Assignment 2 Implementation Notes

Jeremy Clark
CSCIE-97, Fall 2017

Extensions of/Deviations from design

While I spent a lot of time (perhaps too much time) on my design phase, I still had to make changes when it came to implementation. Most notably:

- I failed to design a proper “CommandInterpreter” to meet the needs of the “command mode” - largely because I didn’t properly plan out the testing pattern and thought it would all be in the TestDriver. As such, I implemented a CommandInterpreter as part of the asn2 package that could pass messages directly to the API and throw the proper exceptions. The CommandInterpreter should also validate the format of the FQNs but I ran out of time.
- I added the concept of a “Fetcher” interface that could be used by the getState() overrides for various configuration items. Once I started building, I thought there might be multiple ways to compose the getState() output, but it turns out I only needed one concrete fetcher “StandardFetcher()”. I may not have needed to implement this way, but it seemed to make sense as I implemented…even though not part of the original design.
- As I broke into implementation, I realized that an association class of “Feature” would be helpful to store and manage Device state values. I updated the design document with this addition even though it wasn’t part of my initial designs.
- Even though not part of the requirements, I originally thought I would allow for the changing of names and removal of devices, but I dropped that once I started implementing, thinking that about potential data integrity and relationship issues.

Was Design document useful?

Absolutely, yes! Like I mentioned, I may have spent too much time on the design - but it was VERY helpful. Given the class map I had built and the use cases I described, I was quickly able to implement the core classes and test basic functionality end-to-end. Because I was able to set the foundation (that really lasted the entire implementation phase), I was able to detect and fix design limitations quickly and focus on algorithms and code sharing.

How could the design have been better or made more useful?

- I could have written better uses cases up front. While I wrote descriptions of the use cases, I came to realize later (during class modeling and implementation) that I had been ‘envisioning” use cases that I didn’t actually detail properly.
- Next time I may use a different template for my class dictionary table - one that includes more columns (in this document I just had two columns - “Attirbute/Operation/Association” and “Desription”.
- Better description and planning of the testing phase. I will definitely do that next time.
- I think I need to hone my process for creating a class diagram as well. I started with class relationships (no attributes or operations), which was working, but then I started to get into a mixed mode where I started adding operations/attributes while continuing to adjust structure.

Did the design review help?

In my case the peer review of my design didn’t help too much, and I’m partially to blame for that. By the weekend of design review, I had a design document but it wasn’t totally complete - so I shared my Class Diagram which was in a pretty complete, and reviewable state. With that said, I didn’t get too much feedback from my group partners - but I should have followed up with questions that I had about my own designs. As for providing reviews, that actually helped me a LOT. Thinking critically about other designs really made me reconsider parts of my own design.

Peer Comments and Peer Reviews

See the following documents in the same folder as this file:

Peer Review of my design.pdf
My Review of Mike.pdf
My Review of Vinay.pdf

These are printouts of the Canvas threads we used to communicate.