**Final Exam Topics**

**Final Exam Topics and Areas**

In general, the final will be about the bigger concepts, the over-arching concerns for the entire course. The final will be less about the specific methods than the mid-term exams, and more about how to apply the techniques to a given situation.

Important concepts from any point in the class may be on the final. HW, TW, IP & TP, mid-term exams, and Readings assigned are a good source for material. You may have to interpret a case study, draw one or more diagrams, and write essay responses.

See also the [Final Exam Instructions page](https://seattleu.instructure.com/courses/1612931/pages/final-exam-instructions-23wq)

Software Architecture, and the Architect

* What is software architecture?  reading
* What do architects do? What should they not do?
* The People Part
* Two ways to evaluate an architecture - works? good economics?

Design Process

* Design Process
* SOLID Principles
* API Design
* Programming Languages - analyzing them
* Design Patterns
  + Gang of Four design patterns
    - strengths - when to use, how they help
    - limitations - when not to use, weaknesses
  + Web Service design patterns
* Violet UML editor project - reading
* Analysis
  + Analysis Models
  + Analysis Patterns
* Modeling
  + diagram types (UML, plus others)
    - Data flow diagrams
      * System Context Diagram - system and external entities
      * System Diagram (Level 1 - the one right below SCD, showing everything internal to your system)
    - Component diagram
    - Deployment diagram
    - Sequence diagram
    - Entity Relationship diagram
    - Class diagram
    - State Transition diagram
  + when to use / when not to use
  + what should they show or communicate
  + how to use effectively
* Architectural
  + Styles
  + Patterns
* REST
  + Fielding paper - reading
  + the model
  + how it works, what to document
  + when to use it
  + when not to use it
  + comparison & contrasts with RPC
* RPC (over HTTP), or gRPC
  + the model
  + how it works, what to document
  + when to use it, when not to use it
  + comparison & contrasts with REST
* Little Languages
  + why, when, benefits, pros/cons, examples
* NFR
  + Non-functional requirements
  + How are they different from functional requirements?
  + What are they? Examples
  + Why are they important in architecture? What is the benefit? What do they represent?
* Web Service Design Patterns
  + What are they?
  + When to use each?
  + Web Service Evolution techniques
* Pragmatic Programming
  + what is it?
  + why is it valuable?
  + what are some examples, and benefits?
* Documenting an Architecture
  + what are the key things to document?
  + what is the purpose of architecture documentation?
  + who uses it, and when?
  + what are properties of good architecture documentation?
* Individual Project, Team Project
  + anything about the projects is fair game
  + things you learned from seeing other teams and individuals architecture for the same project
* Reflections on Software Architecture
  + Why it is important?
  + How it has changed over time?
  + Do we still need it with Agile?
  + What have you learned in this course?
  + How could this course be improved?

End of Final Exam Topics.