One-week SBE course for water locks supervisory control

de Feijter, Ilias I.A.

Award date:
2022

Final bachelor project

One-week SBE course for water locks supervisory control

Quartile 3-4 - 2021-2022

Full Name	Student ID	Study
I.I.A. de Feijter	1470590	Mechanical Engineering

Supervisors: Dr.ir. J.M. van de Mortel-Fronczak, Em.prof.dr.ir. J.E. Rooda

Eindhoven, July 10, 2022

Abstract

Rijkswaterstaat,[24], is the executive agency of the Dutch Ministry of Infrastructure and Water Management. It is responsible for the construction, management, design, and maintenance of infrastructural facilities in the Netherlands, e.g. bridges, tunnels, and water locks. Such systems are called cyber-physical systems (CPS) and are essential parts of the infrastructure, especially in a country like the Netherlands, with many rivers and waterways[24]. It is important for the economy and the safety of people that such systems operate safely and as intended. Therefore, a supervisor that removes all dangerous behaviour from the system must be developed[8]. Synthesized-based engineering, SBE, [8] provides the method to develop such a supervisor correctly, and the Eclipse Supervisory Control Engineering Toolkit (ESCET™)[7], developed by Eindhoven University of Technology, provides the tools to execute the method. Rijkswaterstaat currently outsources its supervisor development for water locks to external companies. However, it is important that its engineers know how SBE works such that they know what they are talking about and can have a critical look at the provided supervisors from the external companies. This project proposes a one-week course in which an engineer familiar with water locks learns how to use ESCET™ to develop supervisory controllers for water locks. The result is a course consisting of 10 modules, of which the first four have been developed in this project. The modules consist of a theory section, quizzes that test whether the participant has understood the theory, an ESCET™ section explaining how the theory is applied in ESCET™ , exercises, and the water lock case. The water lock case goes through the process of synthesizing and testing supervisors for water locks, from defining the system to testing the supervisors on models. In previous projects, similar cases have been applied in courses with success [14][12]. Participants of those courses reported that the cases made them understand the theory better. The course developed in this project explains the development process of supervisors in a general way and focuses on water locks by implementing the water lock case. However, the course focus can be changed by replacing the water lock case with another case, e.g. a tunnel case.

Contents

1 Introduction **3**
- 1.1 Project context . . . 3
- 1.2 Project objective . . . 4
- 1.3 Report outline . . . 4

2 Course setup **5**
- 2.1 Course profile . . . 5
- 2.2 Course structure . . . 6
- 2.3 Water lock case . . . 8

3 Literature **11**
- 3.1 Supervisory control theory . . . 11
- 3.2 Learning theory . . . 11

4 Building Website **15**
- 4.1 Website . . . 15
- 4.2 Figures . . . 15
- 4.3 Recommendations website . . . 16

5 Conclusion and Recommendation **17**
- 5.1 Conclusion . . . 17
- 5.2 Recommendation . . . 17

References **18**

Appendices **19**

A Project files **19**

B Website files instructions and contents **19**
- B.1 Open the website . . . 19
- B.2 Change the content and style of the website . . . 19
- B.3 Create an exercise or assignment . . . 19
- B.4 Create a quiz . . . 20
- B.5 Other files . . . 21

C Latex files instruction **22**
- C.1 Make figure . . . 22
- C.2 Change figure settings . . . 22

D CIF files contents **23**

1 Introduction

This chapter introduces the project by giving the project's context and objective. At the end of the chapter, the outline of the report is provided.

1.1 Project context

Rijkswaterstaat,[24], is the executive agency of the Dutch Ministry of Infrastructure and Water Management. It is responsible for the construction, management, design, and maintenance of infrastructural facilities in the Netherlands, e.g. bridges, tunnels, and water locks . Such systems are called cyber-physical systems (CPS), [11] which are mechanical systems controlled by a computer-based algorithm. Figure 1 shows a schematic overview of such a system.

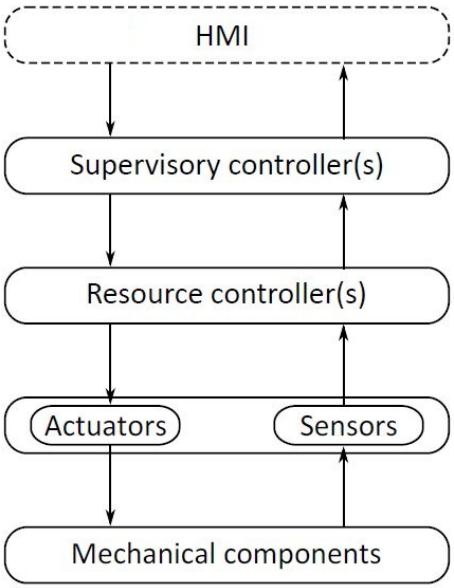

Figure 1: Schematic overview of a cyber-physical system

A CPS consists of mechanical components, actuators, sensors and resource controllers. The actuators of a CPS actuate the mechanical components, and sensors provide the feedback on the position or state of the mechanical components to the resource controllers that control the actuator's movement. Based on the feedback from the sensors obtained via the resource controllers, the supervisory controller (supervisor) can disable the movement of the actuators via the resource controllers. The supervisor is between the human operator of the CPS and the CPS itself to ensure the safe and desired operation of the system. The supervisor does not allow dangerous behaviour to occur in the system. For example a supervisor disables opening a water lock gate when the water level at both sides of the gate is not equal.

Therefore, these supervisors must be correct for the safe operation of infrastructure and the safety of people. A correct supervisor starts by making the supervisor in a proper systematic way that minimizes human errors from occurring in the process. The synthesis-based design process synthesizes a supervisor, with an algorithm, out of a discrete-event (plant) model, also called plant model, of the CPS and a model of the requirements[8]. A discrete-event model only contains all the states a system can be in and the transitions between these states without taking time into account, where the system modelled is called a discrete-event system [8]. This synthesis algorithm incorporates all supervisor requirements correctly into the supervisor [8]. When synthesized, the supervisors are validated and tested on a model of the CPS to ensure it only allows desired behaviour of the CPS and removes dangerous behaviour. When the testing shows that a requirement is wrongly defined by the engineer or an additional requirement is needed, it is easily solved by changing the requirement model and re-synthesizing the supervisor. When the supervisor is finished,

PLC code can be generated to control the actual CPS or a digital twin. This way of designing supervisors significantly reduces the development time and increases the quality of supervisory controllers, and is also referred to as Synthesis-Based Engineering (SBE) [8].

Eindhoven University of Technology (TU/e) has developed the Eclipse Supervisory Control Engineering Toolkit (ESCET™)[7], which provides the tools for this synthesis-based design process. The toolkit includes the modelling language, editor and simulator, and tools to perform synthesis and code generation. With ESCET™, supervisors can be synthesized and tested with simulation and PLC code can be generated from them.

