Experience Report for Modeling the Central Door Locks in ClaferMPS

By: Jordan A. Ross

How to Read this Document

- This is an experience report for creating a central door locks model in Clafer MPS.
- I (Jordan Ross) tried to be as specific as possible in my details. I apologize for any proof reading mistakes ☺
- I make some references to the power window experience report so it is assumed the reader has read/skimmed through it.
- I have named/color-coded some of the steps/entries as follows:
    - **Observations**: These are observations that I have made while working with the tool. Some of these observations can be classified as bugs, possible improvement, or limitations.
    - **Mitigation**: These are nested under observations which were added after the modeling was complete to provide some comments for how the bug could/was fixed, or how the tool could be improved, or how to address a limitation.
    - **Classification**: These are nested under observation which were added after the modeling was complete to provide the classification of the observation:
        - Bug – An unintentional defect in the tool that was not planned and can readily be fixed.
        - Requested Feature/Possible Improvement – A feature that the user requests or a possible improvement that could be made to aid the user in using the tool.
        - Insight – An insight into using the tool or a feature that stood out to the user as *nice to have*. It also includes an insight for ways to model with using the tool.
        - Limitation – A limitation of the tool in which no mitigation strategy has been identified.
        - Limitation (solved) – A limitation of the tool in which a mitigation strategy has been identified which solves the limitation or provides an equivalent alternative with no side effects.
        - False observation – This is an observation that was made but ended up being solved. The mitigation strategy for these observations details how it could have been avoided.
        - Possible Pitfall – An observation which could lead to problems in the future by either allowing incorrect models or the user not knowing something is wrong in the model.
    - **Bug**: This was a type of observation that was classified as a software bug in the tool which was usually minor. These were either annotated as fixed or not.
    - **Fixed**: Notation used to note that the bug was fixed. These were not added in chronological order.
    - **Tip**: A general tip for other Clafer MPS users that I found useful.

- Some bugs are repeated in the document to show when they were fixed.
- Happy Reading ☺

Input Material

The material that the door locks model was based off of was the plain Clafer encoding of the door locks system (put link here)

Planned Steps for Modeling Door Locks in MPS

1. Since this system is much larger I am thinking I will start with just implementing the full feature model.
2. Next I will implement the remaining layers only taking into account the basic features (i.e. lock switch position, speed control, etc.)
3. Then I will implement each of the other features (remote key access then passive key entry) one at a time.
4. Lastly, I will add quality attributes and latency information to the model.

Actual Steps for Modeling Door Locks in MPS

1. Since I already have the power window modeled and I am just adding another case study, I use the same CaseStudies project I setup when modeling the power window. What I do create is a new Solution called "DoorLocks" which will contain all the models that I create in the remainder of this document.
    a. I add the same following dependencies as I did in the power window solution.
        i. org.clafer
        ii. org.clafer.architecture
        iii. org.clafer.core
        iv. org.clafer.expr
        v. org.clafer.architecture.baseConcepts
2. Next I created a new model called "CentralDoorLocks" to represent the concrete system used for the entire car to lock all four doors.
    a. Under model properties I added the following languages used:
        i. org.clafer
        ii. org.clafer.architecture
        iii. org.clafer.core
        iv. org.clafer.expr
3. Next I want to create the feature model for the system. I create a ClaferModule DoorLocksFM under the CentralDoorLocks model.
    a. I first just write down the features in my system, with the appropriate nesting.
    b. Next I go through and use the intention menu to select with features are optional
    c. Next, I write a constraint to imply that if we have the PKE feature we have to have the RKA feature.
    d. Lastly, I wanted to add some comments to the different features so users had an idea of what the feature was.

i. **Observation**: There is no way to place a comment on the same line as a feature (I personally like this style for indication what feature I am talking about).
    1. **Mitigation**: Allow creating a comment to the side of the definition.
    2. **Classification**: Requested Feature/Possible Improvement
ii. There are two alternatives for doing this. The first is to "add documention to X" which places a comment directly above or adding a margin comment.
iii. I choose to just go ahead and use the "add documentation to X" for the different features.
    1. Bug: The layout is a little buggy in terms of placement. If you just close the module and reopen it is fine though.

