Software Specification (ES) 2018/2019
Instituto Superior Técnico
1^{st} Project

Alloy Model Due: November 9, 2018, 20:00

1 Introduction

The development of large information systems is a complex process and demands several layers of abstraction. Specifying the problem in a rigorous way is the first step. Then one may perform formal a analysis and prove correctness or, in case the specification is not correct, to unveil faults that would be hard to detect by other means.

The first project of the Software Specification course unit consists on the specification and verification of a fragment of a system called *Adventure Builder* that models an application for reserving adventures.

The main objective of this project is to encourage the practice of specification of problems in a rigorous way using a formal specification Language (the Alloy language), and then to perform verification of the developed model. The second aim is to give some experience in using automatic verification tools (the Alloy analyzer).

2 Problem Specification

The problem that should be specified is a simplification of a module system Adventure Builder that models (1) operations on client bank accounts, (2) hotel room reservations, (3) activity reservations, (4) reservations of adventures for clients via brokers, (5) tax (on what is being bought) and annual tax reduction. Adventure Builder is responsible for managing the *accounts*, *activity offers*, *activity reservations*, *room reservations*, *adventures* and *invoices*, available in the system.

The rules for clients, bank accounts, etc. in Adventure Builder are enforced by a given policy (detailed later in this document).

This problem can be divided into several different sub-problems/dimensions, dealing with different levels of abstraction. It is the responsibility of the students to identify at which modelling level it is appropriate to define each property/operation[1].

The following sets should be defined:

- DATE with all dates.
- CLIENT with all the clients that can use the Adventure Builder system;
- BROKER which is a subset of CLIENT;
- BANK with all the banks, where clients can have multiple bank accounts;
- ACCOUNT with all possible bank accounts that clients can open;
- HOTEL with all hotels, where clients can stay during the adventures;
- ROOM with all possible hotel rooms;
- ROOMRESERVATION with all possible room reservations that can be registered in Adventure Builder;
- ACTIVITYPROVIDER with all the activity providers (each activity provider provides a set of activities) connected to Adventure Builder;
- ACTIVITY with all the activities provided by the activity providers;
- ACTIVITYOFFER with all the activity offers, offered by the activity providers, that can be registered in Adventure Builder;
- ACTIVITYRESERVATION with all the activity reservations, that can be registered in Adventure Builder;
- ADVENTURE with all the possible adventures (integrating activity reservation and room reservations) that can be registered in Adventure Builder.
- INVOICE with all the possible invoices (result of payments) that can be registered in Adventure Builder.

Beside the sets above, there is also a tax authority IRS.

We will now briefly describe each entity and in the Sections 2.9 and 2.10 we provide additional requisites.

2.1 Dates

The dates should be modelled as a linearly ordered set. (The set `int` should not be used as dates.)

[1]See Section 7 for tips on how to approach this problem.

2.2 Clients and brokers

We abstract away from all characteristics of clients. A client only has an identity, which is implicit since the clients form a set (members of a set are unique). A broker is a particular kind of client.

2.3 Banks and accounts

The only characteristic of a bank (except for having an identity, which is implicit) is that it provides bank accounts. We say that *an account is in Adventure Builder* if it has been opened (by a client). Each account in Adventure Builder has three attributes: (1) a bank, (2) a client and (3) a balance. The balance may change over time. A bank can provide several accounts, but each account is provided by exactly one bank. Each client can have several accounts, but each account belongs to exactly one client.

Initially there are no accounts in Adventure Builder. New accounts can be added to Adventure Builder as long as they do not exist in the system yet. When a new account is being opened, the balance is 0. It is possible to change the balance by an operation **deposit**.

2.4 Hotels, rooms and room reservations

The only characteristic of a hotel (except for having an identity, which is implicit) is that it has rooms. Each room has two attributes: (1) a hotel and (2) whether it is single or double. A hotel has several rooms, but each room belongs to exactly one hotel. Each room is either a single or a double room.

We say that *a room reservation is in Adventure Builder* if the room has been reserved (by a broker for a client) but not canceled. Each room reservation in Adventure Builder has four attributes: (1) a room, (2) a client and (3) arrival and (4) departure dates. The arrival date is before the departure date.

Initially there are no room reservations in Adventure Builder. New reservations can be added to Adventure Builder as long as they do not exist in the system yet. Existing reservations can be cancelled.