1.2 Project objective

Rijkswaterstaat currently outsources its supervisor development for water locks to external companies. However, it is important that its engineers know how this synthesis-based design process works such that they know what they are talking about and can have a critical look at the provided supervisors from the external companies. Therefore, this project's goal is to propose a one-week course in which an engineer familiar with water locks learns how to use ESCET™ to develop supervisory controllers for water locks.

1.3 Report outline

This report is structured as follows. Chapter 2 provides and substantiates the course setup, the elements of the course and the platform on which the course is provided. Chapter 3 discusses the literature consulted for making the course setup and gaining the knowledge to make the course content. The chapter substantiates, with the literature, the choices made for the course elements and structure. Chapter 4 explains the tools used to build the platform on which the course is provided. Finally, Chapter 5 provides the conclusion and a summary of the recommendations for a subsequent project.

2 Course setup

Before developing the course content, it is important to consider the course setup, including the course profile and structure, to prevent discrepancies between the initial course objective and the final course. By clearly defining the course profile, including the aim and learning outcomes, at the start of developing the course, the course content required to meet the course profile can be easier defined. The structure of this content and how it is presented in the course is defined in the course structure. Defining the course structure correctly ensures that all content required to meet the course profile is present. This chapter gives the course setup and the reasoning behind it.

2.1 Course profile

2.1.1 Assumed knowledge

The participants of this course are assumed to have a university of applied sciences or a university degree at a technical level, meaning that the participants can think in an abstract and problem-solving way. This course is written for the supervisory control development of water locks using ESCET™. So knowledge of water lock control and behaviour and programming is recommended but not required.

2.1.2 Aim

This course aims to enhance and enlarge engineers' understanding and knowledge of supervisory control theory and discrete-event systems. Furthermore, this course aims to teach the participants how to use ESCET™ (Eclipse Supervisory Control Engineering Toolkit) to implement the theory in practice to obtain a proper supervisory controller of a system. In this project, the course focuses on supervisors for water locks. However, the course is set up in such a way that the focus can be changed to another kind of system. How the course focus can be changed to another application than water locks is explained in Subsection 2.2.2.

2.1.3 Target group

The course is written for engineers of Rijkwaterstaat familiar with water locks. Therefore, special attention to water lock is given in the practical application part of the course. The course, however, explains the development of a proper supervisory controller in a general way. So the knowledge gained from this course is widely applicable to a large field of applications and is therefore also useful for people with a different background.

2.1.4 Learning outcomes

At the end of the course, after thoroughly studying the course modules and doing the exercises and assignments, the participant is expected to:

- Know and understand discrete-event systems. The participant knows what a discrete-event system is, what the properties of such systems are, and what their abstraction levels are.
- Know the definition of automata with associated terms: locations, events, transitions, variables, and initial and marked states.
- Be able to apply the synchronous composition on automata.
- Know concepts encountered in supervisory control synthesis, like non-blockingness, controllability, and maximal permissiveness.
- Know the basic supervisory control problem and be able to apply the synthesis procedure manually on a simple uncontrolled plant to gain a supervisory controller.
- Know how automata and state-based expressions specify a requirement model.
- Know how to use ESCET™ and the CIF language.
- Be able to use ESCET™ to make a model of the uncontrolled plant and of the requirements.
- Be able to use ESCET™ to synthesize a supervisory controller from the plant and requirement model.
- Be able to use ESCET™ and Scalable Vector Graphics (SVG) visualization to validate the synthesized controller.

- Develop a controller for a water lock by using ESCET™.

2.2 Course structure

2.2.1 Course platform

The course is provided via a website. This website is straightforward and enables an interactive learning experience in which theory and practice alternate. A website gives much freedom in designing the course. Almost everything is possible and can be added to a website. It enables interaction in the form of quizzes and creates structure by having multiple web pages. Currently, the website is a sort of a text file with some quizzes and exercises that give feedback to allow interaction. However, in a subsequent project, more features can be added to the website (see Subsection4.3. Furthermore is a website easily shared with participants both offline and online. The website is elaborated on in Chapter 4.

2.2.2 Learning activities

The project objective states that the course should require one week, meaning 40 hours, to be completed. The decision was made to divide the course content into 10 modules of 4 hours, each requiring 1.5 hours of studying the theory and 2.5 hours of doing the exercises and assignments. This setup gives the participant the power to plan his learning path. Ideally, the participant finishes one module every day, but he can choose to do two modules a day or only a part (theory or exercise) of a module.

Each module starts by explaining the theory that is divided into sections. At the end of each section, a small quiz is provided to test whether the participant has understood the theory. After the theory part, the ESCET™ section explains how the theory is applied in ESCET™ . After the ESCET™ section, exercises are provided, which are more challenging than the quizzes and require a better understanding of the theory. The module ends with making a part of the water lock case. This case takes the participants through the process of synthesizing and testing supervisors for water locks. The participants can make the case by making assignments with the same difficulty level as the exercises. The next chapter presents the scientific substantiation for the chosen learning activities.

As discussed earlier in this chapter, is the course written in a general way such that it can be used for multiple applications. The water lock case makes this course aim for supervisor synthesis of water locks. The theory, quizzes, and exercises explain the supervisory control theory in a general way with arbitrary examples and exercises. The advantage of this is that the course focus can be changed from water locks to, for example, tunnels by changing the water lock case with a tunnel case. The tunnel case still has to be made, but the rest of the course can remain unchanged.

The CIF models of the exercises, water lock, and the examples used in the theory part can be found in A, and their contents are elaborated in D.

2.2.3 Course modules

The first four modules were developed in this project. The participants know at the end of module 4 how they should make supervisors of discrete-event systems and how they can test their supervisor on a discrete-event model with SVG visualization. In a subsequent project Modules, 5 through 10 can be developed. Recommendations for the contents of these modules are provided in Subsection 2.2.4. Below, a list with the contents of Modules 1 through 3 is provided.

- **Module 1: Introduction and discrete-event systems**
  Automata are used to make discrete-event models [20] and consist of states and transitions between the states, representing the same states and transitions of the system it represents. The state of an automaton is a combination of a location and all the variables' values. The transitions of an automaton are labelled with events [20].

  This module introduces the course by giving the course guide, a general introduction, and how it is set up. Furthermore, it gives an overview of what supervisory controllers are and their design processes. Also, discrete-event systems, automata, locations, variables and events are explained.

- **Module 2: Network of Automata**
  Automata can be reachability, blocking and have deadlock. The term reachability means all states are

reachable from the initial state. An automaton has deadlock when from at least one of its states, no transitions can be taken and is blocking when at least one of its states has no sequence of transitions ending in a marked state. A marked state is a state defined by the system's modeller to be stable. [20]

To model a system, a discrete-event model can contain multiple automata, called a network of automata. Variable and events can be shared between the automata of a network. The network of automata has to synchronize on shared events. To convert a network of automata into a single automaton with the same behaviour, the synchronous composition is applied on the network of automata. [20]

This module explains the terms reachability, deadlock and non-blocking. Furthermore, how automata interact with each other, shared variables, synchronization on events, and synchronous composition on automata are explained.

