# op08/25/17

* Design is important.
* Software­­ ­engineering is not an exact discipline.
* Software design cycles:
  + Customer has a problem and thinks they know the solution (they’re almost always wrong)
  + Do specifications then design
  + Implement/install.
  + Repeat as specifications change.
* Systems analyst: finds the need for the user and the actual problem that the user is having. They also need to integrate it with the company.
* Systems designer: takes what the analyst says and designs the specs of the project to do the job.
* Heap sort if you don’t know what the specs of the sort are.

# 9/6/17

* Minimize human cost
* Cost of implementing a system is minimized when:
  + Parts are solvable separately
  + Parts of the solution correspond to identifiable parts
* Reliability should be optimized.
  + Lower the mean-time-between-failure
  + Lower the mean-time-to-repair
* A specification is:
  + What the problem is and what we see the solution should be
  + A set of functional requirements that describe the precise inputs, outputs, algorithms, and constraints.

# 9/8/17

* SoftEng1: Page 30 –
  + Went over design. Separating tasks.
  + Cost is a monotonically increasing function of problem size
  + According to George Miller.
    - Break up problems as much as possible.
    - Average elements in a problem is 7

# 9/11/17

* According to Graicunas, 8 – peeps to a manager. It’s Miller’s law as well.
  + This has many fallacies. You can’t allow every module to call each other directly.
* Coupling = shared data
  + **Global variables = bad**
* Next: cohesion

# 9/13/17

* A module should only have one Idea per module
  + Failing this leads to program bloat.
* Functional cohesion is best cohesion

# 9/15/17

* Afferent input vs efferent output
  + Best projects had afferent input
    - Afferent: got every single part needed and got information. Then pass it to control then process transaction (only process if you have everything)
    - The result is Efferent Output
      * Afferent: “afferent” data elements are those highest-level elements of data that are furthest removed from physical input, yet still constitute real inputs to the system
      * Efferent: “efferent” data elements are the furthest removed from the physical outputs which may still be regarded as outgoing.
* Tree/Hierarchical structure is better. Faster more efficient.
* Dynamic Data Flow Analysis
  + Based off of “Elements of Software Science:” by Maurice Halstead

# 9/18/17

* You wind up with more networks then Hierarchy.
  + Networks are bad because they have more issues.
* **ALWAYS MAKE SURE IT FOLLOWS:**
  + MILLER’S LAW
  + GRACIANAEUS’S LAW

# EXAM 2

# 9/20/17

* Need to optimize for either space or time. Can’t do both.
* Minimizing the use of main memory requires modules with high cohesion and low coupling
* The designer is in charge of the optimization for space, not the programmer. Because the programmer can’t see all pieces.
* 2 schemes for optimizing space
  + Overlay scheme system
    - (softeng2 pg 23)
    - The modules are mutually exclusive so they can be separated.
  + Virtual Memory system
    - Much more convenient
    - (Break it into pages)
    - use virtual memory to load things for real memory then replace them?
* for both schemes. You need high cohesion and low coupling

# 9/22/17

* Inner nesting 1st
* Volume of reference is more important when available.
* Main memory space may be minimized and execution time reduced by duplicating code frequently used as a target for fan-in.

# 10/6/17

* Keep arrays in memory if possible. It’s faster
* Bounded & unbounded buffer problem : MAJackson.docx pg 24
* OMT.doc - -
  + Object Modeling Technique (and design)
  + UML is approx. 92% of the modeling language…
    - Object Model
    - Dynamic Model
    - Functional Model

# 10/16/17

* Optimization
  + Optimization for time should be considered only after the system has been completely designed and should never influence the design itself.
  + Only 5% of the code in a typical system accounts for 50% of the CPU time