

ASW in RSML^e

To: CSci 8801, All students
CC: Teaching Assistant
From: Dr. Mats Heimdahl
Date: 10/15/2006
Re: Homework Assignment 3

Introduction

The problem we will work on is adopted from the civil avionics industry. It is a very small system (called the altitude switch—ASW) that is responsible for turning on the power to a device (we call it the Device Of Interest—DOI) when the aircraft drops below a certain altitude. The ASW is used to automatically start devices such as ground radar or a ground proximity warning system.

The ASW example is adopted from Dr. Steve Miller at Rockwell Collins.

Background

A large avionics firm worked with a supplier to develop the altitude switch. In the project they developed the following [rather poor] informal requirements specification.

The Altitude Switch

The Altitude Switch (ASW) is a re-useable component that turns power on to a Device Of Interest (DOI) when the aircraft descends below a threshold altitude (nominally 2,000 feet, but this can be adjusted for various installations of the ASW) above ground level (AGL). If the altitude cannot be determined for more than two seconds, the ASW indicates a fault. The detection of a fault turns on an indicator lamp within the cockpit.

The DOI is turned back off again if the aircraft ascends above the threshold altitude plus some hysteresis value. The hysteresis value is defined to be 10% of the threshold.

The ASW receives a status indication from the DOI indicating whether the DOI is powered on. If the DOI does not indicate that it is powered on within two seconds after power is applied, a fault is indicated. The ASW does not apply power to the DOI if the DOI is already powered on.

The ASW is not in complete control of the DOI—the DOI may be turned on and off by other systems or the pilot. If the DOI is turned off after the aircraft descends below the threshold altitude, the ASW does not reapply power to the DOI unless the aircraft again descends below the threshold altitude.

The ASW also accepts an inhibit signal that prevents it from turning on power to the DOI or indicating a fault. All other ASW functions are unaffected by the inhibit signal.

The ASW also accepts a reset signal that returns it to its initial state.

Monitored and controlled variables

The ASW has the following Monitored and Controlled variables (tentative list that may change as you work on the system).

Name	Type	Description
<u>Monitored Variables</u>		
Altitude	0..40,000 feet	The true altitude above ground.
Alt. Quality	{OK, Bad}	Indication whether the altitude reported in Altitude can be trusted.
Threshold	1,000..30,000 feet	The threshold at which the DOI is turned on.
DOI Status	{On, Off}	Indication if the DOI is turned on or off.
Inhibit	{Inhibit, Not-Inhibit}	Indication if the ASW should be inhibited or not.
Reset	{Reset, Not-Reset}	Puts the ASW back into its initial state.
<u>Controlled Variables</u>		
Alarm	{On, Off}	Alarm light used when the pilot needs to be notified.
DOI Command	{On, Off}	Command that turns the DOI on or off.

The Problem

During the inspections and discussion of the English language ASW requirements, the customer got the feeling that there were several areas that needed further clarification (in fact, they were very unhappy). They were also somewhat (read very) concerned about new government regulations being imposed on such systems. Therefore, they have contracted with us to develop a formal model of the required ASW behavior. This is the problem we will tackle in this assignment.

The Language

Since the customer knows we have vast experience with the formalism provided through RSML^e and its execution environment, they want us to use the RSML^e notation. Also, they feel that the relative clarity of the RSML^e models will help communicating with their engineers and the regulatory agency in question.

The Changes

The hardware is finally becoming available in this project (for those of you that worked on this project in CSci 5801 last year). Naturally, there has been a change to the initial assumptions. The following change has been made to the environment (luckily, the change makes our task simpler):

1. The environment will assure that no two events happen at the same time. Thus, we do not need to worry about priorities. Therefore, we will model each input as a message over a separate interface of the RSML^e model.

Besides this, the requirements are the same as outlined above.

The Interfaces with the Environment

The final hardware decisions have not yet been made. For now, we know we will use three altimeters at some point—we are just unclear on exactly which manufacturer to use. We will determine this within two weeks. For now, we will use a MON and CON approach to modeling the environment—that is, we will not worry about the details of sensors and actuators. When we get a decision on the altimeters from the Altimeter Selection Committee, we will add that information to our model (**hint:** this will be in the next assignment so you may as well plan on it).

The interfaces with the environment are defined at the end of this assignment. A skeleton specification taking care of the interface definitions is also available from the web page.

A simulation that we can use to test our system is available. This simulation is compatible with the interfaces defined below and you are encouraged to use it to test your model. Instructions for using the simulation are available on the web site.

