Floating Weather Station Module Guide*[1]

This guide to the modules of the Floating Weather Station Family is patterned after the A-7E Module Guide [6] and uses the structuring principles describe therein and in [41]. Accordingly, we quote, with suitable emendations, including the substitution of FWS for A-7, a section of the A-7 module guide to describe further the purpose of the FWS module guide.

> The FWS module guide provides an orientation for software engineers who are new to the FWS family, explains the principles used to design the structure, and shows how responsibilities are allocated among the major modules.
>
> This guide is intended to lead a reader to the module that deals with a particular aspect of the system. It states the criteria used to assign a particular responsibility to a module and arranges the modules in such a way that a reader can find the information relevant to his purpose without searching through unrelated documentation.
>
> This guide describes and prescribes the module structure. Changes in the structure will be promulgated as changes to this document. Changes are not official until they appear in that form. This guide is a rationalization of the structure, not a description of the design process that led to it.
>
> Each module consists of a group of closely related programs. The module structure is the decomposition of the program into modules and the assumptions that the team responsible for each module is allowed to make about the other modules.

Goals of the Module Structure

The overall goal of the decomposition into modules is reduction of software cost by allowing modules to be designed, implemented, and revised independently. Specific goals of the module decomposition are:

(1) each module's structure should be simple enough that it can be understood fully;

(2) it should be possible to change the implementation of one module without knowledge of the implementation of other modules and without affecting the behavior of other modules;

[1] The FWS Module Guide is taken, with permission, from Engineering Domains: A Family Based Software Development Process, David M. Weiss and Chi Tau Robert Lai, Addison Wesley, in publication.

(3) the ease of making a change in the design should bear a reasonable relationship to the likelihood of the change being needed; it should be possible to make likely changes without changing any module interfaces; less likely changes may involve interface changes, but only for modules that are small and not widely used. Only very unlikely changes should require changes in the interfaces of widely used modules. There should be few widely used interfaces;

(4) it should be possible to make a major software change as a set of independent changes to individual modules, i.e., except for interface changes, programmers changing the individual modules should not need to communicate. If the interfaces of the modules are not revised, it should be possible to run and test any combination of old and new module versions.

As a consequence of the goals above, the FWS software is composed of many small modules. They have been organized into a tree-structured hierarchy; each non-terminal node in the tree represents a module that is composed of the modules represented by its descendents. The hierarchy is intended to achieve the following additional goals:

(5) A software engineer should be able to understand the responsibility of a module without understanding the module's internal design.

(6) A reader with a well-defined concern should easily be able to identify the relevant modules without studying irrelevant modules. This implies that the reader be able to distinguish relevant modules from irrelevant modules without looking at their internal structure.

Design Principle

The FWS module structure is based on the decomposition criteria known as information hiding [35]. According to this principle, system details that are likely to change independently should be the secrets of separate modules; the only assumptions that should appear in the interfaces between modules are those that are considered unlikely to change. Every data structure is private to one module; it may be directly accessed by one or more programs within the module but not by programs outside the module. Any other program that requires information stored in a module's data structures must obtain it by calling module programs.

Applying this principle is not always easy. It is an attempt to minimize the expected cost of software and requires that the designer estimate the likelihood of changes. Many of the changes that are accommodated by the module structure described in this document are guided by the variabilities described in the Floating Weather Station Commonality Analysis.

Module Description

This document describes the module structure by characterizing each module's secrets. Where useful, we also include a brief description of the services provided by the module. Where a module's secret is directly concerned with a variability from the commonality analysis, we also identify the variability.

The remainder of this document consists of two parts:

- a top-down overview of the module structure, and
- a graphical depiction of the module structure.

1. Behavior Hiding Modules

The behavior hiding modules include programs that need to be changed if the required outputs from a FWS and the conditions under which they are produced are changed. Its secret is when (under what conditions) to produce which outputs. Programs in the behavior hiding module use programs in the Device Interface module to produce outputs and to read inputs.

1.1. Controller

Service

Provide the main program that initializes a FWS.

Secret

How to use services provided by other modules to start and maintain the proper operation of a FWS.