2.5 Activity providers, activities, activity offers and activity reservations

The only characteristic of an activity provider (except for having an identity, which is implicit) is that it provides activities. Each activity has two attributes: (1) an activity provider and (2) a positive capacity (maximum number of of people for the activity). An activity provider may provide several activities, but each activity is provided by a single provider

We say that *an activity offer is in Adventure Builder* if the activity offer has been made (by a provider). Each activity offer in Adventure Builder has four attributes: (1) an activity, (2) a begin date, (3) an end date and (4) an availability.

The begin date is before the end date. The availability is a number indicating how many more persons that can register for the corresponding activity. This number may change with time.

Initially there are no activity offers in Adventure Builder. New offers can be made to Adventure Builder as long as they do not exist in the system yet. When the offer is made, the availability must not exceed the capacity of the corresponding activity. Moreover, when an activity reservation (below) is made, for a number of people, the availability of the activity offer decreases with the same number.

We say that *an activity reservation is in Adventure Builder* if the reservation has been made (by a broker, for a client). Each activity reservation in Adventure Builder has three attributes: (1) an activity offer, (2) a client and (3) a positive number (of people that will do the activity).

Initially there are no activity reservations in Adventure Builder. New reservations can be added to Adventure Builder as long as they do not exist in the system yet. Existing reservations can be cancelled.

2.6 Adventure

We say that *an adventure is in Adventure Builder* if the adventure has been created (for a client, by a broker) and not canceled. Each adventure in Adventure Builder has nine attributes: (1) a client (who is paying for the adventure), (2) a positive number (of people that the adventure is for), (3) the broker (who arranged the adventure) (4) a room reservation, (5) an activity reservation, (6) a positive amount (cost of the adventure), (7) one of client's accounts, (8) one of broker's accounts and (9) the state of the adventure. The client is the same as in the room reservation and the activity reservation. The adventure can be in three different states: InitialState, PayedState and ConfirmedState.

Initially there are no adventures in Adventure Builder. New adventures can be added to Adventure Builder as long as they do not exist in the system yet. Adventures can be created (added), payed, confirmed and canceled. When an adventure is created, it is in the state InitialState.

2.7 Invoice

We say that *an invoice is in Adventure Builder* if it has been made and not canceled. Each invoice in Adventure Builder has four attributes: (1) a client, (2) a type describing the kind purchase, (3) the amount (without tax) and (4) the tax. There are two kinds of purchases: Leisure and Business.

Initially there are no invoices in Adventure Builder. New invoices can be made to Adventure Builder as long as they do not exist in the system yet. Invoices can also be canceled.

2.8 IRS

IRS is the tax authority. IRS provides functions for calculating tax on purchases and annual tax reduction for made purchases. The tax and depends on the price (of what is bought) and the type of purchase. It is payed directly. When a client pays a broker for an adventure, the client pays the price plus tax. The broker receives only the price. Once per year the client's accounts are credited with tax reduction for all purchases made during the last year. The annual tax reduction is calculated from the invoices, of the respective client, in the system.

2.9 Operations

The system should provide the following operations:

Input	**Effect**
openAccount	
`account, client, bank`	Opens the account `account`, at bank `bank` for client `client`.
clientDeposit	
`account, amount`	Deposits the amount `amount` into the account `account`. The amount can be negative.
makeActivityOffer	
`offer, activity, begin, end, availability`	Adds the activity offer `offer` with activity `activity`, begin date `begin`, end date `end` and initial availability `availability`.
createAdventure	
`adv, client, num, broker, actReserv, roomReserv, amount, fromAccount, toAccount`	Adds the adventure `adv` for client `client` and the number of people `num`, with broker `broker`, activity reservation `actReserv`, room reservations `roomReserv`, price for the adventure `charge`, client account `fromAccount`, broker account `toAccount`. After this operation, the state of the adventure is InitialState.
payAdventure	
`adv, client, num, broker, actReserv, roomReserv, amount, fromAccount, toAccount, invoice`	Makes, for client `client`, the activity reservation `actReserv` and hotel room reservations `roomReserv` needed for all participants, calculates tax of the purchase, debits the client's account `fromAccount` with the amount `amount` plus tax, credits the broker's account `toAccount` with `amount`, makes an invoice `invoice` for the purchase with purchase type Leisure. After this operation, the state of the adventure is PayedState.
cancelAdventure	
`adv`	Cancels the adventure (undos all operations done by **payAdventure**) and removes it from Adventure Builder.
confirmAdventure	
`adv`	Changes the state of the adventure `adv` to ConfirmedState.
makeAnualTaxRed	
`accounts`	Credits each account in the set of accounts `accounts` with annual tax reduction for the respective client. This operation also removes all invoices from Adventure Builder.

