Your Paper

frai, wihe, hecn

September 22, 2022

1 C#

1.1 Types

1. Class: Can have a constructor and are reference type.
2. Struct: Can't have a constructor and are value type..
3. Record Class: Is similar to a class but the main difference is that all fields are final once initiated
4. Record Struct: Is similar to a struct but the main difference is that all fields are final once initiated

1.2 Extension Methods

1.
   ```
   public static IEnumerable<T> Flatten<T>(this IEnumerable<IEnumerable<T>> items)
   => items.SelectMany(x => x);
   ```

2.
   ```
   public static IEnumerable<int> GetNumbersDivisibleBy7AndGreaterThan42(
   this IEnumerable<int> numbers) => numbers.Where(x => x > 42 && x % 7 == 0);
   ```

3.
   ```
   public static IEnumerable<int> GetLeapYearsOfNumbers(
   this IEnumerable<int> numbers) => numbers.Where(IsLeapYear);

   private static bool IsLeapYear(int year)
       {
           if(year < 1582) return false;
           if(year % 400 == 0) return true;
           if(year % 100 == 0) return false;
           return (year % 4 == 0);
       }
   ```

1.3 Delegates / Anonymous methods

1.
   ```
   public static readonly Func<string, string> Reverse = s =>
   new string(s.ToCharArray().Reverse().ToArray());
   ```

2.
   ```
   public static readonly Func<double, double, double> Square = (x, y) => x * y;
   ```

3.
   ```
   public static readonly Func<string, int> StringToInt = Int32.Parse;
   ```

2 Software Engineering

2.1 Exercise 1

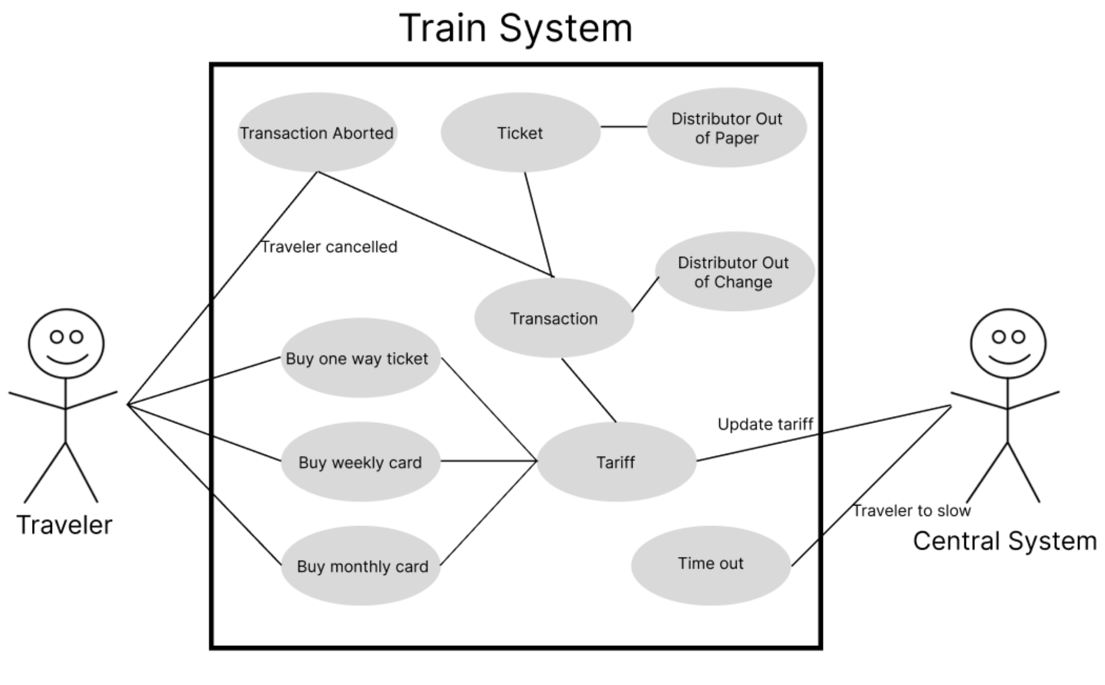
- **Scenarios** are often used when the customer is having trouble giving an abstract description of the desired system requirements. A scenario is like a story where the customer will use a real-life example explaining how and when a system should do something. A scenario is more detailed than a story otherwise very similar.
- **Use Case** are used to visually show the interactions between an actor and the product, but can be to difficult for stakeholders to understand when talking requirement engineering. They are unfamiliar with the term use case and don't find the graphical model useful. Use cases can be more helpful when working with system design.
- The main difference is that a scenario will use a real-life example when explaining the requirements and nothing more visual, where a use case is more advanced because they use high-level use case diagrams and also have a textual description similar to a scenario. So some would say that a use case have a scenario for every requirements and potential edge cases.

