Tutorial 1 Solution Guidelines: Inf2C-SE 2014/2015

Friday 3rd October, 2014

Below I have given some notes to the questions posed in the first tutorial. Naturally, the solutions the students give will vary widely, much of what is "right" or "wrong" is subjective.

I've given a couple of sample use-cases that can be used as examples to get your students started if they are having difficulty getting on with their own.

I've included my thoughts on the "bad" use case examples and some thoughts on the more advanced discussion section. I hope you find these useful but you may of course have some thoughts of your own.

1 Basket Sub-total Calculation

Use Case Name Multiple Cheapest Item Free Offers

Iteration Draft

Summary The customer has items in their basket which make up multiple special offers. For example there is a 3-for-2 cheapest-item-free offer and the customer has 6 qualifying items.

Basic Course of Events The customer has multiple instances of multiple-item offers. The total should group these such that the end total is as low as possible for the customer, since the customer may well be expecting this behaviour.

Alternative Paths The customer could of course split up their offer-items into distinct transactions to force the most optimal grouping of the items. Customers are likely to annoyed at this though.

Exception Paths If we aim to keep a 'running total' as items are placed into the basket, or scanned at the checkout, we must be willing to break apart previously grouped items to achieve the optimal grouping. In addition there may be multiple ways to group the items which still evaluate to the optimal total, in which case we impose no constraints on the grouping.

Extension Points This is an extension of the simple case in which items are grouped as they are added in a greedy manner which may not provide the optimal grouping for special-offer items.

Trigger The customer adds an item which means that there are multiple ways to group special-offer items.

Assumptions

- We assume that the customer would prefer to pay the lowest amount possible.

- We assume that the optimum grouping can be calculated quickly. If we cannot guarantee this algorithmically a timeout should be considered. However it is not clear what should happen in the event of a timeout. *Note that this should be cleared up in a future iteration of this use-case.*

Precondition The contract between this use case and the outside world. All things that are outside the scope of the system, but are required to be there for the use case to work.

Postcondition Assuming no timeout on the calculation the sub-total of the user's items should be displayed and this sub-total should be the lowest prices corresponding to the optimal grouping of special offer items.

Author Use-case author

Date Use-case date.

Use Case Name Item Qualifies for multiple offers

Iteration Draft

Summary The customer has an item which qualifies as part of more than one special offer. For example it is available as a buy-one-get-one-half-price but also as a part of a 3-for-2 cheapest item free. The customer has two of the given item *and* a third item in the 3-for-2 offer.

Basic Course of Events This is a specific case of the above use-case. In this case we wish for the sub-total to be re-calculated to the optimal (lowest) price for the customer regardless of the order in which the items were added. So in this case it depends on whether the cheapest of the three items in the 3-for-2 offer is priced higher or lower than half the price of the item in the buy-one-get-one-half-price offer.

Alternative Paths As before the customer could manually arrange the items into separate baskets/transactions, but this represents an undesirable burden to place upon the customer.

Exception Paths There are no obvious error paths here. The optimal grouping algorithm may take too long, or the customer may wish to know why the items were grouped as they were. These points should be expanded in a later iteration.

Extension Points This use case is a specific case of the "Multiple Cheapest Item Free Offers" use case.

Trigger As described in the summary.

Assumptions The assumption is that the user wishes to pay the lowest price possible.

Precondition There is some function that maps items onto the special offer groups for which it is valid.

Postcondition The optimal sub-total for the customer's current items is calculated and displayed.

Author Use case author.

Date Use case date.

2 Synced Push Notifications

For synced push-notifications we concentrate on those that are different from the standard push-notification system.

Use Case Name Clear Notification Without Reading

Iteration Draft

Summary The user should be able to clear a notification without actually opening the application from which it has originated.

Basic Course of Events
- A notification has appeared in the notification bar.
- The user swipes the bar down to reveal the notification and all other uncleared notifications.
- The user swipes the notification to be cleared either to the left or to the right.
- The notification is removed from the notification bar.
- The notification is also removed from and will not appear on any of the other devices the user owns.

Alternative Paths
- The user may touch the notification to open the corresponding application. In this case the notification is not by default removed, but may be by the application itself.
- The user may press and hold a notification to remove it from the current device without removing it from any others devices.

Exception Paths
- There is no current internet connection and the user has swiped a notification to remove it from all devices. In this case the removal request should be queued until an internet connection becomes available. This means that a device should be able to handle receiving a request to remove a notification that the user has already manually removed.

Extension Points This is an extension of simple push notifications that can be removed on the current device.

Trigger An application adds a notification to the notification bar and the user wishes to remove it.

Assumptions There is some service to enable the syncing between devices. Devices should not need to know about each other or communication with each other directly.

Precondition The contract between this use case and the outside world. All things that are outside the scope of the system, but are required to be there for the use case to work.

Postcondition The removed push-notification is removed from the current device. For all other devices the push-notification is removed soon after the user has removed it from the remote device *and* a usable Internet connection is achieved.

Author Use Case Author

Date Use Case Date.

3 Bad Use Cases

This section offers some discussion of why the "bad" example use-cases were considered to be bad. There may of course be additional reasons as well but the bad use-case examples were intended to highlight the mistakes detailed here.

3.1 Poll Rooms

This is mistaking steps required to bring about the result of a use-case for a use-case itself. The user does not need to know how, when or even if the monitored rooms are polled. All the user cares about is that they can set the desired temperature and the heating system will be turned on/off appropriately.

3.2 Efficient Route

This is more of a functional requirement. It is less visable to the user. In theory one could suggest that it provides some added value to the user (at least in aggregate) because their elevator calls are responded to more quickly. However, even with this, this is too much of a requirement of the system, rather than a use case from the perspective of the user. All use cases should be from the perspective of the user. You may of course include that the user would wish for speedy service which may then be translated into some kind of requirement, which *may* talk about an efficient route.

In addition, this use case manages to sneakily include a further requirement. That each user of the elevator is moved monotonically towards their destination floor. Again this is more of a requirement, but if included as part of a use-case then it should certainly be a part of a use case for the elevator which begins with the user entering the elevator and selecting their destination floor.

Note; of course this use-case is outside the scope of the question since I explicitly suggested the students do not consider multiple elevator users. That was to keep the system as simple as possible. However, they can still decide why they consider this to be an inappropriate use case.

3.3 Clear Push Notifications

This is an example of a bad use case because it is too prescriptive of the design used to implement the use case. It can be well worth commenting on design constraints that you feel are imposed by the use case. However, one should refrain from doing actual design and in particular making design choices that are not enforced by the use case.

4 Advanced - Parameterised Use Cases

This is free discussion so any points that you think are valid are fine. The students may well think of things you have not thought of. A couple of obvious points: firstly, there are situations in which use-cases for different functionality can be very similar. The common example of this is searching. Searching for different entities may be mostly the same, the user wishes to search for all students by degree, the user wishes to search for all students by class, etc.

Secondly, one must remember that use-cases typically must be accessible to both the software engineers *and* the other stakeholders such as users. Parameterised use-cases may well be accessible to software engineers who will find the notion of a parameterised entity rather intuitive (or they would not be software engineers). But the general public may have more difficulty

in understanding a parameterised use-case. This could be somewhat mitigated by having some tool support to expand the instantiated instances of the parameterised use-cases.

A slightly more subtle point would be that care needs to be taken that the design/structure of a set of parameterised use-cases does not unduly influence the design of the system. Just because there is a generic search-for-students use-case does not mean that this needs to be represented in the code-base.