Assignment 3
Mini-Example ATAM Utility Tree
for Deployment Blueprint #2
in Assignment 3 Example

- UTILITY
    - **Performance**
        - Minimize transaction response time
            - *Objective:* Perform function 'Process add requests' in scenario 'Register for semester courses' within 3 seconds under load of 1K concurrent requests (estimated peak based on 10K students registering over 10 days) (H,M)
                - *Metric:* Record response times for 'Process add requests' request while executing a test script from 20 machines with 50 web client processes each. (H)
        - Maximize scalability
            - *Objective:* Demonstrate that web server farm/cluster will scale by performing function 'Process add requests' under progressively larger loads (M,M)
                - *Metric:* Record response times for 'Process add requests' request while executing a test script from 20 machines with 1, 5, 10, 15, 20, 15, and 30 web client processes each. Conduct test with one web server and two web servers and compare response as load increases. (H)
    - **Consistency (consistent student master record)**
        - Ensure record integrity
            - *Objective:* Ensure student master record reflects most recent changes
                - *Metric:* Conduct the following test: (i) Retrieve a student's master record for viewing, emulating access by a faculty member when reviewing the student's grades and status during advising. (ii) Update the student's master record on another client. (iii) Attempt to save changes on the faculty member's client -- the faculty member should be notified that the record has been updated by another client. (H)

- Minimize wait time for student record
  - *Objective:* Ensure no process waits more than 1 second for access to a student's master record (M,M)
    - *Metric:* Execute function 'Update student profile' for a given student id concurrently from 20 machines with 5 processes each. Record row lock time using database profiler. (H)

- **Extensibility**

  - Enable pluggable advising service
    - *Objective:* Minimize coupling between "Advisor" component and other BB components (where "Advisor" is implemented by "Server catalog rule checkers" solution in DB#2) (L,M)
      - *Metric:* Measure coupling using "Number of dependencies between components" (M)

    - *Objective:* Maximize inheritance from "Abstract Catalog Rule Checker" (M,M)
      - *Metric:* Number of properties inherited by rule checker instantiation from "Abstract Catalog Rule Checker" (M)
      - *Metric:* Number of methods inherited by rule checker instantiation from "Abstract Catalog Rule Checker" (H)

- **Cost**
  - Minimize cost
    - *Objective:* Minimize cost of development (H,M)
      - *Metric:* Determine the initial hardware investment. (H)
      - *Metric:* Determine the initial investment in off-the-shelf software, including the database and web server. (H)
      - *Metric:* Estimate the number of person-hours required to code in-house modules, including base catalog rule checkers. (L)

    - *Objective:* Minimize cost of maintenance (L,M)
      - *Metric:* Estimate the number of person-hours required to define a new catalog rule checker. (L)

- *Objective:* Minimize cost of operation (H,H)
    - *Metric:* Estimate the number of person-hours required to set up the system before each semester registration period. (M)
    - *Metric:* Estimate the number of staff required to run the system, excluding students and advisors. (H)
    - *Metric:* Estimate the maintenance fees on off-the-shelf software. (M)

Notes

- Since some of these metrics are unique to the technologies of DB#2, the overall utility value will only allow you to compare two implementation variations under DB#2, not necessarily DB#1 vs. DB#2.
- Some metrics can be calculated or estimated before development (e.g., determining cost or estimating person-hours), while others can only be gathered after implementation (e.g., measuring response time). Naturally, the more of the deployment you have implemented, the less opportunity you have to make refinements.
- For objectives, the first H/M/L rating refers to the importance of the objective to the success of the system, while the second H/M/L rating refers to the ease by which the objective can be achieved.
- For metrics, the H/M/L rating refers to the degree to which the metric accurately predicts/assesses whether the respective objective has been met.
- One possible means of aggregating into an overall quality assessment value is as follows:
    - Determine thresholds for each metric, such that metric results can be normalized to a "metric scale" such as "good" = 3, "acceptable" = 2, and "poor" = 1. For example, maybe a transaction response time under 2 seconds is "good," while under 3 seconds is acceptable and anything greater than 3 seconds is unacceptable.
    - If an objective has more than one metric, aggregate the "metric scale" values by weighting them according to the degree to which they accurately predict/assess whether the respective objective has been met.
    - Similarly, aggregate the resulting objective values into sub-factor values by weighting based on the importance of the objective to the success of the system and the ease by which the objective can be achieved.
    - Finally, aggregate sub-factors and qualities based on the priorities defined for the deployment blueprint, in this example (i) performance, (ii) consistency, (iii) extensibility, and (iv) cost.