2.2 Exercise 2

A computer system may by subject to several non-functional product requirements such as efficiency, dependability, security and maintainability requirements.

- Efficiency requirements set expectations to how fast the system should operate and respond to input, as well as limiting the space used for internal memory.
- Dependability requirements are met to ensure that the software does what it is expected to do, without failures. Further, the software should be available for use when required.
- Security requirements are met to make sure that no hostile outsides has access to sensitive or private information. Outsiders tempering with a system's data can have severe consequences for many different stake holders.
- Maintainability requirements are met to make sure the software can be easily adapted to meet the changing needs of the customer.

2.3 Exercise 3



2.4 Exercise 4

- Discover formulations in the requirement that are ambiguous.

  ”[...] til at effektivisere tunge arbejdsgange via genbrug af data, deling af data og ikke mindst, at løsningen er medvirkende til god lovmedholdelighed i sagsbehandlingen, hvilket tilsammen kan understøtte en faglig stolthed.”

  The above statement holds non-functional requirements for the system. The statement, however, does not really have meaning. ”genbrug af data” is really hard to interpret for the developer. So is ”deling af data”; in what ways do they want to share data? How much data? ”God lovmedholdelighed” is almost impossible to translate into the problem domain whithout further explanation.

  The statement ”En stabil og driftssikker løsning” is also a somewhat vague statement and could use further explanation.

- Is there any information missing in the requirement?

  In general more specific constraints on how the requirements should be met would help the developer.

- The requirement specifies a set of non-functional requirements. What is problematic about there formulation?

  A requirement for the system states that the employees should enjoy their work place more because of the integration of the new IT-system. This is in no way testable and makes it hard for the developer to find a solution to their problem.

- Rewrite the requirement according to what you identified as problematic in the three bullet points above.

  Our formulation of how to integrate a user friendly interface would incorporate more concrete solutions, such as how few clicks it must take to reach a certain point in the system, or what features needs to be easily accessible.

2.5 Exercise 5

Identify actors that interact with a music tracker software system.
An actor that interact with a music tracker software system could be a game developer as said in the video. Other examples of actors could be anyone who intend to create old school music or general looped background music.

Formulate three use cases in structured language that a software music tracker system has to support.

UC1 Produce music
Main Scenario

1. User turns on device
2. User assign an instruments to the 8 tracks
3. User hits play
4. System plays music added to the 8 tracks

UC2 Change number of steps
Main Scenario

1. User press length button
2. System changes number of steps

3. User can choose any amount from 12 to 128

UC3 Fill function
Main Scenario

1. User highlights steps to fill
2. User press fill and euclidean
3. User choose amount of events
4. User choose chromatic
5. User choose from and to notes
6. User press fill to fill

Express three non-functional requirements for a music tracker software system

Requirement 1
The music tracker software system should
have a simple and easy to understand UI, so that a new user can easily get started with creating a track

Requirement 2
The music tracker software system
should support advanced shortcut for the experienced user to efficiently create new tracks from scratch

Requirement 3
It should be possible for a new user to create a mix with 4 tracks involved within 30 minutes with no prior experience.

2.6 Exercise 6

2.6.1 Use Case

Summary: People in the canteen, who wants buffet, weigh their food and proceeds to pay with credit card on a weigh-in machine. People who wanted to pay for snacks, drinks, or prepackaged food needed to be serviced manually by a worker/owner.
The main focus of this use case will be the buffet, weigh-in and payment.

Use Case	
Title	User can weigh and pay for their own buffet
Primary Actor	User
Goal and Scope	The actor should be able to take buffet, weight and pay without any manual interaction from owners or workers of the canteen.
Initiation event	User enters the canteen, wanting buffet.
Starting condition	The canteen needs to be open 9:00-15:00 and buffet will only be served at dinner which is from 11:30 to 13:00
Story	The user enters the canteen and proceeds to get a plate. Next they take the food they want from the canteen and go to the payment platform. They choose buffet on the system and place the plate of food on top of the scale. The price is shown to them and they can choose to press ‘pay‘ to continue. If they agree they can use their credit card touch less or by inserting.
Final Result	User has paid, gotten their food and is happy.
End State	1 plate less food.

2.6.2 Requirements

Based on the use-case above we propose the following 3 requirements:
Two of them which are functional:

1 User should be able to pay for food without the aid of any other actor.

2 The user should be able to weigh their food and know how much it will cost.

And one which is non-functional:

3 The user should go through the weigh and payment process less than 2 minutes.