**Mid-Term #2 - Study Guide**

**Updated Wed, Feb 21**

**What is covered by this mid-term exam (MT#2) ?**

* Below is a list of the major topics we have covered since the first mid-term exam (MT#1)
* All reading, lectures, homework, assignments, quizzes, and teamwork are great resources for reviewing for the mid-term.
* **Exam covers: reading assignments for Weeks 5-8 inclusive, L8-L13B, HW2 and TW4-TW7.**

**How can you prepare for the mid-term?**

Should you be prepared to summarize facts about the topics we have covered?

* No, this is not what you should expect.
* All the questions are essay questions, and you will be asked to analyze the topics we have covered, and compare, contrast, or discuss how they are relevant, useful or not, and how to apply them.
* You may have to draw diagrams based on a case study. You may have to interpret information from a diagram or a case study.

**Topics covered by this Mid-Term Exam #2:**

* Architectural styles and patterns - definition, how are they applied, how are they different, examples, how do you choose which to use for a given problem?
* Analysis process using Object Modeling Technique - be able to do some modelling, based on a case study, what do you keep, what do you omit
  + Objects, actions, sequence diagram, data flow diagrams - Context, Level 1
* REST architectural style - where it came from, what it is and isn't, core model, how it is implemented, when it makes sense, when it doesn't, strengths, weaknesses
* Learnings from our Guest Speaker - what were some of the interesting topics she discussed, why, how are they relevant.
* Functional vs Non-Functional Requirements - definition, how they are different, why they are both important, examples of each type
* Web Service Design Styles and Patterns - definitions, how are they applied, how are they different, examples, how do you choose which to use for a given problem? (Architecture Styles, Interaction Styles, Web Service Implementation Style, Web Service Infrastructure)
* Little Languages - domain specific, when might they apply, when not, why useful, benefits
* Pragmatic Programming - what is it, why is it useful, examples, when to apply.
* Software estimation techniques - common techniques in Agile estimation, how do they compare for accuracy and granularity, when do they apply, what are the deadly sins of estimation, techniques to do better estimation
* Documenting Architecture - how can you document a software architecture, what are the benefits, what are the tradeoffs, when should you do each type, how can you organize? Who will use the architecture documentation, and when?
* What you have learned working on the Individual Project
* What you have learned working as a team on the Team Project