Group
Stefan Brütsch

Software Engineering
Solution Exercise 3

24. November 2017
Maximilian Fischer (799511)[1]
Wilhelm Kerle (763284)[2]
Manuel Prinz (748340)[3]

Software Requirements Specification:

Software: "Digital Zoo"

28. November 2017

[1]maximilian.2.fischer@uni-konstanz.de
[2]wilhelm.kerle@uni-konstanz.de
[3]manuel.prinz@uni-konstanz.de

Task 1: UML Case Tool (0P)

Done, using Papyrus.

Task 2: Requirements Specication (38P) 35/38

The Requirements Specification is detailed on the following pages.

Note *We focus only on the ten requirements chosen and on related areas, which necessarily form only a subset of the requirements of the complete software requested in the initial e-mail.*

Inhaltsverzeichnis

1 Introduction

1.1 Purpose

The creation of a zoo managing software. The software should be used in the daily work and support management work, especially in terms of budget and food-order.

1.2 Scope

The software product will be an online, cloud based platform. It enables keepers to manage and control the food ordering and handle delayed orders.

1.3 Definitions, acronyms, and abbreviations

SRS Software Requirement Specification

IEEE Institute of Electrical and Electronics Engineers

1.4 References

Three informal letters of requirements showing the perspectives of zoo-keeper, zoo-director and secretary as provided in assignment 3.
Additionally the email from Susan Tapir of assignment 2.

1.5 Overview

The SRS is categorized in three sections. The first section gives an introduction, which defines the purpose of the system and the scope of the final software product. It also includes a list of abbreviations and relevant definitions. The second section of the SRS is a non-formalized description of the software product developed for the customer. The third section lists the specific requirements according to IEEE-Standard and illustrates the system workflow use-case diagrams and sequence-diagrams.

2 Overall description

2.1 Product perspective

The zoo director wants a system to organize the zoo which can be used by all employees. It has especially focus on work schedules, the regulation of the zoos budget and the feeding of the animals.

2.2 Product functions

The product helps the secretary to organize the work schedule for the employees. The system therefore provides the secretary with the possibility to view the employees information. Also the secretary is provided a view into the monthly budget that is planned for personnel. Furthermore the secretary can request an increase of the budget for employees.
Just as the secretary the zoo keepers can request for increased budget if it is not enough for the according animal. The zoo keepers can view the monthly budget for the animals they are responsible for and can purchase feed through the system.
The product shows the requested budget increases to the zoo director just as it shows the total budgets. The zoo director can alter the monthly (reserved) budget or accept or deny requests for more budgets with the system.

2.3 User characteristics

The users of the software system are the zoo-director, the secretary and the zoo-keepers. All of them show average or sub-average computer skills, so the software shall be easy to use.

2.4 Constraints

The developer is not allowed to actually order feed to test this function.

2.5 Assumptions and dependencies

The employees do have devices to run the application.
Also everyone has knowledge of their area of responsibility. Therefore they only have to be made familiar with the usage of the product.
The product itself does not yet know which feed is mandatory for which animal, so correct handling depends on the knowledge of the users.

2.6 Apportioning of requirements

The system itself cant find out which person has which role (secretary, zoo keeper...) by itself, so it has to be fed with that information by the zoo director.

Bei apportioning geht es um die Abgrenzung, was nicht getan werden soll (also wenn genannte requirements nicht umgesetzt werden. Sprich "out of scope"). Dass das System das nicht weiß, ist etwas anderes.

3 Specific requirements

3.1 External interfaces

Sequence Diagram The sequence diagram is shown in figure 1. It describes the default ordering procedure. The two interacting entities are a zookeeper and the system. Firstly the user has to log i to use the software. The system asks for the login data. The zookeeper then enters his username and password and confirms by pressing the login-button.
The system then computes the password-hash checks if the a user with this username and password-hash exists. If such a user exists the system confirms the login-data and shows the main software. When the zookeeper clicks the button new orderthe software shows the offers of the whitelisted feed-dealers.
Now the zookeeper can select the offer he wants and confirm the order. The System then sends the order and confirms the user that the order was sent.