- **Module 3: Plant model and basic supervisory synthesis procedure**
  The basic supervisor synthesis procedure allows to remove deadlock and blocking behaviour in automata, resulting in a supervisor that is controllable, non-blocking and maximally permissive. A supervisor is controllable when it does not disable uncontrollable events. Uncontrollable events are events the supervisor cannot disable, defined by the system's modeller. A supervisor is non-blocking when the system under the supervisor's control is not blocking. A supervisor is maximally permissive if it only disables events for the purpose of removing blocking behaviour. [20]

  This module explains the basic supervisory synthesis procedure and how one can make a supervisor for a simple uncontrolled plant model by explaining the basic supervisory control synthesis algorithm and applying it to simple problems. To do this, the terms controllability, non-blocking, and maximal permissiveness are explained.

- **Module 4: Requirement model and SVG**
  The discrete-event model can contain undesired behaviour which is not blocking. To remove such behaviour, requirements are specified. Requirements can be specified by automata or state-based expressions [20]. To validate and test synthesized supervisors, SVG visualization can be used [3].

  The final result of this module is that the participants can model the plant and requirements in ESCET™using the correct CIF language and tools. Furthermore, the module explains two types of requirement specifications: requirements specified by automata and state-based expressions. Furthermore, SVG visualization is explained to validate the synthesized supervisors.

2.2.4 Recommendations for Modules 5 through 10

To properly test supervisors, the time must be added to the plant models, making the models hybrid instead of discrete-event [8]. As an alternative to 2D SVG visualization, 3D digital twins of the system can be used to test the supervisors, giving a closer representation of reality. In a subsequent project, it is recommended to fill the additional modules by explaining how to make hybrid models and connecting the CIF models to a digital twin to test the supervisors. The water lock case can be extended to support these modules. The case models can be used immediately to make hybrid water lock models. A digital twin has not yet been made of the system.

Erick Hoogstrate has started making a configurator with which a digital twin can relatively fast be made for a specific water lock by defining its components [9]. This configurator is not yet finished at the time of writing, but during a subsequent project, it possibly is. If so, the course can explain how this configurator works, and as practice, the participant can make a digital twin of the water lock case with this configurator. Below, a list with recommendations for the contents of Modules 5 through 10 is provided for a subsequent project.

- **Module 5**
  This module could be dedicated to explaining hybrid models and how they are made. The participants already know the CIF language needed for hybrid modelling. However, they should understand the difference between discrete-event and hybrid models, the implications for modelling and for what purposes both are used.

- **Module 6**
  Possibly, one module is not enough. If so, this module can be used to finish hybrid models. The rest of the module can explain an organized file structure, like in [17], and how the supervisor derived from the discrete-event plant model is connected to the hybrid model.
- **Module 7**
  With the project of Erick Hoogstrate, this module could be dedicated to explaining how a digital twin is made. A digital twin of the water lock case can then be made as practice.
- **Module 8** Possibly this module is also needed to explain digital twin production.
- **Module 9** This module could be dedicated to explaining PLC code generation out of the supervisor and how to connect this PLC code to the digital twin for testing.
- **Module 10** This module can be used as a buffer if a topic requires more space and time to be explained.

Note that the course profile and structure need to be updated if the recommendations are implemented. For example, the learning goals as they are now would not suffice anymore when hybrid modelling and digital twins become part of the course.

2.3 Water lock case

The course explains the development process of supervisory controllers, with a focus on water locks. The water lock case makes that this course focuses on water locks and allows the participants to apply their knowledge into practice on a problem that is representative of problems they encounter in their profession. This case goes through the process of synthesizing and testing supervisors, from defining the system to testing the supervisors on models. This section presents the setup of the water lock case and the reasoning behind it.

The end product of the case is a synthesized supervisory controller of a simplified version of the Westsluis of the Terneuzen lock complex, shown in Figure 2. This lock is chosen due to personal preference of the course developer. The actual Westsluis consists of 5 sliding gates, two at each end and one in the middle. In this case, this is simplified to two gates, one at each end, two bridges with traffic barriers and traffic lights (horizontal two lights) at each end, two pairs of culverts responsible for filling and draining the chamber, two pairs of entering traffic lights (vertical three lights) and two pairs of leaving traffic lights (vertical two lights). In Figure 3, an overview of the simplified Westsluis is shown.

Figure 2: Westsluis located in Terneuzen. Image obtained from scheepvaartkrant.nl

At the start of the course, the participants receive a CIF file containing the uncontrolled plant model [17]. The SVG drawing shown in Figure 3 and the tooldef file to simulate the model with the SVG drawing is also

provided. The participants can play around with these files and detect that this system needs control for safety reasons. The CIF file contiaining the uncontrolled plant model is, when opened, not informative for a starting participant of this course, so they cannot 'cheat'. In Modules 1 through 3, the participants re-make the CIF model of the uncontrolled plant step by step. In Module 4, the participants add the requirements to synthesize a supervisor for the Westsluis.

In each module, a part of the case is finished with the knowledge they gained in that module. The participants define the system and its components in the first module. Which components the system has and what their discrete behaviour is, is answered. They have to see that the system consists of two gates, two bridges, two pairs of culverts, two pairs of entering/leaving/bridge lights and two pairs of traffic barriers. Of these components, they make simple models in CIF that do not yet include models of subcompenents like sensors and actuators. For example, the model for a gate is an automaton with two locations: Closed and Open.

In the second module, the participants learn about networks of automata. They apply the learned theory to the case by extending their simple CIF models made in Module 1, with automata representing subcomponents, like sensors and actuators. The participants learn that each component contains sensors to give feedback on the system's state and actuators to change the system's state [8]. At the end of this module, they have modelled all components with sensors and actuators just like in [17].

The water lock case in Figure 3 has symmetry; the north (right side in the figure) and south (left side in the figure) sides have the same components. The west (above in the figure) and east (below in the figure) sides also show symmetry for all components except the bridge and gate. Therefore, it is burdensome to model all components separately. Automaton and group definitions, and instantiations can be used in more than one component model to define similar automata[3]. The participants learn in the third module how automaton definitions work and apply them to make a complete model of the water lock case with the component models they made in Module 2. At the end of this module, the plant model is ready to add requirements.

Module 4 explains requirement modelling and SVG visualization. With this knowledge, the participants model the requirements and synthesize a supervisor with their plant and requirement model.

