Software Specification (2IX20) 2021–2022
Assignment 1: Requirements and UML

Updated 9 February 2022

The purpose of this assignment is to prepare a first model of a software system. This includes both a (non-formal) requirements specification, as well UML models for several aspects of the system.

1 Introduction

Because this assignment concentrates on the specification and documentation of requirements, and developing UML models, instead of eliciting requirements from clients or stakeholders, the problem domain is one that you should already be familiar with. You are expected to take into account your own knowledge of the problem domain in completing this work.

You are to complete this exercise in teams of 2, as registered in the "project groups" on this course's Canvas page. When working in teams, your colleagues may have differing views about the requirements and models. You should reach some kind of consensus about the requirements and modelling choices, and document them as well as you can. There is more than one possible "right answer" for this project.

Figure 1: Tjongerkanaal Turfroute locks.

2 System description

Study this overview of the system you are to specify.

A lock is a construction consisting of two doors that cannot be opened at the same time. Ships can either be moved up or down. We show an example of a simple lock in Figure 1.[1]

In some cases, a passage requires passing through a sequence of locks. Sometimes, these are spread over a longer distance, such as in the Panama canal, where ships need to pass through six locks (three up and three down). In other cases, multiple locks are in close proximity. In these situations, we refer to this as a flight of locks.

In this assignment, we consider a flight of locks that are used to move ships between different water levels.

2.1 Lock system

For this assignment, a flight consisting of a sequence of n locks is to be installed in a canal, so there are $n + 1$ different water levels that ships can move between. Figure 2 shows a flight of five locks.

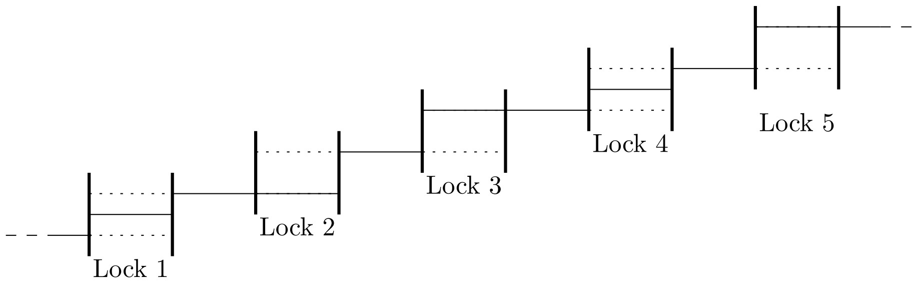

Figure 2: A flight of locks

The doors of the locks and the control mechanism of the doors and the slides are supplied by the manufacturer. The internal mechanisms of these are assumed (given). What you need to design and model is the control unit, and the corresponding logic to change the water levels and open the doors allowing ships to pass through the system of locks according to the constraints described below.

In Figure 3, we show the kind of lock that is part of the flight of locks, and that should be controlled by the control software that we need to specify.

Each *lock* has two pairs of doors. On one side of the lock, the water level is *low*, and on the other side of the lock, the water level is *high*. Each pair of doors contains a *slide* that can be opened or closed. If the slide in the high doors is opened, water is allowed to enter the lock, raising its water level. If the slide in the low doors is opened, water flows from the lock, lowering its water level. A door can only be opened if the water level on both sides of the door is equal. Opening and closing slides not only results in a variation of the water level inside the locks. It also results in a variation of the water level between the locks. The variation in the water level between the locks must be bounded.

[1]H.P.Burger, CC BY 3.0, via Wikimedia Commons

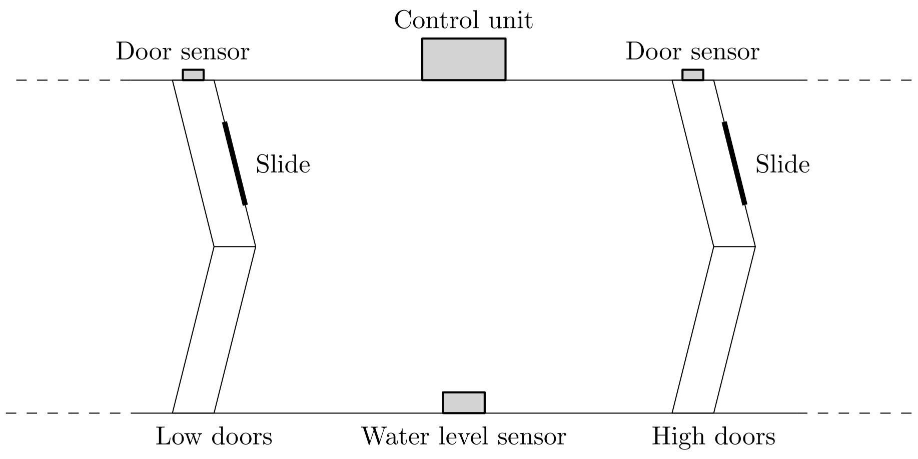

Figure 3: A simple lock system

=actor

The lock keeper (Dutch: sluiswachter) controls the locks using a control panel. On this control panel, there is an *open button* and a *close button* for every set of doors in the lock system. Every door has a *sensor* indicating whether it is closed or open. Each lock has a *water level sensor* indicating the water level inside the lock.

The *control unit* runs the *control software* that operates the lock.

2.2 Control unit

As described before, you are to model the control unit that interfaces with the locks systems described before. The control unit receives requests from the control panel and then starts opening or closing the doors of the locks. Using the model(s) you create, it should be possible to study how a ship is allowed to move through the lock system.

The interface of the control unit is given in Figure 4. The control unit is connected to the doors, slides, and sensors from the locks, as well as the control panel.