4. Next I want to create the functional analysis architecture for the basic feature set. I create a ClaferModule BasicFAA.
    a. Inside the functional analysis architecture I first define the functional devices and connectors for the cylinder switches.
        i. I also define the DoorLockControl analysis function for the communication from the switches.
        ii. **Observation**: There are a lot of functions which are nicely grouped together in this system. I would like to use some sort of "virtual grouping" or even just comments to break up the textual architecture.
            1. **Mitigation**: Allow users to insert comments anywhere.
            2. **Classification**: Requested Feature/Possible Improvement
    b. I continue by adding the remaining functions for just the basic features (not the lock position or speed sensor features).
        i. **Observation**: Autocomplete/name checking is a very nice to have when you have long names such as "RearRightPassDoorLockMotor" because I remember having to fix a lot of compiler errors due to misspellings of the names. What made it worse was the compiler took about a minute to compile the model.
            1. **Mitigation**: None
            2. **Classification**: Insight
    c. At this point I wanted to visually check that I had the correct connections so I viewed my functional analysis architecture using the EAST-ADL projection view
        i. **Bug**: We should change the name from EAST-ADL project view I think.
            1. **Fixed.**
        ii. **Observation**: This was an awesome tool to check that I had the correct connections between the different functional devices and the control.
            1. **Mitigation**: None
            2. **Classification**: Insight
        iii. **Observation**: The names of function connectors can get quite large. What would be good is to not show them but rather when you mouse over it would show the name in a popup.
            1. **Mitigation**: Remove names from connectors and only show on mouse over. The names are just placeholders most times, what matters in the connection.
            2. **Classification**: Requested Feature/Possible Improvement

d. Next I added the functions for the individual lock position switch feature. To do this I nested a functional analysis fragment underneath "BasicFAA" for "DoorLockButtonFAA" which was an xor grouping for two different functional analysis architectures "IndividualLockSwitchFAA" and "CentralLockSwitchFAA"
    i. I added the respective functions and connectors under each one.
    ii. This is a similar pattern to what we did when modeling the power topology of the power window.
e. Next, I added the functions for the speed smart lock feature which again I used a nested function analysis architecture.
    i. I then again use the graphical view to check the connections
    ii. **Observation**: I also like how the nesting is captured in the graphical view.
        1. **Mitigation**: None
        2. **Classification**: Insight
    iii. **Observation**: It would be good to have "jump to definition" for the nodes in the graphical editor (i.e. you can traverse between FAA's)
        1. **Mitigation**: See observation.
        2. **Classification**: Requested Feature/Possible Improvement
    iv. **Bug**: The autolayout of the functional analysis architecture fragments is kind of annoying, still usable though.
f. Lastly, I needed to link the functions to the features. I captured this through a set of constraints in the functional analysis architecture.

5. Next I moved to the hardware architecture for the basic feature set. I don't define a separate hardware architecture for each feature but just the individual components (i.e. device node classification and power topology). I start by modeling the device node classification for the basic feature set by creating a clafer module BasicDN.
    a. I first model the *local nodes* that are needed for the basic feature set (i.e. motor assembly, switches).
        i. Similar to the FAA for the buttons, I created nested device node classification fragments for the two possible configurations.
    b. I then modeled the types of the device nodes.
        i. **Observation**: There needs to be a better way to add a device node type. When using the intention menu it automatically makes it smart. Then I have to add a new type "electr" then finally delete the type "smart". This is way to cumbersome.
            1. **Mitigation**: In the intention menu on the disclosure arrow, bring up a list of possibilities for the type.
            2. **Classification**: Requested Feature/Possible Improvement
    c. Next I modeled the *central nodes* which are references to nodes that are shared with other subsystems. At this point we have the power window solution which defines this "Car" BCM. Both systems will need access to this "Car" BCM.
        i. **Observation**: I have a couple of options here for modeling. The first is I could just do exactly what I did for the power window and just define this local "Environment" virtual package that defines a "Car" module with the BCM and other EC. The other option is to create some separate *shared* solution that will

represent the environment. Then both the door locks and power window could add the environment solution as dependency and use the define BCM and EC form there.

1. **Mitigation**: None
2. **Classification**: Insight
3. I chose to do the second option but only applied to the door locks (I will update the power window experience and make the adjusts then to the power window).
4. To do this I created a new solution called "Environment" which include same dependencies as the other solutions.
5. Next I created a model called "Car" to represent any components that belong to the car and not one of the systems modeled.
6. I then created Clafer module "Car" (like we did in the power window and define my central nodes there (i.e. BCM and EC).
7. I then define the types for the device nodes.

ii. Once I have the environment defined I add it as a dependency to my door locks solution. I also add the model Environment.Car as a dependency to my CentralDoorLocks model.

iii. Next I then defined the references I need for my base features.

d. Lastly, I wrote the constraints to link the features to the device node groups.

