Exam Review

- 2 hour final
- Exam Schedule:
  - http://www.artsandscience.utoronto.ca/current/exams/aprilmay.html
- No aids allowed (no calculators, no cheat sheets)
- The final will include material from the entire course
  - However, the focus will be on material that you haven’t been tested on yet
    - That includes material from the entire course (not just the 2nd half).
    - That does not mean that you will not be retested on material from the midterm.

Course Outline

- Lecture 1
  - What is Requirements Engineering?
  - What are Requirements?
- Lecture 2
  - What is Engineering?
  - What is a System?
- Lecture 3
  - Requirements Specifications
  - Formal Inspections
- Lecture 4
  - Feasibility Studies
  - Stakeholders Goals
- Lecture 5
  - Elicitation Techniques
  - Risk
- Lecture 6
  - Intro to Requirements Modelling
  - Modelling Enterprises
- Lecture 7
  - Modelling Objects
  - Modelling Relationships
- Lecture 8
  - Modelling State
  - Modelling Events
- Lecture 9
  - Modelling Interactions
- Lecture 10
  - Non-functional Requirements
  - Verification and Validation
- Lecture 11
  - Prioritizing Requirements
- Lecture 12
  - Software Evolution
  - Moving into Design
- Lecture 13
  - Software Architectures
Calculations and Modelling Notations

Formulas

You are responsible for knowing all formulas taught in the course.

Modelling notations

You should know

- the syntax,
- what you can express with each modelling notation,
- how the diagrams interrelate, and
- how to verify your models (cross-checks).

UML

We’ve looked at the following UML diagrams:

- Activity diagrams
  - capture business processes involving concurrency and synchronization
  - good for analyzing dependencies between tasks

- Class Diagrams
  - capture the structure of the information used by the system
  - good for analysing the relationships between data items used by the system
  - good for helping you identify a modular structure for the system

- Statecharts
  - capture all possible responses of an object to all uses cases in which it is involved
  - good for modeling the dynamic behavior of a class of objects
  - good for analyzing event ordering, reachability, deadlock, etc.

- Use Cases
  - capture the view of the system from the view of its users
  - good starting point for specification of functionality
  - good visual overview of the main functional requirements

- Sequence Diagrams (collaboration diagrams are similar)
  - capture an individual scenario (one path through a use case)
  - good for modelling dialog structure for a user interface or a business process
  - good for identifying which objects (classes) participate in each use case
  - helps you check that you identified all the necessary classes and operations
Non-UML Modelling Notations

- Goal Models
  - Capture strategic goals of stakeholders
  - Good for exploring ‘how’ and ‘why’ questions with stakeholders
  - Good for analysing trade-offs, especially over design choices

- Fault Tree Models (as an example risk analysis technique)
  - Capture potential failures of a system and their root causes
  - Good for analysing risk, especially in safety-critical applications

- Strategic Dependency Models (*n*)
  - Capture relationships between actors in an organisational setting
  - Helps to relate goal models to organisational setting
  - Good for understanding how the organisation will be changed

- Entity-Relationship Models
  - Capture the relational structure of information to be stored
  - Good for understanding constraints and assumptions about the subject domain
  - Good basis for database design

- Mode Class Tables, Event Tables and Condition Tables (SCR)
  - Capture the dynamic behaviour of a real-time reactive system
  - Good for representing functional mapping of inputs to outputs
  - Good for making behavioural models precise, for automated reasoning

Basic Cross-Checks for UML

- Use Case Diagrams
  - Does each use case have a user?
  - Does each user have at least one use case?
  - Is each use case documented?
    - Using sequence diagrams or equivalent

- Class Diagrams
  - Does the class diagram capture all the classes mentioned in other diagrams?
  - Does every class have methods to get/set its attributes?

- Sequence Diagrams
  - Is each class in the class diagram?
  - Can each message be sent?
    - Is there an association connecting sender and receiver classes on the class diagram?
    - Is there a method call in the sending class for each sent message?
    - Is there a method call in the receiving class for each received message?

- StateChart Diagrams
  - Does each statechart diagram capture (the states of) a single class?
  - Is that class in the class diagram?
  - Does each transition have a trigger event?
    - Is it clear which object initiates each event?
    - Is each event listed as an operation for that object’s class in the class diagram?
  - Does each state represent a distinct combination of attribute values?
    - Is it clear which combination of attribute values?
    - Are all those attributes shown on the class diagram?
  - Are there method calls in the class diagram for each transition?
    - ...a method call that will update attribute values for the new state?
    - ...method calls that will test any conditions on the transition?
    - ...method calls that will carry out any actions on the transition?