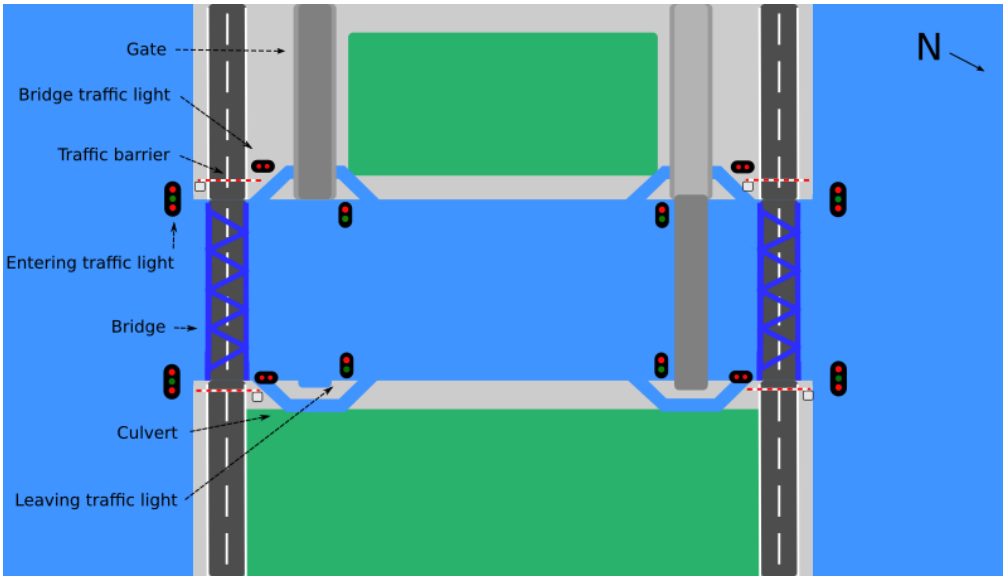

Figure 3: SVG drawing of a simplified version of Westsluis used for the water lock case

2.3.1 Possible extensions

The water lock case now only models the bridges, gates, entering/leaving/bridge lights, traffic barriers and culverts. It would be beneficial if there also would be a sensor giving feedback on the water level. If the water level is not equal at both ends of the gate, it cannot open. This requirement can not be met if there is

no sensor. Sensors that give feedback on whether a ship is lying in front of the bridge or gate would also be a good addition to the case. If a ship is in front of a bridge or gate, that bridge or gate cannot close. This requirement can also not be met without a sensor. Due to a lack of time, these sensors were not added to the case.

A ship can be added to the SVG visualization to make the case more representative of reality and more interactive. This ship can be spanned at either side of the lock to pass the lock. At the start of the course, when the uncontrolled plant and SVG visualization is provided to the participants, they can instruct the ship to move and can operate the uncontrolled lock. When a dangerous situation occurs, for example, the bridge is instructed to close when a ship is underneath it (for this, a sensor is needed), the visualization spans a text box with sinking ship. Just like when a gate or culvert is opened when the gate or culvert at the other side is not closed, the SVG visualization spans flood. The latter is already incorporated in the SVG visualization. By adding this kind of feedback to the SVG visualization, the participants can find all dangerous behaviour of the uncontrolled lock and define the requirements themselves more easily. This will also enhance their learning as they find the requirements themselves by doing and observing [15], which is elaborated in Chapter 3.

At the end of Module 4, this added ship in the SVG visualization can also aid in testing the supervisor the participants made. With that ship, the participants can more easily see whether the lock behaves correctly and that all dangerous behaviour is prevented.

In a subsequent project one can also decide to give the hybrid models and SVG visualization at the start of the course to give an even more realistic experience to the participants. It is then, however, maybe not so clear to the participants why the discrete-event model has to be modelled in the first 4 modules. If the discrete-event SVG visualization, instead of the hybrid SVG visualization, is given at the start of the course, it is clear to the participants that at the end of Module 4, the obtained models are not yet realistic because no time is added, and that hybrid models are the following step in the process.

3 Literature

This chapter presents the literature that was consulted to make this project. In addition, this chapter explains how the knowledge needed to make this course was obtained. The chapter consists of two sections. The first focuses on how theoretical knowledge about supervisory control theory was obtained. The second section is about how the knowledge of how the course needs to be set up for optimal learning was obtained. This section gives the scientific substantiation for the course setup chosen in the previous chapter.

3.1 Supervisory control theory

At the start of this project, the course 4TC00[3] was already followed and passed successfully. This gave a good basis for learning Supervisory control theory and its implementation in ESCET™. Because this course explains automata and the CIF language, the CIF language was already mastered at the start of this project which was useful. Therefore, it is recommended that the next person working on this project follows this course. The website belonging to this course [3] is a good guide for the CIF language and was consulted during this project.

Previously an educational toolbox has been developed for supervisor synthesis using Matlab Simulink[12]. This toolbox was used in a course with success. Students reported that they learned SBE while enjoying it. This also inspired this project and the developed water lock case.

A literature study was performed to get acquainted with supervisory control theory. One paper about the application of supervisory controllers in the Prinses Marijke Complex [8] was provided at the start of the project. This paper was useful to read as it gave a good explanation of the use of supervisory controllers and a glimpse of how the controllers are derived. Sources [1],[21], [4] and [19] of [8], were also worthwhile to read. The other sources of [8] are about the mathematical background, which is not relevant for this project. Source [17] of [8] are the CIF files of the application in the Prinses Marijke Complex. These files are useful in understanding how to use ESCET™ to develop supervisory controllers and structure the files. These files also were the inspiration for the water lock case.

The synthesising of supervisory controllers only applies to discrete event systems (DES). Source [4] introduced DES and was consulted when writing the course. However, only Chapter 1.3 about DES was used in the course to explain DES. The rest of the book is also interesting to read but goes into quite some detail for this project.

The rest of the theory explained in the course was gained from the lecture notes of the course 4CM30 [20]. The setup of the modules is based on the setup of these lecture notes because these lecture notes explain the supervisory control synthesis in a clear and structured manner. However, some parts of the lecture notes are not relevant for this project and are left out.

3.2 Learning theory

3.2.1 Andragogy

The participants of this course are adults. Because children and adults learn differently, the way the educational activities are organised in this course should be adjusted for adults. The way children learn is defined by Pedagogy, literally meaning the art and science of teaching children. Andragogy, on the other hand, is the art and science of teaching adults, first introduced by Malcolm Knowles in 1980 [13].

Because the brains of children and adults differ, they also have different learning styles and views on learning. In [2], researchers found that teachers believed adults to be significantly more intellectually curious, motivated to learn, willing to take responsibility for their learning, willing to work hard at learning, clear about what they want to learn, and concerned with the practical applications and implications of learning than were children and adolescents. This has implications for the organisation of education for adults.

When designing education for adults, 10 adult learning principles should be tried to be applied. These principles reflect the needs of adult participants in education. Table 1 gives the 10 principles and how they can be implemented into education [6].

