BDSA2021 - Week 2 Exercises

milb@itu.dk - jodi@itu.dk - kong@itu.dk

September 2021

1 C#

1.1 Github Repository

https://github.com/Akongstad/Assignment1.git

1.2 Generics

The first statement indicates that T needs to implement type IComparable. This means that T and U aren't necessarily the same type and therefore can't be compared.

The second statement indicates that both T and U needs to implement the type IComparable. When both T and U are the same type of IComparable, it ensures that they are guaranteed to be comparable.

2 Software Engineering

2.1 Exercise 1

What is meant by "knowledge acquisition is not sequential"? Provide a concrete example of knowledge acquisition that illustrates this.

When modeling the application and solution domain, software engineers turn

gathered information and data into knowledge. The acquisition of this knowledge is not linear, as single pieces of data can make preexisting knowledge and models useless.
Example: Once upon a time humans believed earth was the center of the universe and thus scientific models and ideas were built around this being a fact. When the theory was debunked all the previous knowledge and scientific models were invalidated.

2.2 Exercise 2

Specify which of the following decisions were made during requirements or system design:

- *"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."*
  Part of system design. Technical description of system elements and their interactions.

- *"The ticket distributor will use PowerPC processor chips."*
  Part of requirements. The system will run on specific hardware.

- *"The ticket distributor provides the traveler with an on-line help."*
  This sentence also falls under 'requirements'. It is written as a functional requirement because it states a service the system provides.

2.3 Exercise 3

In the following description, explain when the term account is used as an application domain concept and when as a solution domain concept:

*"Assume you are developing an online system for managing bank accounts (**Application domain**) for mobile customers. A major design issue is how to provide access to the accounts (**Application domain**) when the customer cannot*

*establish an online connection. One proposal is that accounts (**Solution Domain**) are made available on the mobile computer, even if the server is not up. In this case, the accounts (**Solution domain**) show the amounts from the last connected session."*

The first 2 uses of "Account" portray the objects and relations. The last 2 transforms the established application domain into the solution domain

2.4 Exercise 4

A passenger aircraft is composed of several millions of individual parts and requires thousands of persons to assemble. A four-lane highway bridge is another example of complexity. The first version of Word for Windows, a word processor released by Microsoft in November 1989, required 55 person-years, resulted into 249,000 lines of source code, and was delivered 4 years late. Aircraft and highway bridges are usually delivered on time and below budget, whereas software is often not. Discuss what are, in your opinion, the differences between developing an aircraft, a bridge, and a word processor, which would cause this situation.

The difference between building an aircraft and a software solution, lies in the uniqueness of the project. The blueprint and model is already laid out when assembling an aircraft. When building a software solution, even tho you might be able to reuse parts, there will always be an unknown part that needs to be fitted to exact problem domain.

This unknown part makes it hard to estimate a delivery time and a budget when building software. Furthermore, you do not have a physical and visual representation of the progress when building a software solution.

2.5 Exercise 5

Specify which of these statements are functional requirements and which are *nonfunctional requirements:*

"The TicketDistributor must enable a traveler to buy weekly passes."

- Functional requirement

"The TicketDistributor must be written in Java."

- Non-functional / Domain requirement

The TicketDistributor must be easy to use.

- Non-functional

The TicketDistributor must always be available.

- Non-functional (not realistic)

The TicketDistributor must provide a phone number to call when it fails.

- Functional requirement

2.6 Exercise 6

What is the purpose of modeling?

Modelling has been widely used in all fields of science. Modelling is used to generate an abstract and understandable/easier to work with visual depiction of a system. When building software it is not necessary to become an expert in the field of the problem domain, but only understand enough to be able to build a model a of the application domain in order to ultimately build a model of the proposed solution domain.