Acceptance Criteria

The customer promised to provide a small acceptance-test suite. The test suite will be made available about one week before the delivery date. It is highly desirable that the ASW model passes the acceptance tests before it is delivered. Should your ASW model not pass the acceptance tests and explanation why your solution is acceptable is required.

Interaction with the Customer

Questions regarding the behavior of the ASW shall be posted on the Forum.

Technical Support

Questions regarding RSML^e and Nimbus shall be shall be posted on the Forum.

Information and sample files will also be posted on the web pages—please check them regularly.

Your Task

Develop an RSML^e model of the ASW to clarify the rules that govern the system.

Deliverables

I want you to turn in one solution either individually or in a team of two. I expect you to make sure the specification is (1) liberally commented, (2) syntactically correct and type checked (done when parsed into Nimbus), and (3) tested (with at a minimum the acceptance tests). Hand in the following:

1. A printout of the specification.
2. A soft copy of the specification (email is preferred).
3. Should you be unable to make the model conform to the acceptance test suite, you must fill out a bug report (available on the web page) for the test case that did not pass.

Due Date

The deliverables must be presented to the customer (Professor) no later than in class on Thursday, October 26.

Interfaces to the Environment

Communication with the DOI

```
TYPE_DEF DOIStatusType {On, Off}

MESSAGE DOICommandMessage {
    command IS DOIStatusType
}

MESSAGE DOIStatusMessage {
    status IS DOIStatusType
}

IN_INTERFACE DOIStatusMessageInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    INPUT_ACTION : RECEIVE(DOIStatusMessage)

    RECEIVE_HANDLER :
        CONDITION : TRUE
        ASSIGNMENT
            DOIStatus := status
        END ASSIGNMENT
    END HANDLER

END IN_INTERFACE

OUT_INTERFACE DOICommandInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    OUTPUT_ACTION : SEND(DOICommandMessage)

    HANDLER :
        CONDITION : TBD
        ASSIGNMENT
            command := TBD
        END ASSIGNMENT
        ACTION : SEND
    END HANDLER

END OUT_INTERFACE
```

Communication with the Alarm

```
MESSAGE FaultMessage {
    fault IS BOOLEAN
}
```

```
OUT_INTERFACE FaultDetectionInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 200 MS
    OUTPUT_ACTION : SEND(FaultMessage)
```

```
    HANDLER :
        CONDITION : TBD
        ASSIGNMENT
            fault := TBD
        END ASSIGNMENT
        ACTION : SEND
    END HANDLER
```

```
END OUT_INTERFACE
```

Communication with the altitude device

```
TYPE_DEF AltitudeQualityType { Good, Bad }
```

```
MESSAGE AltitudeMessage {
    Alt IS INTEGER,
    aq IS AltitudeQualityType
}
```

```
IN_INTERFACE AltitudeMessageInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    INPUT_ACTION : RECEIVE(AltitudeMessage)
```

```
    RECEIVE_HANDLER :
        CONDITION : TRUE
        ASSIGNMENT
            TBD := Alt,
            TBD := aq
        END ASSIGNMENT
    END HANDLER
```

```
END IN_INTERFACE
```

Communication with the inhibit button

```

TYPE_DEF InhibitType { Inhibit, NoInhibit }

MESSAGE InhibitMessage {
    i IS InhibitType
}

IN_INTERFACE InhibitMessageInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    INPUT_ACTION : RECEIVE(InhibitMessage)

    RECEIVE_HANDLER :
        CONDITION : TRUE
        ASSIGNMENT
            TBD := i
        END ASSIGNMENT
    END HANDLER

END IN_INTERFACE

```

Communication with the reset button

```

MESSAGE EmptyMessage {x IS INTEGER}
/* The current environment simulation requires a message field */
/* even if one is not needed. X will never be used. */

IN_INTERFACE ResetMessageInterface :
    MIN_SEP : 50 MS
    MAX_SEP : 100 MS
    INPUT_ACTION : RECEIVE(EmptyMessage)

    RECEIVE_HANDLER :
        CONDITION : TRUE
        ASSIGNMENT
            ivReset := TRUE
        END ASSIGNMENT
    END HANDLER

    HANDLER :
        CONDITION : TRUE
        ASSIGNMENT
            ivReset := FALSE
        END ASSIGNMENT
    END HANDLER

END IN_INTERFACE
/* The variable ivReset is used to capture if a reset message */
/* was received. The interface will set ivReset to TRUE when */
/* a message arrives and to false immediately in the next */
/* instance of time. */

```