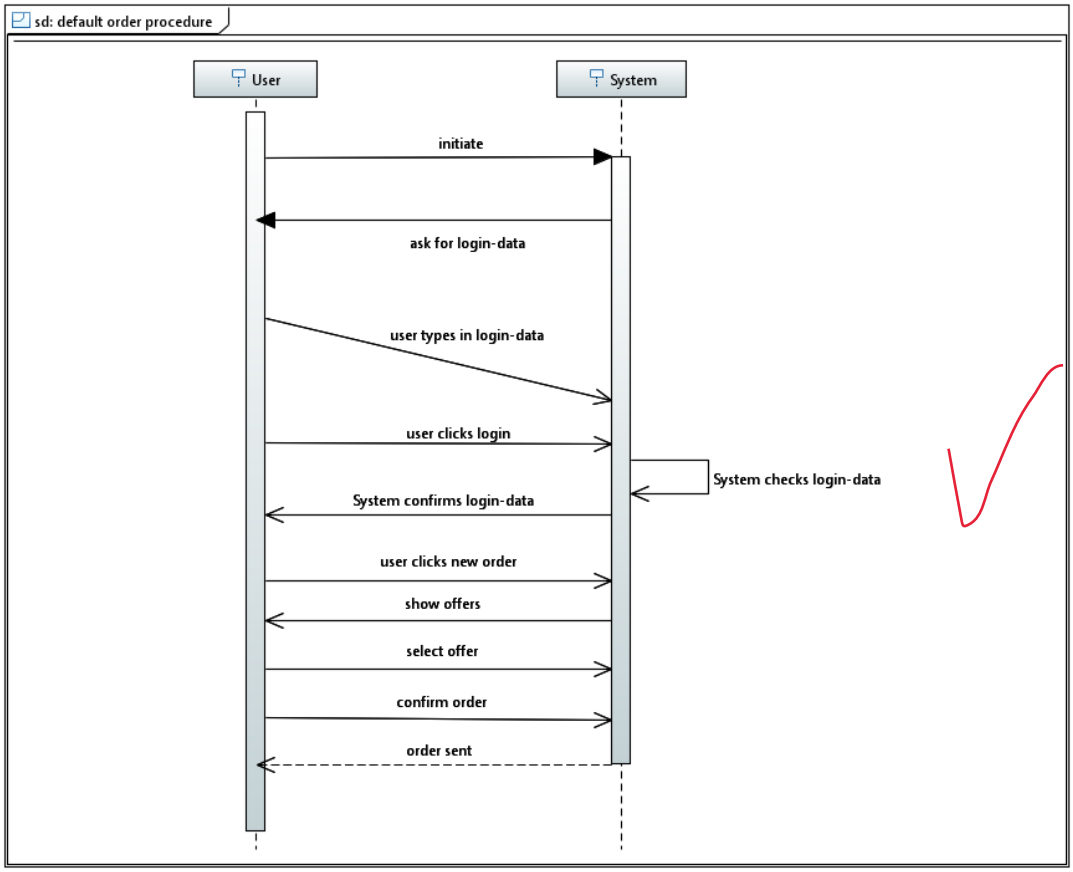

Abbildung 1: Sequence diagram.

Use Case Diagram The use case diagram is shown in figure 2. It describes the usage of the system by various users. The users first log in to verify their roles. A zookeeper and the secretary both can access the overview of orders. All the relevant information about the orders is listed in this view like the price quantity and type of the products.
A zookeeper can select an offer and order it. The secretary has the option to delete orders they select. All usecases directly or indirectly include the login-usecase because it the users have to be logged in to view sensible information, order or delete an order.

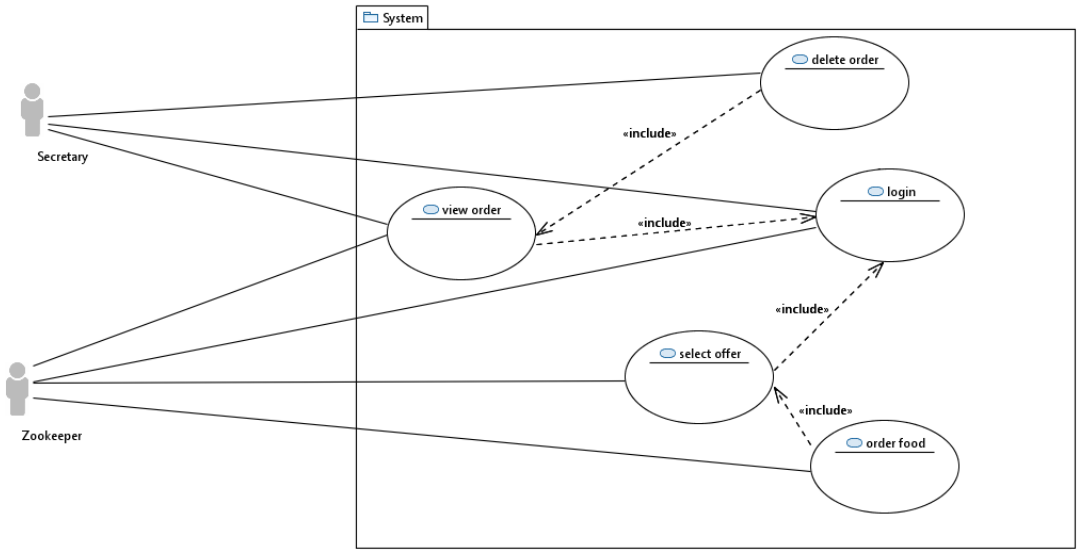

Abbildung 2: Use case diagram.

