Specification Critique

Group 03
se2020.grp03@cse.unsw.edu.au

05 August 2012

Contents

1 Specification Critique

1.1 Introduction

Invariants & Theorems

Invariants should express all constraints on the machine that are important to the integrity of the system. They are not merely used as a method to declare variable types. The invariants should be used to specify the semantic relationships between variables. It is also said that invariants should be as strong as necessary but no stronger. Theorems provide a way for checks to confirm those properties that are *"obviously"* true.

Correctness & Consistency

Correctness & consistency is one of the important aspects in a model. Not only the model has to be correctly modelled, i.e. all proof obligations discharged, it has to be consistent with the requirements. The model also has to be complete, fulfilling all the requirements.

Concrete

The models should describe behaviour, not details of how that behaviour is obtained. Therefore, models should be abstract, rather than concrete.

Machine Sequence

The way the refinements are carried out in the model should be in a way that the previous machine has a link to the next. The refinements should carry over properties is related to the previous machine. Machine Sequence affects the flow of implementation and deployment sequence.

Clarity

The model should be readable by humans. A model that looks complicated and impressive might not be a good model. Don't confuse complexity with quality. Readability of naming convention is also very important. Clarity will allow for easy implememtation.

1.2 Invariants & Theorems

Invariant used to reinforce semantic relationships

Group 4

Semantic relationship are reasonably obvious in group 4's model- there were strong constraints in some machines that show linkage between each variables. These constraints were primarily relationship between size and numbers, and were particularly stronger in areas such as inventory and warehouse stock management. Such example would include usedSize and maxSize from the storage area machine.

However in machines where integrity constraints were lackluster, it can be seen that this is due to the model's several 'small refinements', where only one to four variables were only introduced. This became apparent with the newer variables having no linkage with the

former variables, such as warehouseThreshold and a storage's usedSize. Furthermore, basic number values and size comparisons were neglected, for example the relationship between the currentDay, a product's timeToExpire and its expiryDate, despite including many of these in the earlier refinements. These relationships were only reinforced in the guards in the relating events.

In one respect, the lack of constraints can be seen as acceptable, owing to the fact that this spec attempts to model real world restrictions through its idea of having general attributes, and essentially having one variable representing the many viable 'loss' constraints.

Group 7

Group 7's invariants and their properties are similar to group 4's, in that some constraints were included and appropriate at times, and lacking in the later refinements- many of the later invariants were type based constraints. It was also noted that this group's spec had the most number of refinements out of the four.

Similar to group 4, each of these refinements, especially the latter ones introduced one to three variables. There was also a distinct lack of important associations between older and newer variables, for example, the stock threshold in each floor and the floor's capacity as well as orders and their amount not exceeding the capacity. This can be largely due to high numbers of refinements, where the separated related variable become easily lost between each refinement.

The constraints were largely relevant to warehousestock, and floor locations, similar to group 4's, where it establishes separate compartments and entities a product has to exist in. For example, the linkage between products, stock, floor, backroom and warehouse in the location machine. Compartmentalisation is also seen with registers, in particular moneybox, storeSafe and sumOfRegisters, in the later refinements.

Group 11

Group 11 had both appropriate and unnecessary invariants. Given its high number of undischarged POs, it can be seen that many of these unnecessary invariants backfired and rooted logical and correctness issues. Similar both group 4 and 7, many of these integrity constraints were not sustained throughout the refinements, with lesser in the loyalty, discounts and schedule machines.

Integrity constraints were stronger in areas such as transactions, where linkage was very clear between how the instore cashtill operates with its topay and transactionInProgress variables, accompanied by comments that describes this relation. The spec also at times associates variables from its previous refinements, such as products in the shoppingcart with the number of stocks available to purchase.

There were no invariants in the Users machine where authorisation and privileges were enforced. Many of these relationships depended on its event based guards, with no invariant that ties together what a higher level access level can general 'modify'. The lack of constraints in the more complicated concepts such as the spec's scheduling system created unclear and ambiguous functionalities in the spec. For example, whether if a schedule can be both a specialSchedule and orderSchedule at the same time and function concurrently. Moreover, this ambiguity in the model underscores flaws in the original requirements and design, rather than just the spec/event b model itself.

Group 3

Based on the above three specs, we (group 3) derived a number comparable improvements to be made. The main issue with our spec concerns the lack of constraints that describes the semantic relationship between variables, and this can be seen through the number of proof obligations our model had in total. Through identifying and understanding other models, we were able to highlight how weak constraints impact the cohesion as a system, and the corrective value of the model.

It was also observed that while the above three specs operates with multiple functioning trolleys of products by different users simultaneously, our model handles one transaction at a time, given one moneybox and trolley, and this can somewhat infer a distinct relationship with the number of complex constraints our spec had to consider.