6. Next I move to model the power topology of the basic features. I do this by defining a new clafer module BasicPT.
    a. The power topology is pretty simple so we only need to model the load power but there is no variability there.
7. Next I move to the communication topology of the basic features. I do not model the buses here because they are shared among the different features.
    a. Bug: Device power and load power are available under the communication topology when they shouldn't' be.
    b. I also include the constraints for when optional communication connectors are needed.
8. With the hardware architecture components defined I return to the BasicFAA and define the deployments for the different functions. I don't have any cases where I need to use clafer constraints to express my deployment thus I don't create a deployment.
    a. **Observation**: Actually specifying deployment directly where the functions is turns out to be very nice because when I did it in plain Clafer in a separate deployment "component" I had to use constraints to imply the appropriate variability nesting and such. By specifying it directly where the functions are defined you can avoid this would can lead to less mistakes.
        i. **Mitigation**: None
        ii. **Classification**: Insight
9. At this point I have the basic feature set completely modeled. What I choose to do is group the basic components together in a virtual package to create some separation from the other components.
    a. This is just a personal preference.
10. Next I start with the remote key access feature (RKA) components. I again begin with the FAA.

a. This is pretty straightforward.

11. Next I continued by modeling the device node classification and power topology. I did not model the communication topology because there is only the bus that is used which we will define later because it is shared among all features.
    a. Note that each of these components will be made optional so any nested components (i.e. device nodes) don't need to be made optional unless there is variability inside having the remote key access feature.
12. Next I specify the deployments for the RKA FAA and like the basic set of components I group them in a virtual package.
13. Next I move on to the last feature, passive key entry. Again I begin with FAA.
    a. **Observation**: When modeling the function connector from the outside door handle sensor (either the button or capacitive) to the PKE control, in plain Clafer I could give more than one possibility for the sender (or receiver). I can't do this here.
        i. **Mitigation**: Allow for a union of device nodes to be given to either the sender or receiver.
        ii. **Classification**: Limitation
    b. To ensure that I have all my connections modeled correctly I use the graphical projection to check my work.
        i. **Observation**: I actually found a mistake in my model where I had given the source and target of passTransMsg to be the same.
            1. **Mitigation**: None
            2. **Classification**: Insight
    c. **Observation**: One really nice thing about how I have laid this out in terms of features is I can view just the feature FAA one at a time which is much more manageable.
        i. **Mitigation**: None
        ii. **Classification**: Insight
14. Next I move on to the device node classification for the PKE feature.
    a. Nothing really new here.
15. Next I move on to the power topology and communication topology.
    a. **Observation**: It would be really beneficial to have the graphical project for the communication topology here because it is very hard to visualize all the possibilities in text and to ensure that you have all of them correct.
        i. **Mitigation**: Implement graphical projection for topologies
        ii. **Classification**: Requested Feature/Possible Improvement
16. Next I went back to the PKE FAA and specified the deployments.
    a. **Observation**: Having some way to visualize these deployments would be really useful. I think it would be really helpful to have a way to pictorially show what can be deployed to what so modelers can double check what they have in the textual model. I don't think I would want to edit in this visualization (personally).
        i. **Mitigation**: Have some sort of table that shows the possible targets for a function
        ii. **Classification**: Requested Feature/Possible Improvement
17. I then did the same thing as I did with the other features and group the passive key entry components into a virtual package.

18. Now that I have all of the components specified what I need to do now is to create the system module and bring everything together. I start by defining the DoorLocksSys module for the system.
    a. Inside the new system I instatiate the DoorLocksFM.
    b. Next I create a new clafer module for the DoorLocksFAA where I instantiate the various functional analysis architectures.
    c. I then import DoorLocksFAA and instantiate it into the system.
    d. I repeat this process for the other components.
    e. When I get to the communication topology however I also need to include the two buses that have yet to be modeled.
        i. Bug: Adding types to the buses needs to be improved. Similar to adding device node types there should be an option the use can specify.
    f. Lastly, we need to write the constraints linking the features to the different components.
    g. With the system defined, the structure of the model is complete. All that remains now is to add quality attributes.
19. To begin with we start by creating the quality attributes module for the door locks. In addition, we also copy the preferences from the power window for the constants, since they are the same, and we rename for the door locks.
20. We begin by modeling the mass, cost, and warranty cost of the system.
    a. Starting with the basic DN we model the mass, cost, and warranty cost for the various device nodes.
        i. **Observation**: Whenever I have a device node classification I have to sum the values or it will be an open integer and impact solving. This can be lead to many mistakes. What we should have is a way of writing a general equation.
            1. **Mitigation**: Allowing general equation writing in the quality attributes model. This is an observation in the power window experience document as well.
            2. **Classification**: Limitation
    b. I then move to the remote key access where I do the same thing as I did for the basic DN
    c. **Observation**: It is such a pain to write out this equations…
        i. **Mitigation**: This is covered by an observation in the power window experience document.
        ii. **Classification**: Limitation
    d. Once the individual components have mass, cost, and warranty cost for the device node I sum them again in the DoorLocksDN
    e. Next I add lengths, costs, and mass to the power topologies and then sum again in DoorLocksPT
    f. **Observation**: I was just thinking it would be really cool if you could configure a setting such that when you viewing a topology graphically you could view one of the quality attributes associated with the connectors (such as length) on the line.
        i. **Mitigation**: See observation.
        ii. **Classification**: Requested Feature/Possible Improvement

g. Next, I do the same thing for the communication topology. I also assign the latency related qualities as well.
h. Now that each of the components has a total we need to complete the modeling by propagating the sums up to the system.