Table 1: 10 adult learning principles[6]

Principle	Application
1. Adults have accumulated a foundation of life experiences and knowledge.	Connect life experiences and prior learning to new information.
2. Adults are autonomous and self-directed	Involve participants in the learning process, serving as a facilitator and not just a supplier of facts
3. Adults are goal-oriented.	Create educational programs that are organised with clearly defined elements, clearly showing how the program will help participants reach their goals.
4. Adults are relevancy-oriented and practical.	Help learners see a reason for learning something by making it applicable to their work or other responsibilities of value to them.
5. Adults (all learners) need to be respected.	Acknowledge the experiences that adult participants bring to the learning environment, allowing for opinions to be voiced freely.
6. Adults are motivated to learn by both intrinsic and extrinsic motivation.	Show learners how the learning will benefit them and create a comfortable and appropriately challenging learning environment.
7. Adults learn best when they are active participants in the learning process.	Limit lecturing and provide opportunities for sharing of experiences, questions, and exercises that require participants to practice a skill or apply knowledge.
8. Not all adults learn the same way.	Accommodate different learning styles by offering a variety of training methods (e.g., group discussion, role-playing, lecturing, case studies, panel/guest expert, games, structured note-taking, individual coaching, demonstration, and variation in media used) and by using visual and auditory.
9. Adults learn more effectively when given timely and appropriate feedback and reinforcement of learning.	Provide opportunity for feedback from self, peers, and instructor.
10. Adults learn better in an environment that is informal and personal.	Promote group interaction.

Most of these principles have been applied in the course and guided the making of the course setup. Below, the application of the principles is elaborated on.

Principle 1: Because most of the participants of this course are engineers familiar with water locks, principle 1 is applied. The course connects the participant's previous life experiences with water locks with new knowledge about synthesising a supervisory controller of water locks with ESCET™.

Principle 2: By letting the participants follow the course at their own time and pace and providing enough exercise material, they are in control of their own learning process.

Principle 3: The course is structurally divided into 10 modules, and each module has clear elements: theory, quizzes, ESCET™, exercises and water lock case. These elements are elaborated on in Chapter 2.

This structure show the participants how this course helps them reach their goals.

Principle 4: This course applies to the participant's work, giving the participants a reason to follow this course properly.

Principle 5: The participant's experiences are acknowledged in the fact that the course is written at a technical level, requiring technical knowledge. The rest of principle 5 does not apply to this type of online education.

Principle 6: The water lock case challenges the learning process and shows the participants what is possible with the theory learned. This hopefully motivates the participants to learn.

Principle 7: The participants can practice using ESCET™ with the water lock case and the exercises. The rest of principle 7 does not apply, as the course is taken individually and does not directly allow for sharing experiences and asking questions.

Principle 8: Hopefully, the different elements provide enough different teaching methods to accommodate most participants learning styles. Participants that are more theoretically oriented will like the theory elements and quizzes. More practically oriented participants will like the exercises and water lock case. The reviews of ex-participants of the course will reveal whether this principle is applied correctly.

Principle 9: The quizzes aim to test whether the participant has properly understood the theory presented in the sections. The answers to all exercises and assignments for the water lock case are directly available after completion. These elements enable the participants to improve and learn as effectively as possible and act as a feedback mechanism.

Principle 10: This principle does not apply to online education.

7 out of 10 principles (principles: 1, 2, 3, 4, 6, 8, 9) are correctly applied in the course. Principles 5, 7, and 10 are partly or not applied in the course because this course is given online and taken individually. This makes it hard to allow interaction between participants to improve the learning process. In a subsequent project incorporating a discussion forum into the website can be looked into.

So adults learn differently than children. The course has adult participants and should therefore be set up based on andragogy by applying the 10 adult learning principles, which for principle 1, 2, 3, 4, 6, 8, 9 was possible so far.

3.2.2 Cone of experience

Many learning activities can be chosen when designing education to transfer the knowledge to the participants. Some activities, for example, are lectures, exercises and giving a presentation. Not all activities result, when completed, in the same amount of knowledge retention in the participants. The cone of experience, [15] shown in Figure 4, shows the level of retention and the category of learning activity. The participants do not retain much information when they learn through abstraction. They retain more when they learn through observation, by attending lectures and by looking at pictures. The participants learn the most by doing, applying the theory in practice, and doing exercises. Therefore, this way of learning should also form the majority of the course.

From this pyramid, one could conclude that learning by doing is the best way to learn. However, it is stated that a combination of the three ways of learning is the best [15]. If one way is dominant, it can obstruct the process of meaningful generalisation. For example, a mathematician could not develop a system of higher mathematics by counting on his fingers (learning by doing) only.

The course consists of theory divided into sections that need to be read (learning through abstraction), with examples (learning through observation) demonstrating the theory. Each section is closed with a quiz (learning by doing), and after the last section, exercises are provided (learning by doing). Once the exercises are completed, a part of the water lock case is done (learning by doing).

These learning activities correspond to the theory presented in [15]. The participants encounter time-wise the most activities that make them learn by doing (quizzes, exercise and water lock case). Most of the theory is supported with a demonstration of an example to explain the theory, learning through observation. Learning by abstraction and reading theory is encountered the least, time-wise.

The water lock case is especially useful to effectively transfer the knowledge to the participants as it directly applies the theory in practice to a real-life simulation. Moreover, such a case has been applied in a similar course before, with success [14].

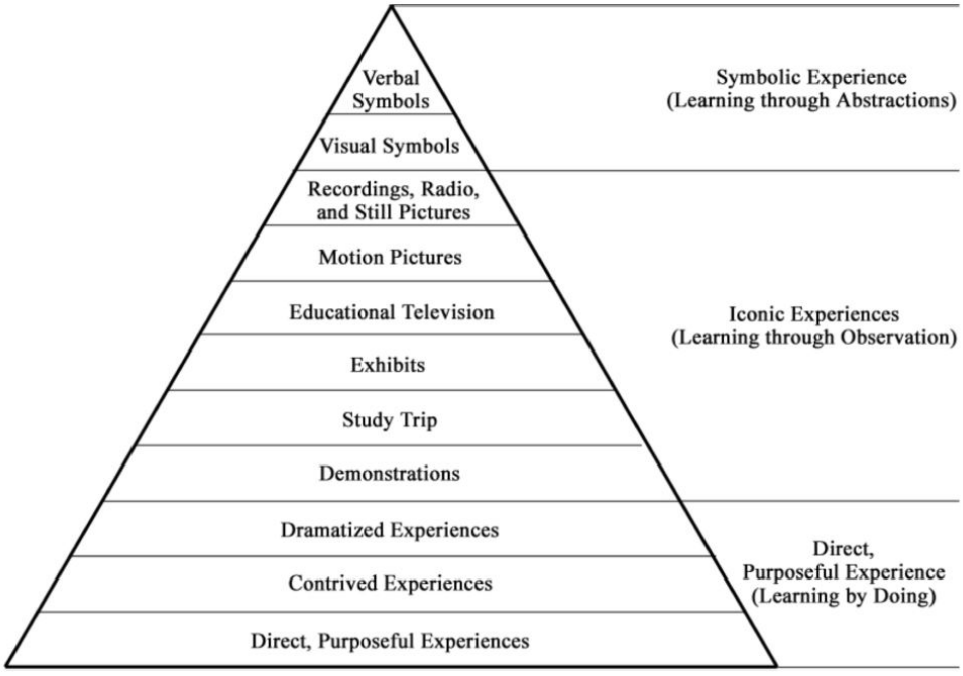

Figure 4: Cone of experience [15]

4 Building Website

As platform of the course a website was chosen because it gives a lot of possibilities in designing the course, allows for interaction in the form of quizzes and provides structure to the course by having multiple webpages. This chapter shows how the website was built and how the figures are made. The first section explains the building of the website, and the second the making of the figures. Finally, recommendations are given on how the website can be improved.

