit activity), completion transition, sending signals)	8	5
Component & Deployment Diagram (Component and Interface design, notations and modeling, Deployment diagram of the system, understanding package design)	9	

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)	10	6
Software Architecture (Architecture Views, Context of software architecture in business, technical, stakeholders, Designing an architecture, Quality attributes of software architecture, Architecture Evaluation)	11	B-2(1,3, 21)
Design Patterns & Frameworks (Creational Patterns, Structural Patterns, Behavioral Patterns, Reuse of frameworks, black box framework, white box frame works)	12	Instructor Notes
Intellectual Property	13	R-1 (4)

Lab Topics:

Topics	Week
Introduction to UML Tool	1
Use Case Diagram Exercises	2
Implementing extend, include, and generalize relations Exercises	4
Activity Diagram Exercises	4
Sequence Diagram Exercises	5
Collaboration Diagram Exercises	6
Class Diagram Exercises	7
State Diagram Exercises	8
Component, Deployment Diagram Exercises	9
Creational Patterns Exercises	10
Structural Patterns Exercises	11
Behavioral Patterns Exercises	12

ACM / IEEE Knowledge Area Coverage

IEEE Knowledge Areas that contain topics and learning outcomes covered in the course

Knowledge Area	**Total Hours of Coverage**
SE/Requirements Engineering	4
SE/Software Design	8
SP/ Intellectual Property	2

IEEE Body of Knowledge coverage

KA	**Knowledge Unit**	**Topics Covered**	**T1 hours**	**T2 hours**	**Elective hours**
	SE/Requirements Engineering	Describing functional requirements using, for example, use cases or users stories • Properties of requirements including consistency, validity, completeness, and feasibility Software requirements elicitation • Describing system data using, for example, class diagrams or entity-relationship diagrams • Non-functional requirements and their relationship to software quality (cross-reference IAS/Secure Software Engineering) • Evaluation and use of requirements specifications	**1**	**3**	
	SE/Software Design	System design principles: levels of abstraction (architectural design and detailed design), separation of concerns, information hiding, coupling and cohesion, re-use of standard structures • Design Paradigms such as structured design (top-down functional decomposition), object-oriented analysis and design, event driven design, component-level design, data-structured centered, aspect oriented,	**3**	**5**	

		function oriented, service oriented • Structural and behavioral models of software designs • Design patterns Relationships between requirements and designs: transformation of models, design of contracts, invariants • Software architecture concepts and standard architectures (e.g. client-server, n-layer, transform centered, pipes-and-filters) • Refactoring designs using design patterns • The use of components in design: component selection, design, adaptation and assembly of components, components and patterns, components and objects (for example, building a GUI using a standard widget set)			
SP	**Intellectual Property**	Philosophical foundations of intellectual property • Intellectual property rights (cross-reference IM/Information Storage and Retrieval/intellectual property and protection) • Intangible digital intellectual property (IDIP) • Legal foundations for intellectual property protection • Digital rights management • Copyrights, patents, trade secrets, trademarks • Plagiarism	**2**		