Despite having no integrity based constraints, some of our guards were sufficiently comprehensive, and thus can be translated and generalised to become more appropriate invariants. For example, the interaction between the purchases of ReservedStock, ReservedNum, stock and trolley with ActiveProd can be readily prepared and translated to an invariant based on the BuyReserve and ReserveProduct events. With these guards, our team can consider outlining possible and more effective invariants to add and improve in the later stages.

Use of theorems

In all four groups, no theorems were included, and will be disregarded for this basis of this critique.

1.3 Correctness & Consistency

Group 04

Gruop 04 was able to model most of their requirements. There

Group 07

Group 11

Group 03

1.4 Concrete

Group 04

This group demonstrated good abstract design for their requirements. As we can see, the product requirements only describe the behaviour of the system instead of depicting specific stuff on how the requirement will be accomplish. However in event-B, their way of constructing the user access is too concrete because they have to specify which user it is in each function. Looking at figure 1 (appendix A), they just model the behaviour of what the system is going to do without any specific information that suggest how the system would meet the goal. For example in appendix B.1, the function Set Threshold contains the guard specify the user have to be a manager in order to use this function.

Group 07

In appendix C, the product requirements are too specific. They should move those requirements down or make them more abstract. For example, PD–2.5.3 states that items are placed under the same classification so this will tell where this item should be placed. This requirement should be replaced as the system can handle item storage. As for the Event-B, they have too many guards in each function that would make their model too specific. For instance in appendix D.1, the function Add Product to Warehouse has a lot of guards while there are only two actions. They should delegate the invariants at each machine instead of within each function. Then in appendix D.2, the function contains too many details which would make the model too concrete. The guard that restricts the staff from making the transaction because the registrar doesn't have enough money to give the changes is a huge problem for the staff. The staff would have to go get the changes from somewhere else then come back to notify the system that the registrar got more money then it could process the transaction.

Group 11

Referring to appendix E, the requirement describes how it would achieve the goal making it too specific for a product requirement. The requirement should be more abstract like the system can handle replacement orders or else you can't extend that product requirement further down to design level. Looking at this simple function Give Change in appendix F, the function has too many guards for only one operation. More guards should be delegated at the top of the machine and the invariants could be more general to make the model more abstract.

Group 03

In appendix G, the requirement is specific to list out the attribute that the system would store. It can just be the system can record product details and then extends that requirement down to design level where you can then state that it can store these types of attributes. Referencing to appendix H, the function contains too many guards that would make the function too specific. The function could be better to rearrange the guards out of the functions.

Overall, all the groups have some parts that are well model and some parts are too specific. Out of all the groups, group 4 have the most abstract model. Our group (group 3) is place in the middle of the bunch. Group 11 have too many details in some of the functions making it to cause too many errors in the model.

1.5 Machine Sequence

Group 04

Machine Sequence

- StorageArea
- WarehouseR0
- WarehouseR1

- InventoryR3
- StoreR4
- StoreR5
- StoreR6
- AdministrationR7
- WarehouseR8
- StoreR9
- CustomerR10
- ExpiryR11
- ExpiryR12
- SplittersR13
- ExpiryR14

Critique

Group 04 has the most number of machines among the 4 groups that we have decided to critique on. Coming in at a total of 15 machines. They started their model well. As the machines have strong relationships/linkage between each other as the refinement progresses. But started showing problems in their from AdministrationR7 onwards. Having not planned well before hand, they had to refine AdministrationR7 into WarehouseR8 to add in *event SetThreshold* and *event MoveStock refines AddObject*. They could have easily done that in StoreR6.

Following on, they modelled StoreR9 to be able to choose, purchase, refund and move products. CustomerR10 to model loyalty points and memberships. Next, they followed on by modelling ExpiryR11, ExpiryR12. These refinements could have been incorporated into previous machines and clearly do not have any relations to the previous machine CustomerR10.

They tried to split the stores/warehouse into different areas with the next refinement being SplittersR13. And with the new areas, they had to set the expiry to move stock to the new designated areas resulting in the refinement ExpiryR14. Due to lack of foresight and planning, the group did not have a good machine sequence.

Instead of continuing to refine the machines in this way, they could have saved themselves much trouble by actually editing the previous machines to reflect the changes in work flow instead of pressing on.

Group 07

Machine Sequence

- POS-Product-R0

- POS-Stock-R1
- POS-Location-R2
- POS-Trolley-R3
- POS-User-R4
- POS-Return-R5
- POS-StorageCapacity-R6
- POS-UserClasses-R7
- POS-Days-R8
- POS-Threshold-R9
- POS-Registers-R10
- POS-ProductType-R11
- POS-Discount-R12
- POS-PaymentMethods-R13

Critique

Group 07 has quite a number of machines, 14 machines. Although unlike Group 04, overall, they have a better thought out model in terms of machine sequence. Most of the refinements are clearly related with each refinement building on to the previous machine. Improvements could still be made though.