4.1 Website

The foundation of the website is made with HTML5 code. With HTML5, the website's textual and graphical content (figures) is defined, and the hierarchy between these different elements is defined. Furthermore, in the HTML5 file, the link to the CSS and JavaScript file is defined for the styling and functionality of the website.

The HTML5 code is sufficient to make a website. However, it would be a not so interesting website as it would be just a black-coloured text on a white background, like a text document. For styling a website, the CSS language is used. With CSS, a style sheet can be made where all the styling commands are given. These styling commands can change, for example, the background colour, size of a figure, font size and colour. Without CSS, every website on the internet would be a black-coloured text on a white background. So everything that makes the website not just a text file, is defined with CSS.

With HTML5 and CSS, a beautiful website can be built that shows text and figures, but further than that has no other features or functions or allows interaction. With JavaScript, the website can be made interactive by, for example, adding multiple-choice quizzes or buttons that show the answer to a question when pressed. JavaScript has been used on this website to make the exercises and quizzes. For more detail on how the JavaScript file works, look at the comments in the files.

At the start of this project, no experience was present with HTML5, CSS or JavaScript. These three programming languages were learned by searching the internet. The internet is full of sources that explain HTML5, CSS and JavaScript perfectly well. The main source consulted during this project to learn and solve problems was [23]. The code files of the website can be found in A and the instructions for and structure of the files are provided in B.

For coding, Visual Studio Code was used from Microsoft [16]. This text editor makes coding for various languages, including HTML5, CSS and JavaScript, easier and more structured. In addition, it is advised to install the code spell checker extension [5], which is a spell checker and prevents stupid spelling mistakes.

For sharing the website with participants, the website files can be compressed to a Zip file together with the CIF files and sent. The participants can then open the website in a browser. Because the files are on the computer, the participants do not have to be connected to the internet. If the website is put online, the participants can just search the internet, but then hosting costs should be paid.

So to summarise, HTML5 was used for the content and structure of the website, CSS for the styling and JavaScript for functionality and interaction of the website, for which plenty of educational material can be found on the internet. The text editor used was Visual Studio Code with the code spell checker installed. Moreover, the website can be shared both offline and online.

4.2 Figures

While making the course, many figures were made of automata and CIF code. It is important for a professional look of the website that these figures are sharp and that the automata and CIF code snippets in these figures have the same size and style. If, for example, the location of an automaton in Figure A is twice as big as that of an automaton in Figure B, this does not look professional. The same applies to the code snippets. The font size of the code in all figures should be the same. This section explains the process of making the figures in the website, the problems encountered and how they were fixed.

The automata figures were made with the Tikz automata package in Latex to ensure the figures look the same [22]. This package allows for easy drawings of automata. The details on how this package works can be found in the documentation [22]. In a subsequent project the same Latex file A must be used as was used

in this project because some parameters have been set to make the figures so far. For example, the distance between locations, the location size, the font size, arrow size and marked location style are set in that Latex file. The figures will not be the same if this file is not used. Instructions on how to use the Latex files are provided in C.

For the CIF code snippets, the Lstlisting package has been used. With this package, code can nicely be displayed. Because this package does not recognise CIF, the language was manually defined in Latex. Doing this is well described and documented in the Latex file by means of comments. A piece of code can be copy-pasted into the Latex file, which is nicely displayed to make a figure of it.

Now that it is clear how the figure's content can be made systematically of the same quality, it should be figured out how the figure itself should be made. The automata drawing or code snippet is nicely displayed on A4 size but can only be exported as a PDF file when using a standard Latex file. Instead, it should be exported as a PNG or JPG file to be usable on the website.

This can be solved by adding a new file to a project named "latexmkrc" with no extension [10]. In this file, the following line of code is added.

```
END { system('convert -density 700 -resize 15% output.pdf myImage.png'); }
```

When the file containing the automata drawing or code snippet is recompiled, the file can be exported as a PNG file in the tab "logs and output files" with the same name specified in the line of code (myImage.png).

Changing the size of the figures must be done in the line of code by changing the "-resize" percentage number. Because when the size of the figures is changed in the website, html5, the figures become blurry. Again this is well documented in the Latex file.

The above enables the figures to be exported as PNG, but the figures are still the size of an A4. Using the document class "standalone", the output file only contains the document's content with no extra (or specified) white margins [10].

So a way of making the figures is found by making a Latex file where the automata can be drawn with the Tikz automata package, and the code snippets can be displayed with the Lstlistings package. The "latexmkrc" file makes sure the figure is exportable as a PNG file, and the standalone document class makes sure that the document wraps around the content. This results in a Latex file A that consistently can make figures of the same size and quality that can directly be imported into the website.

4.3 Recommendations website

In this section, some recommendations are given to improve the website.

The course participants have to open the website page and the Eclipse interface separately on their screen. In a subsequent project, it would be beneficial for the user experience to integrate this into the website. Then, the participants can code on the website and do not have to leave the website. The result would be something similar to Jupyter notebook, where users can code and run their code on the same page where they read textual material [18].

In the third chapter, it was mentioned that adult learning principles 5,7 and 10 are not applied in the course because the course does not allow for interaction between fellow participants. This could be improved by adding a sort of discussion forum to the website where questions can be posted and answered by participants.

5 Conclusion and Recommendation

5.1 Conclusion

This project aimed to propose a one-week course in which an engineer familiar with water locks learns how to use ESCET™ for supervisory controller development. The proposed course setup is backed by literature and covers all aspects of supervisor development. The first 4 out of 10 planned modules have been developed in this project. For the other modules, a recommendation is given about its contents for a subsequent project. Modules 1-4 explain how supervisors are synthesised by making a model of the discrete-event model of the plant and a model of the requirements. The participants can practise their learned skills on the water lock case specially created for this course. The course can easily be changed to another application by replacing the water lock case with a case about another application (e.g. tunnel).

As platform for the course, a website was chosen because a website gives much freedom in designing the course. It allows for interaction in the form of quizzes and enables structure by having multiple web pages. The website was built with HTML5, CSS and JavaScript. The figures of the website have been made in Latex.

5.2 Recommendation

In this section, all recommendations are summarised.

Course content

- Module 5 and 6 could explain how hybrid models are made and how the supervisor is connected to such models.
- Module 7 and 8 could be about making digital twins with the configurator from Erick Hoogstrate if the configurator is finished by that time.
- Module 9 could explain how PLC code is generated and how this is connected to a digital twin.

Implementation

- It is recommended to add a ship to the SVG visualization and include feedback when a dangerous situation is encountered in the visualization. This will enhance the participants' learning experience and ease the supervisor's testing. Moreover, will it make the case represent reality more closely.
- A possible option is to give the hybrid model at the start of the course, when it is finished, instead of the discrete-event model, to give the participants a more real experience. However, this has pros and cons, as discussed in Subsection 2.3.1
- To allow for interaction between fellow participants, it is an option to add a discussion forum on the website, where participants can ask and answer questions.
- To make the website more user-friendly, the Eclipse interface could be integrated into the website such that the participants do not have to leave the website anymore.

