Assignment 02

midf, stmp, ehel

September 2022

Types

1. Class
   A class is a reference type so when you pass it around you pass the reference

2. Struct
   A value type so when you pass it to a function or method you copy it and provide the copy

3. Record Class
   Immutable class

4. Record Struct
   Immutable struct

Extension Methods

1.

```
xs.SelectMany(x => x);
```

2.

```
ys.Where(x => x % 7 == 0 & x > 42);
```

3.

```
ys.Where(x => x % 4 == 0 & x > 1582 & (x % 100 == 0 ? x % 400 == 0 : true))
```

Delegates

1.

```
(string s) => String.Concat(s.Reverse())
```

2.

```
(decimal x, decimal y) => x * y
```

3.

```
(string x, int y) => int.Parse(x) == y
```

Exercise 1

A scenario is a description of a flow of events while a use case is a description of a functional requirement. A scenario is an element of a use case. A fully dressed use case includes a description, actors, main success scenario, and alternative scenarios, etc.

Scenario
A scenario is used to make a very relatable example of how the product is going to be used and describes a users usage of the system from beginning to end. It is used to get a common understanding of the system.

Use Case
A Use Case is used to understand a functional requirement, it describes the actor(s) involved and what parts of the system they can interact with, but does not go into detail of how. The Use Case is thus used to provide common knowledge of each functional requirement.

Exercise 2

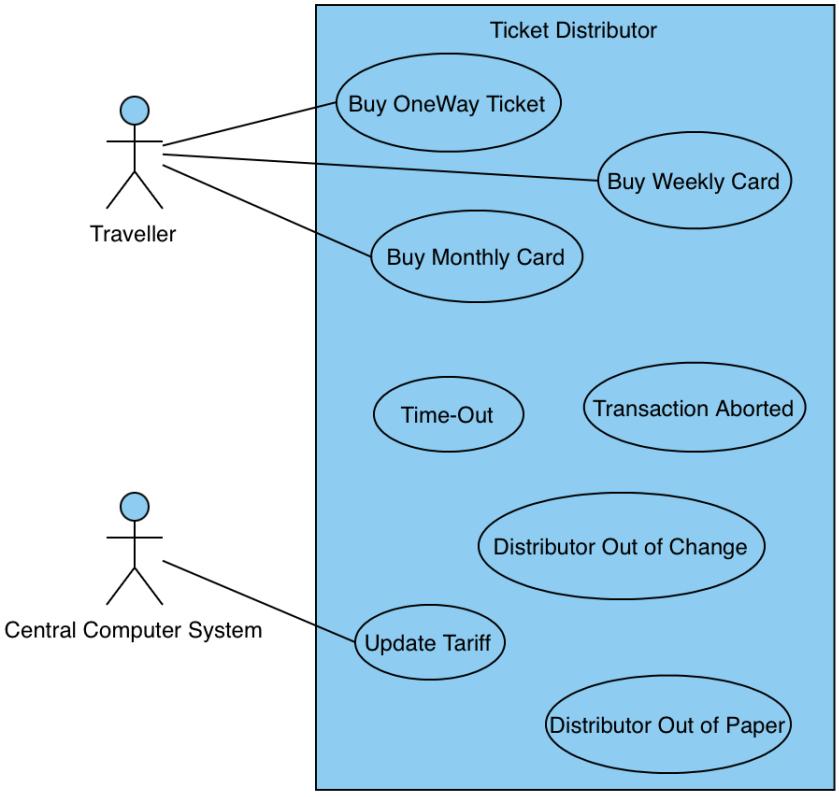
User requirements
Statements in natural language, plus diagrams of what services the system is expected to provide to the system users and the constraints which it must operate under. The user requirements may vary from broad statements of the system features required to detailed, precise descriptions of the system functionality.