Class Diagram The use case diagram is shown in figure 3. It models how the different parts of the system and the different actors are related and interact. Every user has an username and a password-hash and are able to login. Zookeeper and secretary are users. A zookeeper additionally is associated certain animals.
The whitelisted feed dealers have a company name. They create offers which contain price, weight and product-type. A zookeeper can choose an offer and create an order. The order consists of an offer and data about the delivery and the animal the product is for. The secretary has the ability to cancel every order.

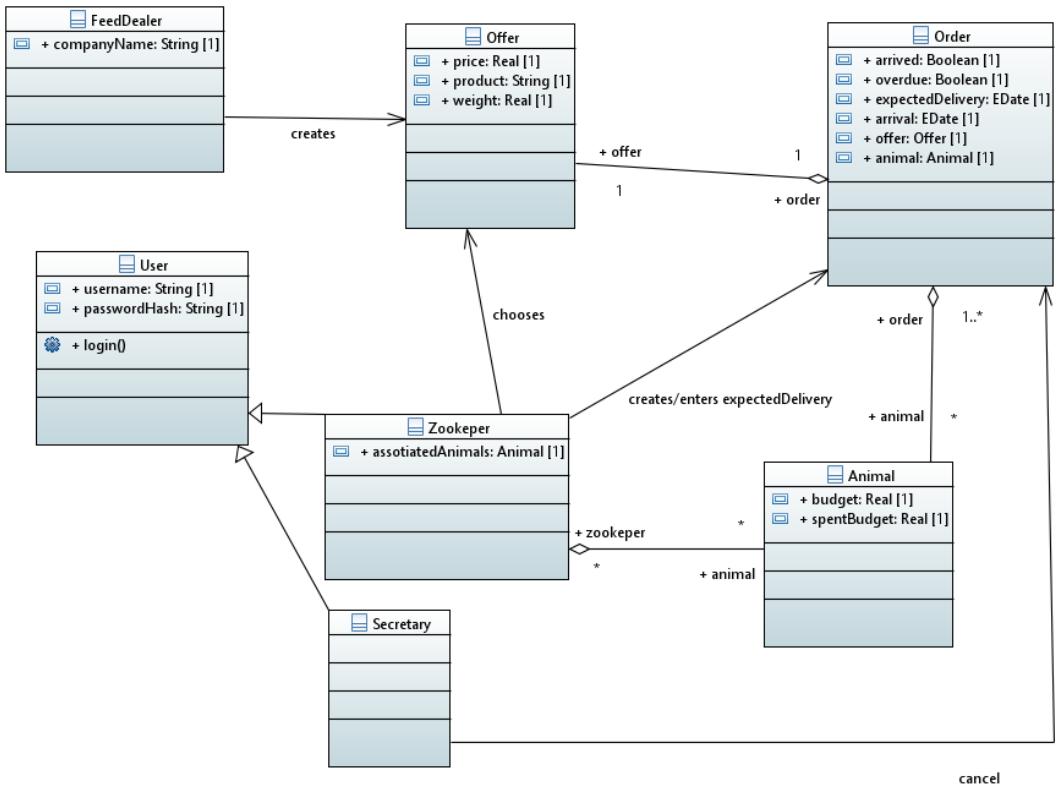

Abbildung 3: Class Diagram.