References

[1] Jos C. Baeten, Joanna M. van de Mortel-Fronczak, and Jacobus E. Rooda. "Integration of supervisory control synthesis in model-based systems engineering". In: *Complex Systems* (2016), pp. 39–58. DOI: 10.1007/978-3-319-28860-4_2.

[2] Harold W. Beder and Gordon G. Darkenwald. "Differences Between Teaching Adults and Pre-Adults: Some Propositions and Findings". In: *Adult Education* 32 (3 Mar. 1982), pp. 142–155. ISSN: 0001-8481. DOI: 10.1177/074171368203200303.

[3] D van Beek. *4TC00 model-based systems engineering*. 2022. URL: https://cstweb.wtb.tue.nl/4tc00/index.html.

[4] Christos G Cassandras and Stéphane Lafortune. *Discrete Event Systems Second Edition*.

[5] *Code spell checker - visual studio marketplace*. 2022. URL: https://marketplace.visualstudio.com/items?itemName=streetsidesoftware.code-spell-checker.

[6] Jannette Collins. "Education Techniques for Lifelong Learning". In: *RadioGraphics* 24 (5 Sept. 2004), pp. 1483–1489. ISSN: 0271-5333. DOI: 10.1148/rg.245045020.

[7] *Eclipse ESCET™*. 2022. URL: https://www.eclipse.org/escet/.

[8] *Hardware-in-the-loop Set-up for Supervisory Controllers with an Application: the Prinses Marijke Complex*. 2019. ISBN: 9781728127675. DOI: 10.0/Linux-x86_64.

[9] Erick Hoogstrate. *HIL simulation with waterway locks digital twins*. 2022, Eindhoven University of Technology.

[10] *How to convert a latex PDF to PNG-*. Apr. 2021. URL: https://latexdraw.com/how-to-convert-a-latex-pdf-to-png/.

[11] N. Jazdi. "Cyber physical systems in the context of Industry 4.0". In: *Proceedings of 2014 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2014* (2014). DOI: 10.1109/AQTR.2014.6857843.

[12] Claudius Jordan, Canlong Ma, and Julien Provost. "An educational toolbox on supervisory control theory using MATLAB Simulink stateflow: From theory to practice in one week". In: *2017 IEEE Global Engineering Education Conference (EDUCON)* (2017). DOI: 10.1109/educon.2017.7942912.

[13] Malcolm S Knowles. *The modern practice of adult education*. URL: https://pdfs.semanticscholar.org/8948/296248bbf58415cbd21b36a3e4b37b9c08b1.pdf.

[14] O. T. Laseinde et al. "Educating tomorrows engineers: Reinforcing engineering concepts through Virtual Reality (VR) teaching aid". In: vol. 2016-January. IEEE Computer Society, Jan. 2016, pp. 1485–1489. ISBN: 9781467380669. DOI: 10.1109/IEEM.2015.7385894.

[15] S. J. Lee and T Reeves. *Edgar Dale and the Cone of Experience*. 2018. URL: https://edtechbooks.org/lidtfoundations..

[16] Microsoft. *Visual studio code - code editing. redefined*. Nov. 2021. URL: https://code.visualstudio.com/.

[17] F. F. H. Reijnen J. J. Verbakel J. M. van de Mortel-Fronczak and J. E. Rooda. *Models and implementation code for the Prinses Marijke complex*. 2019. URL: www.github.com/ffhreijnen/PrinsesMarijkeComplex..

[18] *Project jupyter*. 2022. URL: https://jupyter.org/.

[19] F F H Reijnen et al. *Supervisory Control Synthesis for a Waterway Lock*. 2017. ISBN: 9781509021826. DOI: 10.0/Linux-x86_64.

[20] Ma Reniers and JM van de Mortel-Fronczak. *4CM30, Supervisory control*. 2022, Eindhoven University of Technology.

[21] R. J.M. Theunissen M. Petreczky R. R.H. Schiffelers D. A. Van Beek J. E. Rooda. "Application of supervisory control synthesis to a patient support table of a magnetic resonance imaging scanner". In: *IEEE Transactions on Automation Science and Engineering* 11 (1 Jan. 2014), pp. 20–32. ISSN: 15455955. DOI: 10.1109/TASE.2013.2279692.

[22] Satyaki Sikdar. *Drawing Finite State Machines in L A T E X using tikz A Tutorial*. 2017. URL: https://www3.nd.edu/~kogge/courses/cse30151-fa17/Public/other/tikz_tutorial.pdf.

[23] *W3Schools free online web tutorials*. 2022. URL: https://www.w3schools.com/.

[24] Ministerie van Infrastructuur en Waterstaat. *Our organisation*. June 2022. URL: https://www.rijkswaterstaat.nl/en/about-us/our-organisation.

Appendices

A Project files

The project files, including the website, Latex, and CIF files, can be found here.

B Website files instructions and contents

The instructions for and contents of the website files, found in the "Website_final.zip" A file, are explained in this appendix.

B.1 Open the website

1. Extract the "Website_final.zip" file.
2. Open the index.html (or another .html file) in a browser (Internet Explorer is not supported).

B.2 Change the content and style of the website

. The content of the websites is defined in the HTML5 files (.html). The styling of the website is done in the CSS files (.css).

- To change the content of moduleX open the "moduleX.html" file, with X the module number, e.g. 2.
- To change the styling of moduleX open the "moduleX.css" file.
- To change the styling of all modules, open the "generalmodule.css" file, which is imported in all the "moduleX.css" files. A change in the "generalmodule.css" is displayed in all modules.
- To change the styling of the navigation bar, open the "navbar.css" file

B.3 Create an exercise or assignment

All the interactive parts of the website, like the quizzes and exercises, use functions defined in JavaScript files (.js). Each "moduleX.html" file imports its own "mainX.js", with X the module number, e.g. 2, where the functions used are located. In this subsection, the creation of an exercise or assignment is explained. Exercises and assignments have the same structure.

1. The exercises and assignments have a results button that displays the answer when clicked. To achieve this, the "hide()" function is created in the JavaScript file, displayed in Figure 5. Each element in an HTML5 file can be labelled with an id. The hide function takes an id as input and changes its display attribute value to "block" when it is "none" and vice versa.

```
function hide(id) {

    var x = document.getElementById(id);
      if (x.style.display == 'none') {
      x.style.display = 'block';}
      else {
        x.style.display = "none";
      }}
```

Figure 5: Hide function

2. In the "moduleX.html" file, the button has to be created and the elements that contain the exercise and the answer to the exercise. Figure 6 shows how this is done. First, a div element is created with an id name. This id name is given as input to the hide function, which is executed when the button

is clicked. Each element can also belong to a class. A class can contain multiple elements where an id is specific to only one element. The classes "submit" and "answer" are used for styling. In the "generalmodule.css" file, the style of these classes is defined. Every element that belongs to such a class takes the same style.

```
<h2>Exercise (number)</h2>
<p>
    Type here the exercise.
</p>

<button class="submit"  onclick="hide('ansexercise')">Get Results</button>

<div class="answer" id="ansexercise">
    Type her the answer to the exercise.
</div>
```

Figure 6: Template of an exercise or assignment.