System requirements
Detailed descriptions of the software systems's functions, services and operational constraints. The system requirements document (sometimes called functional specification) should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

Functional requirements
Statements of services the system should provide, how the system should react to particular inputs, and how the system should behave in particular situations.

Non-Functional requirements
Constraints on the services or functions offered by the system. They include timing constraints, constaints on the development process, and constaints imposed by standards. Non-functional requirements often apply to the system as a whole rather than individual system features or services.

Exercise 3


Ticket Distributor
Buy OneWay Ticket
Buy Weekly Card
Buy Monthly Card
Time-Out
Transaction Aborted
Distributor Out of Change
Update Tariff
Distributor Out of Paper
Traveller
Central Computer System

Exercise 4

Ambiguous formulations

1. "være medvirkende til, at kommunen er et attraktivt sted at arbejde."
   This is ambiguous because what is "et attraktivt sted"?

2. "hvilket tilsammen kan understøtte en faglig stolthed."
   This is ambiguous because what is "faglig stolthed"?

3. "der understøtter og guider dem i deres daglige opgaveløsning." This is ambiguous because it is not said what "understøtter og guider" exactly is in this situation.

Any missing information?
There is no mention of the UI. There is no mention of how long it should take to learn to use the system.

Problems with their formulation
Since none of the requirements are objective, there is no way to test if the system meets the requirements.

Rewrite the requirement
Medarbejdere skal have et digitalt system der hjælper og undersøtter dem i udførelsen af deres kerneydelser. Denne digitalle understøttelse skal øge tilfredsheden på arbejdspladsen blandt medarbejderne med X procent.

Kommunen har forskellige faglige målgrupper, som møder borgerne forskellige steder, og som arbejder, hvor borgerne er. Derfor er det vigtigt, at store dele af løsningen kan afvikles på mobile enheder.

Løsningen skal effektivisere medarbejdernes arbejde med X procent, via genbrug af data, deling af data og at løsningen sikre korrekt sagsbehandling.

Løsningen skal være intuitiv for kommunens medarbejdere, så den bidrager til deres arbejde fremfor at blive en byrde (dette vil kunne ses på at det effektivisere deres arbejde fremfor at hæmme det).

Exercise 5

Actors
Music Producer

Use cases
UC1 - Add note to track
Main Actor: Music Producer
Main Success Scenario:
1. The music producer changes the selected track using horizontal arrow keys
2. The music producer changes the selected step using vertical arrow keys
3. The music producer adds the note by pressing enter

UC2 - Create random drum notes
Main Actor: Music Producer
Main Success Scenario:
1. The music producer setup a pattern of length 128
2. The music producer highlight the entire track 1 using the shift and arrow keys
3. The music producer use the fill function, and adds a random drum notes to track 1
4. The drum notes are generated and added to the track

UC3 - Delete half of track
Main Actor: Music Producer
Main Success Scenario:
1. The music producer navigates to the middle of the track using the arrow keys
2. The music highlights the first half of the track by holding the shift key and pressing the up arrow.
2. The music producer press the delete key to delete the first half of the track.

Non-functional requirements
1. A typical user should be able to learn the basics of how to use the system within ten hours.

2. The system needs to work with the associated music tracker hardware.

3. The system should never have a loading time of more than 0.1 secs.

Exercise 6

UC1
Main Actor: Customer
Main Success Senario:
1. Costumer selects an item they wish to buy on the monitor
2. The item is added to the cart
3. Customer chooses a payment method
4. The customer pays
5. The system prints a receipt
Extensions:
1.A If the food is from the buffet
1.A.1 The costumer weighs their food on the scale
1.A.2 Go back to step 2
1.B The customer wants to buy another item
1.B.1 Go back to step 1

Requirements:
1. The system needs to support both a Danish and English interface. - Non-Functional

2. The system should accept both card as well as mobile-pay as payment methods. - Non-Functional

3. The system should detect when it does not have enough paper to produce a receipt, and provide an error message until given more paper. - Functional