The control unit displayed in Figure 4 is very simple, since it is connected to only one lock. Your objective is to model a system with n locks.

2.3 Robustness and unexpected actions

The description given above focuses on the normal behavior of the lock controller. Once you have a model that describes the normal behavior, you should extend the model with exceptional behavior. For this, think about what can go wrong, and how you could model this.[2]

We next give some ideas of exceptional situations that you can consider, but you may of course use your imagination to add more or to model others.

For instance, you may wonder what should happen if an emergency happens while opening or closing the door. Can the door be stopped while it is opening

[2]This will surely give you extra fun (and of course, depending on how well you execute it, extra points).

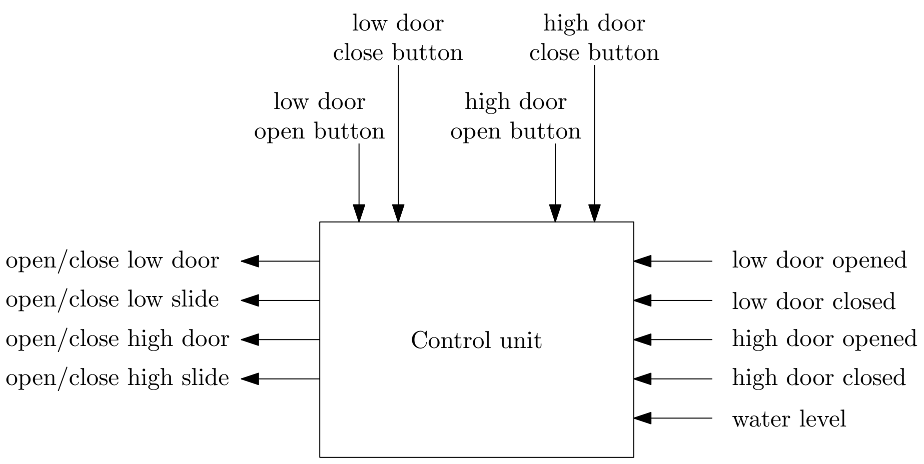

Figure 4: Control unit interface

or closing? You may also consider what happens, e.g., if a door does not close, or a slide does not open.

3 Assignment

The description in the previous section gives a general idea of how the lock control software works. Although fairly precise, once you start working on the requirements and the models, you will observe that a lot of things are unclear. When developing software as a software engineer, you will discover that this is a fact of life. Things that are unclear are typically resolved by talking to the stakeholders in the requirements elicitation process.

You may be familiar with the functioning of locks. You can use your own experience to help you understand the details of the system for which you are trying to specify the requirements. You can also discuss the lock systems with your tutor in the tutorial sessions, essentially relying on the tutor as customer or domain expert. Using your own background knowledge or information that you uncover while discussing with the domain expert, you may make assumptions and design decisions in order to interpret the description in the previous section. You are also allowed to deviate from the interface of the control unit described above. For instance, you are allowed to decide that additional water level sensors are required. **In all situations where you make assumptions or design decisions, or where you deviate from the description in this assignment, motivate your assumptions and document your design decisions in the report.**

When modelling the system, keep in mind that your models should not depend on specific numbers of locks. If, somehow, you cannot avoid bounding the number of locks, it is required to support at least 4 locks.

In this assignment, you will create a report that covers the following topics:

1. Requirements of the control software
2. Use cases and use case diagrams for the control software

3. A class diagram for the control software
4. A state machine diagram for the control unit
5. A sequence diagram
6. An activity diagram

Each of these will be specified in more detail below.

3.1 Deliverables

The deliverable for this assignment is a report that contains the requirements and all the models, as a **single PDF file**. The assignment is to be handed in using the submission page in Canvas. Models should be drawn using **UMLet** or its online version **UMLetino**. For readability, it is allowed (but not required) to include the .uxf files for UMLet when you hand the assignment. In that case, clearly refer to them in your report. Files not included in the report or without reference from the report will not be considered when grading. Make sure you give a textual explanation for each of the models you include in the report.

3.2 Tasks

You should complete each of the following tasks, and include the result of the task in your report (1 section for each of the tasks).

1. Describe the requirements of the control software. Make sure to take into account the guidelines described in the reader. Make sure you first identify the system. You should identify at least 4 actors. You shall include at least 7 requirements. The control software shall follow the interface described in Figure 4. To define actors and inputs and outputs, you probably need to extend the interface to deal with exceptional behaviors.
2. Define and organize the use cases. You should follow the methods described in this course. Make sure you extract the use cases from the requirements (and possibly the informal description). Organize the use cases in a use case diagram, and write a detailed description for all the use cases defined in the use case diagram. Your use case diagram shall contain at least 4 actors and at least 4 use cases. Furthermore, it shall contain at least one “include” and one “extend” relationship.
3. Model a class diagram for the control software. Make sure you also include an informal description of the class diagram that motivates the most import design choices.
4. Give UML state machine diagrams that capture (the most important parts of) the behavior of the control unit. Your report shall contain at least 2 state machine diagrams describing different aspects of the control unit.
5. Give a sequence diagram describing the sequence of the interactions needed to allow a ship to move from the low side of the first lock to the high side of the fourth lock under the assumption that no exception occurs. Also describe how this sequence diagram relates to other diagrams in your report.

6. Give an activity diagram for the use case where a ship arrives at the high side of the fourth lock, and it must be allowed to move to the low side of the third lock.

3.3 Reflection

At the end of your report, include a section in which you reflect on the assignment. This should consist of the following:

1. Description of difficulties you encountered, or any questions you still have.
2. Brief description of the contributions of both team members. Should it turn out that there are large differences between the contributions of the team members, this may affect the individual grades.