21. Next we need to model the latency of the functional analysis architectures.
22. With the latency modeled we need to model the timing chains and latency requirements.
    a. **Observation**: We can't use the quality attributes model actually because each Functional analysis will have the defined chains while we only want to define them for the DLockSysFAA. To get around this we will have to use plain Clafer which is going to be ugly. We should have a better mechanism for handling this.
        i. **Mitigation**: Implement a mechanism for handling timing chains in the reference model (first using plain clafer) then implement in MPS
        ii. **Classification**: Limitation
23. Eldar made another release of claferMPS in which I updated so that I could access the new diagrams.
24. First thing I did after updating is to update my model which went smoothly. There were no issues when updating.
25. Next since Eldar has included the diagrams for the hardware architecture I revisited the early observations regarding the hardware architecture. In addition I also included all of the diagrams (at their respective stages) to the experience report. Note these were added out of chronological order.
    a. Revisiting observation 15.a:
        i. **Bug:** Optional device nodes are hard to see the dashes. Can we make it a black outline?
        ii. **Observation:** The combination of the DoorLocksCT and PassiveKeyEntryCT is interesting to visualize due to how we modeled. If one wanted to include the bus in the passive key entry CT such that they could view the entire communication topology we could define a reference to the bus.
            1. **Mitigation:** None.
            2. **Classification:** Insight
        iii. **Observation:** The autolayout is very messy for the passive key entry CT and is hard to visualize with out re-laying out the diagram.
            1. **Mitigation:** Implement a better auto-layout.
            2. **Classification:** Requested Feature/Possible Improvement
26. Eldar made changes again but this time fixing the generators along with a few other improvements. I proceeded by updating the bugs if they had been fixed.
27. Next, since Eldar informed me that we could not have the "Car" solution when using the generators I removed the references to it and copied the same thing I did for the power window. This is definitely a hack for the time being but it allows testing the generators.
28. After these updates I was able to build the door locks solution with zero errors. At this point we have generated Clafer but now we need to transition to using the Clafer compiler and chocosolver to validate that the generated Clafer is correct.
29. Before validating, Eldar made lots of changes to deployment among other bug fixes. With these changes I need to update my MPS model. Based on the two door power window discussion and

updating the deployment for it we need to write one deployment for the entire model. We unfortunately can't split it up into fragments like we do with the other layers.

a. Thus I created DLocksDpl Deployment in side of DoorLocksSys.
b. Observation: I found another a mistake in my summations for the quality attributes. I thus decided to take the plain Clafer approach and remove all "totals" for the layers and just write a single top level constraint in the system. (i.e., sum DeviceNode.cost)
c. Observation: Splitting the deployment out is nice when debugging **BUT** it leaves a user open to make so many more mistakes with replicating the variability. If the don't it could leave them with an invalid model.
d. Bug: The MPS editor is so buggy… It randomly adds newlines and removes them until you close and open the module.
e. Observation: I forgot to add quality attributes to the Environment device nodes.

30. Now that the model has been updated. I wanted to first "review" my model using the graphical project to ensure I had not mislabeled anything.
31. Next I generated the clafer and see if it would compile. I had to fix the following things right away:
    a. The device node reference
    b. The path of "fa" and "ha" in the deployment didn't take into account nesting.
    c. After fixing those two things I was able to compile the generate clafer.
    d. I then tried to generate the instances and was able to get instances. I wasn't fully certain that we actually got the same number of instances. It requires more testing.
32. Eldar made more changes to MPS to support generalization and specialization by using the references. This is really only needed for the two door power window but we also implemented the door locks with it for consistency.
    a. I started with the basic fragment. I set the dn reference for the power and communication topology to the set of basic device nodes BasicDN.
    b. I then continued by doing the same for the remote key access PT.
        i. Observation: Because the RKA PT uses both basic and RKA device nodes I had to import the DoorLocksDN to access both device node classifications.
    c. For the passive key entry I had to apply the same technique.
33. Eldar also added implementation choices for analysis function and functional devices so I also went through and annotated each of them with their respective choices.
34. I then tried generating the Clafer and was able to obtain results but it took much longer than in plain Clafer.