3. When the website is reloaded, the button works as intended. However, the answer is by default displayed, which is not desired for an answer. To change this, the line of code in Figure 7 is added to the JavaScript file, which changes the display attribute value to "none".

```
document.getElementById("ansexercise").style.display = "none"
```

Figure 7: Change display to none

.

B.4 Create a quiz

Below, the steps to create a quiz are provided.

1. To start, the quiz content has to be defined. This is done by defining a constant in the form of Figure 8 in the "mainX.js" file, where X is the module number where the quiz is supposed to be displayed.

```
const Questions = [
  {
    question: "Question 1.",
    answers: {

      a: "Answer 1.",
      b: "Answer 2."
    },
    correctAnswer: "a"
  }]
```

Figure 8: Quiz questions template

2. After the questions have been defined, the quiz can be built by calling the "quizbuild(Questions, "quiznumber", "id", "type")" function, as shown in Figure 9. The first parameter, "Questions", is a constant that contains the quiz questions, as shown in Figure 8. The second parameter is the quiznumber which is explained in the next step. The quizzes can be of three types specified by the fourth parameter, which can have the following values: "blur", "hide", and "off". A quiz of type blur unblurs the element with id name "id", the third parameter, that is part of the subjects class when the participant answered all the questions correctly. The subjects class should be set to a blur of 6px in the CSS files when this types is used. In the final version of the website, this type does not occur anymore. A quiz of type "hide" unhides the element with id name "id", the third parameter, if the participant answered all the answers correctly. In Modules 1 and 2, this is used. A quiz of type "off" does not unhide or unblur anything; it is just a simple quiz.

```
quizbuild(Questions, "quiznumber", "id", "type")
```

Figure 9: Quizbuild() function instantiation

3. In the "moduleX.html" file, the quiz should be defined at a place on the web page. This is done as shown in Figure 10. Here the second parameter of the quizbuild function comes in, "quiznumber". The quiz is built by replacing "quiznumber" in both the div elements id's and in the quizbuild instantiation with a number, e.g. 1.

```
<div id="quiz{quiznumber}"></div>
<button class="submit" id="submit{quiznumber}">Get Results</button>
<div id="results{quiznumber}"></div>
```

Figure 10: Defining quiz in HTML5 file

B.5 Other files

The list below sums up all the other files found in the "Website_final.zip" A file and their contents.

- The "Casefiles.zip" file contains all the files the participants need to open the uncontrolled plant simulation. It contains the uncontrolled plant, "Plant_provided.cif", the SVG drawing of the Westsluis, "lock.svg", a CIF file containing all in- and output mappings for the SVG visualization, "SVG.cif", and a tooldef file to start the simulation, "simulation.tooldef". The participants can download the "Casefiles.zip" in Subsection "Setup", Module 1.
- The "synthesis.zip" file contains the "synthesis.tooldef" file to synthesize a supervisor. The participants can download this file in Section "Water lock case", Module 4.
- The "Allcasefiles.zip" file contains all the files of the water lock case, which is more elaborated in D. The participants can download this zip file at the end of Module 4.
- The "controllability.zip" file contains the answers to Exercise 1, Module 4.
- The "Spartan.zip" file contains the font style used in the navigation bar.
- The "img" contains all the figures used in the website.

C Latex files instruction

This appendix explains the steps to make the figures in Latex with the files found in the "Latexfiles_final.zip" file A.

C.1 Make figure

- Import one of the figuremaker folders into a Latex editor.
- Open the "main.tex" file where all the figures' definitions can be found. All figure definitions are commented at the top with the same name as the actual figure in the "img" folder in the "Website_final.zip" file.
- To create a figure, uncomment the figure definition and re-compile the latex file.
- In the "Logs and output files", the compiled document can be downloaded as a PNG file.

C.2 Change figure settings

- Under "lstdefinelanguage" in the "main.tex" file, all the parameters are specified to display the CIF code snippets in Latex. Here additional keywords that should be displayed in a certain colour can be defined.
- Under "tikzset" in the "main.tex", the parameters for the Tikz figures are defined, like node size and distance.
- To change the size, dots per inch density or output file name, change the "density", "resize" attribute value and the last entry, respectively, in the line of code in the "latexmkrc" file, shown below.

```
END { system('convert -density 700 -resize 15% output.pdf myImage.png'); }
```

D CIF files contents

This section explains the contents of the "CIFfiles_final.zip" file A.

For each module, a folder contains a content, an exercises and a case folder containing the CIF specifications of the examples used in the theory and ESCET™ sections, the exercises and the case assignments, respectively.

The Marijke sluis folder is retrieved from [19].

Below, a list is provided with the contents of the water lock case folder. These files are also part of the "Allcasefiles.zip" file, found in the "Website_final.zip" file A.

- The "lock.svg" file contains the SVG drawing of the Westsluis used in the water lock case. This file is provided to the participants at the start of the course and is part of the "Casefiles.zip" file.
- The "Plant_templates.cif" file contains all the templates defined by automaton and group definitions for the components and subcomponents of the Westsluis.
- The "Plant.cif" file models the plant and contains all the group instantiations of the group definition defined in the "Plant_templates.cif" file and an automaton to model the commands.
- The "Plant_requirements_.cif" file only contains the plant requirements that make the commands functional. The "Plant.cif", "Plant_templates.cif", and "Plant_requirements_.cif" model the uncontrolled plant.
- The "Plant_provided.cif" file is provided to the participants at the start of the course and is part of the "Casefiles.zip" file. This file is the output file of data-based synthesis with the "Plant.cif", "Plant_templates.cif", and "Plant_requirements_.cif" as input. This makes the participants cannot easily copy-paste the plant model and cheat during the cours.
- The "Requirements.cif" file contains all the requirements for the Westsluis, including the safety requirements that ensure the lock does not contain dangerous behaviour. The "Plant.cif", "Plant_templates.cif", and "Requirements_.cif" model the controlled plant.
- The "Supervisor.cif" file is the output file of data-based synthesis with the "Plant.cif", "Plant_templates.cif", and "Requirements_.cif" as input.
- The "synthesis_Plant_provided.tooldef" and "synthesis_Supervisor.tooldef" file synthesize the "Plant_provided.cif" and "Supervisor.cif" files, respectively, with the data-based synthesis tool, when executed.
- The "SVG.cif" file contains all the in- and output mappings for the SVG visualization. This file is provided to the participants at the start of the course and is part of the "Casefiles.zip" file.
- The "simulation.tooldef" file simulates the "SVG.cif" file when executed. This file is provided to the participants at the start of the course and is part of the "Casefiles.zip" file.