The following operations are not provided by the system, but they should preferably be defined and used by the operations above (which are provided by the system):

Input	**Effect**
deposit	
`account, amount`	Deposits the amount `amount` into the account `account`. The amount can be negative.
reserveActivity	
`reserv, offer, client, number`	Adds the activity reservation `reserv` for activity offer `offer`, client `client` and for the number of persons `number`. This operation decreases the availability of the activity offer with `number`.
cancelActivityReservation	
`reserv`	Cancels the activity reservation `reserv`. (This operation changes the availability of the associated activity offer.)
reserveRooms	
`reservs, rooms client, arrival departue`	Makes a number of room reservations `reservs`, for the rooms `rooms`, for client `client`, at the same hotel, and the same arrival and departure dates (`arrival` and `departure`).
cancelRoomReservations	
`reserv`	Cancels the room reservations `reserv`. (The rooms can now be reserved for someone else.)
makeInvoice	
`inv, client, type, price, tax`	Makes in invoice `inv`, for client `client`, with purchase type `type,` price (of purchase) `price` and tax `tax`.
cancelInvoice	
`inv`	Cancels the invoice `inv`.

It can be assumed that accounts always have associated clients and banks. Then the operation openAccount do not need explicit parameters for client and bank. (The balance attribute, though, may change. So, the operation clientDeposit needs a parameter for the amount being deposited.) The corresponding can be assumed for activity offers, activity reservations, room reservations and adventures.

2.10 Requirements of the Solution

The specifications/operations above should satisfy the following constraints. The most adequate model to define/verify these requirements should be identified:

1. Every account in ACCOUNT can be opened (added to Adventure Builder), as long as it has not been opened yet.
2. Already open accounts cannot be opened again.
3. Each open account belongs to exactly one client.
4. Each open account belongs to exactly one bank.
5. A bank can have several accounts.
6. A client can have several open accounts;
7. It is only possible to deposit in open bank account.
8. The balance of an open account cannot be negative.
9. Open accounts remain open.
10. A hotel can have several rooms.
11. Each hotel room belongs to exactly one hotel.
12. Each hotel room is either singe or double.
13. For each room reservation added to Adventure Builder, the arrival date is less than the departure date.
14. Room reservations for the same room must not overlap.
15. For each activity, the capacity is positive.
16. For each activity offer added to Adventure Builder, the arrival date is less than the departure date.
17. For each activity offer added to Adventure Builder, the availability is always ≥ 0 and $\leq$ the capacity of the corresponding activity.
18. For each activity reservation added to Adventure Builder, the number of people that the reservation is intended for is > 0.
19. When an activity is reserved, the availability of the corresponding activity offer decreases with the number of people that the reservation is for.
20. Activity offers in Adventure Builder remain there.
21. Each adventure in Adventure Builder, is in one single adventure state.
22. For each adventure in Adventure Builder, the number of people that the reservation is intended for is > 0.
23. For each adventure in Adventure Builder, the client is also the client in the corresponding room and activity reservations.
24. For each adventure in Adventure Builder, the corresponding room reservations are at the same hotel.
25. For each adventure in Adventure Builder, the number of people that the corresponding room reservations are for matches the number of people doing the activity.
26. It is only possible to pay for adventures added to Adventure Builder.
27. For each adventure in Adventure Builder, the account that is used for paying for the adventure belongs to the client that the adventure is created for.
28. For each adventure in Adventure Builder, the account that is credited belongs to the broker who arranged the adventure.
29. Only adventures in Adventure Builder that are in state PayedState can be cancelled.

30. Activity reservations in Adventure Builder cannot disappear, unless an adventure is cancelled.
31. If an adventure is payed for, then it has been created.
32. If an adventure is confirmed, then it has been payed.
33. Clients with invoices in Adventure Builder have at least one open bank account.
34. Invoices in Adventure Builder can disappear.
35. If an adventure is payed, then the corresponding invoice is created.
36. An invoice cannot be created, unless the payment has happened.
37. Tax on purchase depends on the kind of purchase and the price, and the tax is always ≥ 0.
38. Annual tax reduction credits exactly one client account for each client with invoices in Adventure Builder.
39. Balances, on open accounts, cannot decrease with annual tax reduction.
40. Balances, on open accounts, can increase with annual tax reduction.
41. Client accounts for which the clients do not have invoices are not affected by annual tax reduction.

