Gerald Arocena

CSCI E-97, Fall 2019

Assignment 5

Results Document

1a.) Below are comments I made on Krithika Sundararajan's submission:

> Hey Krithika! Below are my feedback points on your draft:
>
> - Not sure if I would include Customer Service in the Resource Management's use case diagram.
> - I like that you put Budget in its own class in the Resource Service. Transactions are also used in Customer Service. Perhaps you could reuse this.
> - Interesting, not sure I understand the spaceship association on the Communication System in the class dictionary. I thought of it as more of a dependency. But this was vague in the requirements and association implies dependency anyhow.
> - I think for deleting, could you use an AuthToken so that deletion rules can be managed that way?
> - I like that your interfaces in your class diagrams are intuitive.
> - I like that you suggested adding an observer to the Resource Service to notify of faulty or missing resources in the GUI. That's a thoughtful idea that's beyond the scope of requirements.
> - Since you used interfaces in your design you used the Façade pattern which you can add to your list of design patterns.
> - I didn't see a gui in your class diagrams. Not sure if needed but they should be simple to add. I basically thought of them as like the command processor minus the process command stuff. I think it makes sense in the class dictionary as well.
> - Didn't see much about IPFS. But maybe also consider how and who gets data back. There's a lot of different users information being filed away and retrieved. Not sure how in-depth they want us to go with IPFS. From the lectures, it doesn't look like we have to go that much into it though.
>
> Overall, well thought out and you raised points I didn't think of. I'll let you know if I have anything else to add. Looking good!
>
> As an aside, I know it's late in the semester but I just discovered that the sample solutions he posted from other classmates in Canvas are very helpful. They helped me a lot if I wasn't sure about a certain section. At least they are teacher-approved solutions if anything.

1b.) Below are comments on my design document made by Krithika Sundararajan:

> Hi Gerald,
>
> Overall, the class diagrams and use case diagrams look really detailed! These are some comments / questions I have:
>
> 1. Resource Manager class diagram:

- The Spaceship class aggregates the 3 types of spaceships. I had imagined the relationship would work better as inheritance where the PassengerShip, CargoShip and RescueShip inherit from the Spaceship class.

- A Team can also have a Person. I think the lecturer mentioned during one of the brainstorming sessions that the Composite Pattern may be a good choice for this.
- Since you have the Visitor Pattern on the ResourceImpl class, I wonder if it would make sense to allow visiting the other components like Spaceship and Communication Systems.
- There is a dependency on the Ledger class, perhaps you meant to add a dependency to the interface?

2. Customer Service class diagram:

- What does the Medium class represent?
- There is a dependency on the Ledger class.
- Would the passenger credentials be required to be stored in the Customer Service class? In my design, I simply pass it to the Auth Service. I thought it would be better to only let the Auth Service store it for security reasons.
- I think the customer service module would also have a dependency on the Flight Management module, to get the flights info.

3. Flight Management Use Case Diagram:

- I imagine the ISTS Manager is a person. In that case, should handling events be done by the controller in the Flight Management service instead of the ISTS Manager?
- Since your other 2 use cases diagrams have an Admin role, I wonder why it's not a part of this one.

4. Flight Management Class Diagram:

- I suppose the Subject and Observer interfaces should be defined in the Resource Manager. May be it's good to indicate the package they belong to in the class diagram (with the package element in Astah). Also, these interfaces are missing in the Resource Manager class diagram.
- The Flight class has no properties. Is this intentional?

5. Sequence Diagrams - these look great! Based on your designed workflow, I don't really have any comments for changes.

Let me know if you have any comments or questions. And I look forward to your review!

Thanks,

Krithika

2.) The design patterns I applied in this design were Visitor, Singleton, Factory, Façade, Observer, Command, and Composite. I highlighted where they are implemented in the design document. I think the patterns came in handy in my design and they don’t seem forced in my opinion. The Façade pattern is still trending (and rightfully so) with large companies so using it in my design adds value. I am a big

advocate of hiding the details of an implementation so as to allow for flexibility later on, among other things. I used the combination of the Observer and Command pattern again as was done in the Store 24X7 System for automated control which was a vital tool in my design. I also like how the Singleton design pattern powerfully uncomplicates design.

3.) The modular approach to the design definitely helped. It really allowed me to focus my attention on one module at a time and compartmentalized my thought process in a helpful way. I think it added worth to my design doing so in terms of reliability, maintenance, and other ways. It also made the process easier. For instance, modular programming works to prevent against circular dependencies and this made debugging my design easier.

4.) I found that I did need to go back and update the modules to support modules later on especially in the way I could couple them usefully and modularly. For instance, the human resources (person and team) in the Resource service mapped directly to the passengers in the Customer service. I initially had thought to make the Customer service depend on the Resource Service in order to have the Customer service directly update the Resource service's human resources after a passenger is created in the Customer service, but this sort of broke modularity in my opinion. So, my final design instead has the Customer service create a Passenger Registered event so that the Manager can do the passenger and human resources synchronization as it's the Manager's job to do the updates around the ISTS.

Also, I wasn't sure where to keep the list of flights. I initially had them in the Flight Manager since it's the one who provisions flights for the ISTS. But later on, I decided against that since I wanted the Manager to embody the behavioral (e.g., stateless) aspect of the ISTS as much as possible, among other things. I felt like if I kept flights and even the other objects the Manager handles like mission reports and discoveries inside the Manager, then both the Customer service and Manager would need to access the IPFS which could complicate matters or break modularity.

I also moved the status attribute from Spaceship to Flight later on in my design. I felt like the flight status was something more customer-oriented so I ended up moving it from Spaceship in the Resource Service to Flight in the Customer Service.

5.) The design review definitely helped improve my design. I think with long projects like this it's easy to forget that other people can easily get lost by what you're saying. My peer review partner was very helpful in pointing out things that could be very confusing to others and that I didn't catch. I went back and fixed the confusing areas she pointed out and at times in the process I thought to myself, "What was I thinking?" It also helped in that I got to see how my review partner interpreted the assignment which offered new perspectives. I think the peer review strengthened my design document a lot.

6.) The reuse of the Authentication and Ledger services definitely helped my design. The reuse of the Authentication service was very practical. I found a neat way to use it in my Customer service design in how I retrieved data for many different users and different privacy levels. Also thinking about access permissions for things like authorizing a passenger to board a plane, for instance, I came to realize how useful access permissions are and their range of use. The Ledger service was also very handy. I think conducting business transactions safely, but also conveniently, is something we take for granted. In my design, I just needed to refer to the Ledger service and just doing that added so much to the design's worth in my opinion.

7.) I think a development team could definitely implement this design. I tried to use the KISS principle in my design so that it'd be just simple enough to understand but also retain a level of open-endedness. I hoped to leave the creative process open so that another person could improve upon the design or find their own way of doing something. For instance, I didn't add functionality unless it was really necessary (for the most part), and other things. I also largely modeled my design after the previous Store 24X7 assignment where the Flight Management service and Resource Management service map to the Store Controller and Store Model service, respectively. I was happy with my results there so hopefully that transferred over to this design as well.