Assignement 01 - BDSA 2021

Dagrún Eir Ásgeirsdóttir and Albert Rise Nielsen

16th september 2021

Contents

C#

1.1 Generics

In the first snippet, the value of type T, has to be comparable which is necessary to check if one value is greater than another. In the second snippet the addition of "where T : U" adds the constraint to T that it has to implement or inherit the type U, which in turn has to be comparable.

Software Engineering

2.1 Exercise 1

The aquisition of knowledge can result in invalidating previously aquired knowledge.

2.1.1 Concrete example

Writing a test, and an accompanying implementation, which you think will work, then having the test fail, would disprove the hypothesis of the implementation passing the test.

2.2 Exercise 2

“The ticket distributor is composed of a user interface subsystem, a subsystem for computing tariff, and a network subsystem managing communication with the central computer.” is part of the system design.
“The ticket distributor provides the traveler with an on-line help.” is part of the requirements.

“The ticket distributor will use PowerPC processor chips.” could be both. It could be an external requirement to use the exisiting infrastructure of the client, or it could be a part of the system design infrastructure.

2.3 Exercise 3

"Assume you are developing an online system for managing bank accounts for mobile customers. A major design issue is how to provide access to the accounts when the customer cannot establish an online connection." is part of the application domain
"One proposal is that accounts are made available on the mobile computer, even if the server

is not up. In this case, the accounts show the amounts from the last connected session." is part of the solution domain.

2.4 Exercise 4

A bridge and an aircraft have a very well defined domain in common. Both have been built for year, have well defined and simple use cases as well as a very specific user base.
Word was a newly designed application within a completely new domain, with the whole world as it's user base. So it had to be designed for everyone and as many use cases as possible.

2.5 Exercise 5

"The TicketDistributor must enable a traveler to buy weekly passes." is functional.
"The TicketDistributor must be written in Java." is nonfunctional.
"The TicketDistributor must be easy to use." is functional.
"The TicketDistributor must always be available." is nonfunctional.
"The TicketDistributor must provide a phone number to call when it fails." is functional.

2.6 Exercise 6

Specifying the applications domain, function, architecture, etc. To determine what is important within the scope of the application/system, and how and what to implement it.