2.11 Extra Predicates and Assertions

In the previous sections we described the expected behaviour and gave the interface for the *Adventure Builder*. The development of the model and the dynamics of it, considering the restrictions of Section 2.10), should be identified by the students. You may define other predicates, functions or assertions, not stated here if you feel the need to. You may also need to find reasonable preconditions for the transitions.

3 Verification of Correctness

Each group should verify the developed model. For that, *assertions* and *predicates* should be derived, from the requirements presented in Section 2.10. The Alloy tool (`http://alloytools.org/`) should be used to verify the assertions and find instances to the predicates.

4 Documenting the Code

The final Alloy file should contain the following.

- Each group should verify the developed model. For that, *assertions* and *predicates* should be derived, from each of the requirements presented in Section 2.10, unless the requirement follows trivially from a fact or a declaration. The final file should include a proper **check** `<assertion>` for each such assertion and a **run** `<predicate>` for each predicate. Appropriate scopes should be used.

- For each operation, it should be clearly marked (with comments) which conditions are pre-, post- and frame-conditions.
- Each function/predicate/pre-condition/post-condition/fact/declaration/assertion should be marked (with a comment) which of the requirements are being implemented (if any). It is enough to mark with the number of the requirement. Note that, if an assertion hold because of one (or several) pre conditions (for instance) then both the assertion and the pre condition should be marked with the same number.

Uncommented assertions will be considered as missing. Failing to do the rest of the documentation described above will be penalised in the Quality of the solution.

5 Hand-in Instructions

The Alloy model for this project is due on the **9th of Movember, 2018, 20:00**. You should follow the following steps to hand-in Project 1.

Saving Alloy Project

- Make sure your project is named `AdventureBuilderGXXP11819` in Alloy, where `XX` is the group number, before saving it. Always use two digits, that is, Group 8's project should be named `AdventureBuilderG08P11819`.
- Select *Save As* and chose the filename `AdventureBuilderGXXP11819.als` to save your project.

Upload the file in Fenix Upload the file `AdventureBuilderGXXP11819.als` in Fenix prior to the respective deadline.

6 Project Evaluation

6.1 Evaluation components

In the evaluation of this project we will consider the following components:

1. Correct development of the model and specification of the requisites: 0–12 points. **Predicates, pre-conditions, post-conditions and facts, corresponding to restrictions from section 2.10, must have a comment with the number of the restriction(s).**
2. Correct development (and verification) of the assertions to be proved: 0–4 points **Asertions, corresponding to restrictions from section 2.10, must have a comment with the number of the restriction(s).**
3. Quality, completeness, and simplicity of the obtained solution: 0–4 points.

If any of the above items is only partially developed, the grade will be given accordingly.

6.2 Other Forms of Evaluation

It may be possible *a posteriori* to ask the students to present individually their work or to perform the specification of a problem similar to the one of the project. This decision is solely taken by the professors of ES. Also, students whose grade in the first test is lower than this project grade by more than 5 may be subject to an oral examination.

In both cases, the final grade for the project will be individual and the one obtained in these evaluations.

6.3 Fraud Detection and Plagiarism

The submission of the project presupposes the **commitment of honour** that the project was solely executed by the members of the group that are referenced in the files/documents submitted for evaluation. Failure to stand up to this commitment, i.e., the appropriation of work done by other groups, either voluntarily or involuntarily, will have as consequence the immediate failure of this year's ES course of all students involved (including those who facilitated the occurrence).

7 Useful Development Tips

The system to be developed can be divided into several sub-problems/orthogonal dimensions. Considering one dimension at each time corresponds to dealing with the different levels of abstraction.

One first task should be to try to separate the requisites of Section 2.10 into these orthogonal dimensions and develop the signatures and transitions that are associated with such requisites. One can then start incorporating the other dimensions and build upon these simpler components.

Notice that several models can be developed that satisfy the above properties. You should aim at developing one that satisfies these properties and no other. Use the generated diagrams to guide your development.

You might also need to have a notion of *Time* and *Dynamics* for the model. For that, it might be useful to look at Section 6 of the Alloy book.

8 Final Remarks

All information regarding this project is available on the course's website, under the section *Project*. Supporting material such as links, manuals, and FAQs may be found under the same section.

In cases of doubt about the requirements, or where the specification of the problem is possibly incomplete, please contact the Professors of the course.

Good Luck!