Development a Safety Critical Submarine Control System Using Ada and proved by SPARK

Dylan Tyrie-Dron

SET10112

1 Introduction

A submarine system was developed using Ada and was proven by SPARK. The submarine system was implemented based on the following rules:

- The submarine must have at least one airlock door closed at all times.
- The submarine can perform no operations unless both doors are closed and locked.
- If the oxygen runs out, the submarine has to surface.
- If the reactor overheats, the submarine has to surface.
- If the oxygen runs low, a warning must be shown.
- The submarine cannot dive beneath a certain depth.
- The submarine must be capable of storing, loading and firing torpedoes safely.

The system must carry out these requirements to a certain proven standard of SPARK. This system will aim to meet the platinum standard with all requirements covered.

2 Controller Structure

The submarine consisted of one air-locked door, an oxygen level indicator, a depth level indicator, reactor level indicator and a torpedo chamber. The separate components were put into different files to categorise the code.

The submarine file was the key file that brought all the other files together. It also contained code for the submarine's movement.

A file was created for the control of the doors of the submarine. This set up how the doors would behave and was included in the submarine file to be used to prevent any action occurring when both doors were not closed.

Another file was created for the reactor which would control its temperature. It was included in the submarine file to be later called when the submarine increased in depth.

A file was created for the torpedo. It controlled the firing of torpedoes by setting up a chamber with two hatches. The process of firing a torpedo involved the opening of the outer hatch the firing of the torpedo and then the closing of the outer hatch. The process of loading the torpedo involved opening the inner hatch and then storing the torpedo.

A file was created for the control of the oxygen level. The oxygen level was decreased as the submarine increased in depth.

3 Desciptions of Procedures and Functions

The controller consisted of many small parts. The first thing to do was to make a *submarineType* record. The record would be used to store an instance of a submarine and allow, for example, its depth to be changed. The *submarineType* contained an array of type "Door" which contained 2 doors that formed an air-lock. The oxygen level was updated using an integer with initial value of 8. An integer was also set up to update the *depthLevel* and the *reactorLevel* both with initial values set to 0.

3.1 First Requirement

The first requirement was to make sure that at least 1 door was closed at all times. This was done by creating a Door record within a Doors package that had a *doorStatus* variable attached to it to set the door to either closed or open. This door also had a lock attached to it, the lock had a *lockStatus* to set the door to be locked or unlocked. An array of Doors was set up with a range of 2 so that it could be used in the submarine package to check if doors were closed before running other functions.

A procedure was created to control the locking of doors: *closeDoors*, the procedure closes both doors of the submarine. The procedure, took as input, an array of doors. Every door of the array was closed and locked in the procedure's body. This was then made certain by checking that the status of every door and lock in the array was closed and locked using a postcondition. An invariant was set up to use in preconditions for other functions in the submarine package that checked if both doors in the array were closed and locked. The invariant was placed in the submarine package so that it could be used by the functions of the submarine to check that the doors are locked before starting each function.

3.2 Second Requirement

The second requirement was to ensure that the submarine could not perform any operations unless 2 doors were locked and closed (airlocked). The second requirement was handled using the same solution as the first requirement (using the *closeDoors* procedure and invariant).

3.3 Third Requirement

For the third requirement, the submarine first had to have the ability to change depths. Secondly, it had to be ensured that a warning had to shown if the oxygen levels of the submarine went below a certain threshold (3). The submarine also has to re-surface after the oxygen level is 0. In the submarine package, the movement was controlled. Three procedures were set up to handle the movement:

– *increaseDepth* - with parameters: depth level, oxygen level and an array of doors.
– *resurface* - with parameters: depth level, oxygen level, an array of doors and reactor temperature.
– *decreaseDepth* - with parameters: depth level, oxygen level and an array of doors.