3.2 Functions

R1: AnimalBudgetOverview
Function: The software *shall* display how much of the budget was already spent on an animal.
Description: The zookeepers have a monthly budget for every animal so they need to track how much money was already spent this month for a certain animal.
Therefore the budget will be deducted by the system if an order is being paid or increased if a budget increase was approved. The overview which is accessible by the zookeepers will show the information about the budget they need.
Source: Zoo-keeper perspective
Dependency: none

R2: FeedOrdering
Function: The system shall provide the option to order feed from whitelisted feed dealers.
Description: The zookeepers can order food using the software. To do so they can access the list of feed dealers which the zoo director has compiled. Once they spotted the feed dealer with the best offer, they can issue an order with the system which then will drop the order in the feed dealers inbox as an email.
Source: Zoo-keeper perspective
Dependency: none

R3: DealerChoice
Function: The software *must* offer the choice of different food dealers for an order *Description:* Before ordering the zookeepers can choose the best offer in the software. The list of allowed feed dealers is created by the zoo director.
Source: Zoo-keeper perspective
Dependency: R2

R4: EasyUse
Function: The software *must* be easy to use
Description: The secretary is is not good at using the computer. Therefore the software has to be as self-explanatory as possible. This also shrinks the time to get used to the software.
Source: initial letter of the zoo director
Dependency: none

R5: MarkDelayedOrders
Function: If an order does not arrive on time the system *must* marke the order as "overdue".
Description: If an order does not arrive on time the stock of the ordered food may be lower than desired. Therefore the order will be marked 'overdue' to notify the staff.
Source: Zoo-keeper perspective
Dependency: R2, R4

R6: DeleteOrders
Function: The software *must* contain the functionality that orders can be deleted by the secretary.
Description: An order may be sent by mistake and has to be taken back. The secretary has the manages all orders and can cancel all of them. When an order is deleted the feed dealer has to be notified.
Source: Zoo-keeper perspective
Dependency: R2

R7: ArrivedCheckbox
Function: The software *shall* contain a 'arrived'-checkbox.
Description: When the order arrives the zookeeper receiving the food checks this checkbox to mark this order as delivered. It is a critical information whether an order has already arrived. This has to be entered manually because automation would be hard and costly. So this checkbox is the way of reciving this information.
Source: Zoo-keeper perspective
Dependency: R2

R8: MarkDelayedOrders
Function: If an order does not arrive on time the software *must* mark is as 'overdue'.
Description: If an order does not arrive on time (the 'arrived'-checkbox is not checked after the expected delivery date) the stock of the ordered food may be lower than desired. Therefore the order will be marked "overdue" to notify the staff.
Source: Zoo-keeper perspective
Dependency: R2, R4, R5, R7

R9: ArrivalDate
Function: The Software *must* enter a date of arrival when the 'arrived'-checkbox is checked.
Description: The date of arrival needs to be stored for the protocol and to plan future orders. This is also necessary for the taxes and the general records.
Source: Zoo-keeper perspective
Dependency: R2, R7

R10: SecureLogin
Function: The software *shall* provide a secure login.
Description: The software contains monetary information and enables ordering of big deliveries. A secure login is necessary to prevent unauthorized access.
Source: Zoo director perspective
Dependency: R2, R7

3.3 Non-Functional Requirements

Copy-paste. Ich habe es schon beim vorigen ÜB angemerkt ;)

R11: Cost
Function: The software *shall not* be too too expensive.
Description: The zoo is small and therefore the software has to be affordable.
Source: initial letter of the zoo director
Dependency: none

3.4 Design constraints

Especially the overviews are supposed to be graphical.

3.5 Software system attributes

- Reliability
  R12: ReliableBudgetAccounting
  Function: The invoice total will only be deducted from budget once the order has arrived.
  Description:
  Source: Zoo-keeper perspective
  Dependency: none

- Availability -

- Security
  R13: DataSecurity
  Function: All the data must be stored securely.
  Description: The data contains monetary information therefore it must be stored securely and only accessible in the software while logged in.
  Source: Zoo-keeper perspective, initial letter of the zoo director
  Dependency: R10

- Maintainability -
- Portability
  R14: MobilePortability
  Function: The software should be portable to mobile devices.
  Description: The software should run on mobile and tablet.
  Source: Zoo-keeper perspective, initial letter of the zoo director
  Dependency: none

3.6 Organizing the specific requirements

-

3.7 Additional comments

-

Task 3: Self-evaluation (2P) 2/2

Max One aspect which complicated things is that we mostly had to stick to the original request, while having an intuitive understanding of what was actually requested or meant. As usual with large projects, managing the people involved is not trivial, especially when different levels of knowledge, commitment, working speed and high demand on the result are present.

Wilhelm Another perspective we had to shed light upon and that becomes complicated quickly is that we of course try to focus on as few requirements as possible - but not less than required - and still want to create a result that satisfies the criteria of a proper and well designed SRS paper.

Manuel In my opinion the greatest challenge was to choose the requirements in a way to prevent overloading the diagrams. The four texts describe multiple systems which should be merged into one big project. Therefore the requirements had to be chosen of one project to limit the size of the diagrams.

Points
37 / 40 P