If they had added in the different user capabilities from POS-UserClasses-R7 into POS-User-R4, they could have brought POS-Threshold-R9 and POS-ProductType-R11 up into the earlier refinements after POS-StorageCapacity-R6 and the later refinements would only be dealing with *Registers, Discounts and Payment Methods* which would have a better flow. In this case, even when dealing with a large number of machines, Group 07 has done well in terms of machine sequencing.

Group 11

Machine Sequence

- BasicPOS
- Orders
- Locations
- Users
- Transactions
- Loyalty Program
- Price Modification and Refund
- Schedule

Critique

Group 11 has less machines compared to the previous 2 groups. The flow is similar to group 07, with the basic warehousing system modelled first, followed by ordering of stock from suppliers. And then on to the Locations which should come before Orders, as it makes for a better machine sequence. The next refinement, Users and then to Transactions followed by the Loyalty Program. Price Modification and Refund machine should come before the Loyalty Program as it has more to deal with Transactions. Schedule should come after Orders or Locations as it deals with the scheduling of stock and not refined from Price Modification and Refund

Group 03

Machine Sequence

- PoSWare
- WarehouseR1
- StoresR2
- RetailR3
- MembershipR4

Critique

Our group also has less machines compared to group 04 and group 07. Comparatively, our group has good machine sequencing. With modelling the behaviour of the overall system in PoSWare and refining it into Warehouses and Stores. Following that, it was further refined to Retail to deal with transactions and refunds. And the finally refined into Membership to deal with various membership benefits.

1.6 Clarity

Naming the project

The name of project should clarify breafly at the first look.In SENG projects, It's a good practise to mention the Group number and the module's name which is moduled. None of the group have done it properly, Group 07 chose the name "SENG", Group 04 chose "PosWare" and Group 11 called it "POS". None of these gives any information about which group are these which just make it hard for marker to figure group's number by himself. However group 3 just called Group3, which is the best in here but not enough. (no mentioning the mudoles name)

Good use of naming varables

Group 4: At early stage of refinement group4 have clear and distinguished names for variables and parameters, but at the end many names are almost repeated for different varable without extra explanation such as : "basket objects" and "basket product".

Group7: chose resemble naming for their variable but they haven't keep the correct style

of naming system, such as instead of "floorShelves" they used "floorshelves" which in a long list will makes hard to distinguished.

Group11: Group , made the same mistake as Group 7. such as "pointsaccumulated" , These mistakes makes it hard to read, which is one of the main purposes of moduling.

Group3 : same mistake but not as much as other groups . As an example "Reorderlevel"

Good naming of refinement and correct positions for refinement, for easy finding.

In General a Week point of Event-B is it will order the machines based on alphabetic order, while it might shows notting and just caused the misunderstand. But having said that it can be handled.So reader can understand which machine is the refinement of which one.

Group 4: Group 4 did a trick by adding R[number] at the end of each machine and context . It number indicate the number of the refinement. However, they haven't done it properly by forgotting to use this system for all machines , such as example "storage area" has no R at the end. Beside the ordering is still confusing . They could use the numbering at the beginning.

Group 7: Group 7 did the same technique but correctly , however they are also having the same ordering issue , including the extra "POS_" at the beginning which again make it harder to read trough the list. Such as "POS_Discount_R12","POS_Days_R8" and "POS_Return_R5". It's a psychological reason that eye first look at the start of line which here it's just lead to confussion.

Group 11: This group haven't done anything to solve this problem and it will make the reader just to look each machine and flow the path backward to reach the beginning. And again search for the refinement. So imature.

Group 3: This group haven't done anything extra than groups 4 and 7.

Conclusion: In order to make it easier to read the machines in project and find out the refinement of a machine, machines should be called 1.[machine_name], based on their ordering system, so evenB order it automatically and also the user knows which machine is the next refinement.

Adding new events properly

In EvenB when new events are added to the refinement they should be added to the top of the list, not at the button. This simple mistake will make the user just scroll down in each refinement to reach the new events.As an example the in group4 for machine ExpiryR12 to reach the new events user needs to pass 14 old events.The only group who did this properly is the group 3.

To sum up, some easy practice which can make the module easier to read have been forgotten by many groups. Such as mentioning the group number and the module name on project name. Clear and distinguished naming for variable events and parameters with us-

ing the correct naming style. Ordering the refinement in-order to make it easy to find the next refinement. And the last adding new events to the top of old event in order to prevent scrolling all the way down.

2 "New" Specification Summary

3 Project Plan

4 Appendix