The *increaseDepth* procedure contained a precondition that consisted of the oxygen level being higher than 0 and lower than or equal to 8. It also made sure that the depth was higher than or equal to 0 and lower than 6 and that both doors were closed. Through a postcondition it was also ensured that the depth would increase by 1 and that the depth would not go any higher than 6.

The *resurface* procedure contained a precondition that consisted of the oxygen level either equalling 0 or was higher than 0 and the reactor temperature being 12. The depth had to be higher than 0 and both doors were closed. Through a postcondition which contained the condition that the depth level should be 0, a while loop ensured that the submarine would go straight back up to the surface.

The *decreaseDepth* procedure contained a precondition that consisted of the oxygen level being higher than or equal to 0 and the depth level was higher than the first instance of the depth level and that both doors of the submarine were closed. A postcondition consisting of the depth level being 1 less than the old depth level ensured that the submarine decreased in depth.

The submarine was also set up to manage oxygen levels. The oxygen level was decreased using a procedure called *decrementOxygen*. It takes in an oxygen level and decrements it. If the oxygen is low enough (below 3), it sends a warning to the submarine. A precondition was set up to check if the oxygen level was lower than or equal to 8. A postcondition ensured that the oxygen level would decrease which consisted of the oxygen level being 1 less than the old oxygen level and the oxygen level was higher than or equal to 0.

3.4 Fourth Requirement

The fourth requirement is that the submarine must surface if the nuclear reactor overheats. Another package was created to keep the reactor code separate. A procedure was set up to increase the heat of the reactor by incrementing the inputted variable *reactorLevel*. A precondition was set up to check that the reactor temperature was less than 12. The postcondition contained the reactor temperature being 1 higher than the old reactor temperature and that the reactor temperature was less than or equal to 12.

3.5 Fifth Requirement

Another requirement is that the submarine cannot dive beneath a certain depth. This was handled from within the *increaseDepth* procedure in the submarine package.

3.6 Sixth Requirement

The last requirement was to make sure the submarine could load, store and fire torpedoes safely. Another package to separate the torpedo code from the main submarine code was created. In the torpedo package the following was set up:

- *torpedoType* that has a *torpedoStatus*.
- *torpedoChamber* that contains a torpedo and 2 hatches.
- *torpedoes* - an array of torpedoes with size of 2.
- *hatch* that has a *hatchStatus*.
- *hatches* - an array of hatches with size of 2.
- *torpedoStatus* which has 3 states fired, stored, loaded.
- *hatchStatus* which is either open or closed.

The torpedo type record contained an attribute *torpedoStatus* which set the states of the torpedo to either fired, stored or loaded. The hatch record contained a *hatchStatus* which set the hatch to either open or closed. The torpedoes and hatches both had an array set to them with ranges of 2. A torpedo chamber was a record that contained 1 torpedo and 2 hatches.

A invariant was created to make sure that both hatches were closed before firing the torpedo.

There were 3 procedures that controlled the torpedo which were *fireTorpedo*, *storeTorpedo* and *loadTorpedo*. The *fireTorpedo* procedure took in a torpedo chamber as a parameter and before running the procedure a precondition consisted of the invariant checking the hatches were closed was set up. A postcondition ensured that the second hatch in the hatches array in the torpedo chamber was closed and the torpedo in the torpedo chamber had been fired.

The second procedure *storeTorpedo* took in a torpedo chamber as a parameter. It also had a precondition, the precondition consisted of the second hatch being open and the first hatch being closed. The postcondition ensured that the procedure closed the first hatch in the hatches array and the second hatch was closed and the torpedo was set to stored.

The *loadTorpedo* procedure also took in a torpedo chamber as a parameter. The precondition that was set was that the second hatch had to be closed and the first hatch had to be open. The postcondition made sure that the hatches were both closed and the torpedo was loaded.

