Report assignment 1

IM0102171811
Ivo Willemsen
851926289

Part 1: Problem analysis

Introduction

An application needs to be designed that enables a user to manage **references** and generate reference. References are stored in a **data store** (for example database or file**).** So the application needs to be able to read and save to the data store. The information in the data store is arranged according to a certain **storage format** (for example bibtex, EndNote, XML).

So at a use-case level, the following actions can be identified:

Action	Actor
Read references from data store	Client
Save references to data store	Client
Generate references	Client

Entities and attributes

The following entities can be identified by reading the case description. This section doesn't focus on responsibilities and actions, it only focusses on the identification of entities and attributes.

Client

That's the application that initiates the use-cases on behalf of the user. It doesn't have any specific attributes, it doesn't store any state.

Reference

A reference consists of the following common concepts:

1. A label which serves an interface. A document can identify the reference to be included in the document by using the label.
2. A list of fields. Each field consists of a name and value. According to the case description ("…. and so forth."), it can be deduced, that there are no predefined set of mandatory fields, this depends on the type of the reference
3. A type. A reference is of a certain type (for example book, journal article, paper in proceedings, thesis, etcetera)

The case description mentions that the difference tween one type of reference and another, is the name of the type and which fields are considered when the tool checks whether the references is complete.

I think it's only the name of the type of the reference, the fields don't matter. Take the example of two types of references with different names that have the same fields. The type of the references determines the order in which the fields are displayed. That can be different in both situations, but that doesn't depend on the fields, it depends on the type of the reference. So in this situation, what makes one type different than the other is only the name of the type. You could have two different types of references with a different name, but same fields and same rule for ordering, but they are still two different types.

Field

A field consists of the following common characteristics:

1. A field has a name and a value
2. The case description mentions several constraints regarding fields. Different terms are used that are a bit vague ("generated", "printed", "should have a value", "free", "completeness "). I will take some time to elaborate on these aspects (to create uniformity), as they are probably important for the design that follows. Of the list of fields that exist in the list, with respect to the generation (printing) of the reference list, there are either mandatory fields (I will use the adjective "**generated**" going forward) or non-generated (I will use the adjective "**free**" going forward) fields. It can be extracted from the case description that a field is either a generated field or a free field; there are no other types of fields
3. A field has a display format, but this is not an attribute of the field itself, it's a derived attribute that depends on the style. Also a display format comes into play only during the generation phase. It's of no importance during the reading and saving of references (and fields) to and from the data store
4. A field has an order in the list of references. Again, this is not a property of the field itself, but depends on the type of the reference

Data store

The case mentions that the references will be stored in a either a database or a file. The concept is called **data store**, and examples of a data store are files and databases, or perhaps even a web service. This seems to suggest that there must be some option (client ?, configuration ?) that decides which type of data store to use, although the case description doesn't mention it. But it's good to identify the existence of more than just one data store type, as this could impact the design at later times when this design needs to be made more flexible!

Storage format
References are stored according to a certain **storage format**. Examples of formats are bibtex, EndNote and XML.

Style
A style consists of the following common characteristics:

1. A style consists of a name
2. A style determines how certain fields are displayed. So this means that each style must know the fields that are displayed. Knowing how fields must be displayed means that there is a notion of fields that belong to the style. And for every field, an enumeration must be available (italic, bold, normal). So a style has a **list of fields**, and for every field a **display format** must be specified (italic, bold, normal)

Rules

The case description mentions that the style determines for each field how it is displayed. Every field can be displayed in a different way. This means that the display of a field is a function of the name of the field and the style.

According to the case descrpiption, the type of the reference determines the order of the fields. So, the order of a field in a reference is a function of the name of the field and the reference type.

Part 2: Design

Part 3: Patterns

Part .4: Design decisions

Part 5: Future changes