3.7 Extensions to the Requirements

Two extensions were made to this project object orientation and a simulation. Object orientation was done by separating the code into packages and record types were made to be reused by different parts of the code. The second extension was a simulation of the submarine. This was done by running a method in the submarine package called: *run* which took in no parameters but took in a global instance of the submarine. The *run* method ran all the procedures increasing the depth to its maximum and draining the oxygen so that the submarine was forced to resurface. It then increased its depth again so that it resurfaced due the overheating of the reactor. The *run* method was run in the main package, the simulation printed its current depth, oxygen level and reactor temperature at the start of the simulation and then at the end of the simulation to prove that the procedures and functions of the submarine worked as intended.

4 Proof of Consistency

Proof of consistency was used to test and prove that the code used to make sure the requirements were met.

A key procedure is the *decreaseDepth* procedure. The precondition is the depth level (dl) is higher than the first instance of the depth level (dl'First) and the oxygen level (ox) is higher than or equal to 0. The doors are also to be closed. The postcondition is that the depth level is one lower than the old instance of the depth level (dl'Old). The procedure's body is such the depth level is assigned to be 1 less than the current depth level.

Therefore the premises are:

- dl > dl'First
- ox $\geq$ 0
- doors are closed
- dl = dl'Old-1

The conclusions are:

- dl-1 > dl'First
- ox $\geq$ 0
- doors are closed
- dl-1 < dl'Old-1

As the doors are closed and the oxygen level stays the same in the premises and conclusions they are omitted.

The proof is:

$$(dl > dl'First, dl = dl'Old - 1) \Rightarrow (dl - 1 > dl'First, dl - 1 > dl'Old - 1)$$

$$A_X$$

$$(dl > dl'First, dl = dl'Old - 1) \Rightarrow (dl - 1 > dl'First \vee dl - 1 > dl'Old - 1)$$

$$R_\vee$$

$$(dl > dl'First \wedge dl = dl'Old - 1) \Rightarrow (dl - 1 > dl'First \vee dl - 1 > dl'Old - 1)$$

$$L_\wedge$$

$$\Rightarrow (dl > dl'First \wedge dl = dl'Old - 1) \supset (dl - 1 > dl'First \vee dl - 1 > dl'Old - 1)$$

Another key procedure is the *incrementTemperature* procedure in the *incrementTemperature* procedure in the reactor package. A precondition was set up to check that the reactor temperature (rt) was less than 12. The postcondition contained the reactor temperature being 1 higher than the old reactor temperature (rt'Old) and that the reactor temperature was less than or equal to 12. The procedure's body checks that the reactor temperature is less than 12 and then assigns the temperature to be one more than the current reactor temperature.

Therefore the premises are:

- rt < 12
- rt = rt'Old+1

The conclusions are:

- rt ≤ 12
- rt +1 > rt'Old +1

The proof is:

$$(rt < 12, rt = rt'Old + 1) \Rightarrow (rt \leq 12, rt + 1 > rt'Old + 1)$$

$$A_X$$

$$(rt < 12, rt = rt'Old + 1) \Rightarrow (rt \leq 12 \vee rt + 1 > rt'Old + 1)$$

$$R_\vee$$

$$(rt < 12 \wedge rt = rt'Old + 1) \Rightarrow (rt \leq 12 \vee rt + 1 > rt'Old + 1)$$

$$L_\wedge$$

$$\Rightarrow (rt < 12 \wedge rt = rt'Old + 1) \supset (rt \leq 12 \vee rt + 1 > rt'Old + 1)$$

5 Conclusion

To sum up the submarine system was fully implemented with some additional functionality. It was proved by SPARK to be of a platinum standard of code. However, the way the system outputted to a console was not very user engaging. This could be improved by printing the system output in a while loop every few seconds to give a more visual representation to the user that the system works and what the system actually does. This could be further enhanced by allowing the user to control the input of the submarine using keyboard controls within this while loop. The system could also be enhanced by improving certain postconditions. The problem with the system's torpedo system is that the amount of torpedoes are not tracked. This could be enhanced by creating an integer to count the number of torpedoes that have been fired and making sure that integer is conditioned accordingly to make sure that the count is at the right number.