Associated Changes

Period at which the wind temperature sensors are read

Period at which the water temperature sensors are read

[While we anticipate that these may change independently, these issues can be easily separated in the module without adding complexity.]

1.2. Message Generation

Service

Periodically retrieve weather data from the Data Banker and transmit it.

Secret

How to use services provided by other modules to obtain weather data and transmit it at a fixed period.

Associated Changes

[9] Transmission Period

1.3. Message Format

Service

Support construction of an output message.

Secret

How to create a message in the correct format for transmission.

Associated Changes

Message format

[3] Number and kinds of weather data [Assume the format will change when the number and kinds of data changes.]

2. Device Interface Modules

The device interface modules consist of those programs that need to be changed if the hardware devices to FWSs or the output to hardware devices from FWSs change. The input from secret of the device interface modules is the interfaces between FWSs and the devices that produce its inputs and that use its output.

2.1. Wind Temperature Device Driver

Service

Provide access to the wind temperature sensors. Provide wind temperature values and an indicator if a sensor has failed. There may be a submodule for each sensor type.

Secret

How to communicate with, e.g., read values from, the sensor hardware.

How to determine if a wind temperature sensor has failed [We assume that this will change when the device changes]

Associated Changes

[2.1] Wind speed sensor hardware.

2.2. Water Temperature Device Driver

Service

Provide access to the water temperature sensors. Provide water temperature values and an indicator if a sensor has failed. There may be a submodule for each sensor type.

Secret

How to communicate with, e.g., read values from, the water temp sensor hardware.

How to determine if a water temperature sensor has failed. [We assume that this will change when the device changes]

Associated changes

[2.2] Water temperature hardware

2.3. Transmitter Device Driver

Service

Transmit weather data upon request.

Secret

The details of the transmitter hardware.

Associated variabilities and parameters of variation

[4] Transmitter hardware.

Note

This module hides the boundary between the FWS domain and the radio transmission domain. The boundary is formed by an abstract interface that is a standard for all radio transmitters. Programs in this module use the abstract interface to send messages to the transmitter to be broadcast.

3. Software Design Hiding Modules

The software design hiding modules hide software design decisions based upon programming considerations such as algorithmic efficiency. Both the secrets and the interfaces to this module are determined by software designers. Changes in these modules are more likely to be motivated by a desire to improve performance than by externally imposed changes.

3.1. Wind Temperature Sensor Monitor

Service

Retrieve data from the wind temperature sensor(s) and deposit valid data it in the Data Banker.

Secret

How to use the services provided by other modules to obtain wind temperature data at and store it for later retrieval.
How data from failed sensors is handled.

Associated Changes

[2?] Number of wind temperature sensors

3.2. Water Temperature Sensor Monitor

Service

Retrieve data from the water temperature sensor(s) and deposit it in the Data Banker.

Secret

How to use the services provided by other modules to obtain water temperature data and store it for later retrieval.

How data from failed water temperature sensors is handled.

Associated Changes

[2?] Number of water temperature sensors

3.3. Data Banker

Service

Store wind temperature and water temperature data.

Secret

The algorithm and data structure used to store and retrieve data

Associated Changes

None

3.3. Determine Wind Temperature

Service

Process the current Data Banker data to produce a current wind temperature estimate. Deposit in the Data Banker.

Secret

The algorithm used.

Associated Changes

[7, 8] Algorithm for accurizing wind temperature from a history of readings.

3.3. Determine Water Temperature

Service

Process the current Data Banker data to produce a current water temperature estimate. Deposit in the Data Banker.

Secret

The algorithm used to estimate the water temperature from stored values.

Associated Changes

[7, 8] Algorithm for accurizing water temperature from a history of readings.

4. Issues

Issue 1: Should the decisions of how often to read sensors and how many sensors there are be hidden in one module?

Al: No. These are two distinct parameters of variation that can change independently and as independent decisions they should be hidden in different modules. In this structure we would have a sensor monitor module and a sensor reader module. (There could be one instance of the sensor reader module per sensor.) The sensor monitor module's responsibility is to inform the sensor reader module(s) what the SensorPeriod is. This could be done by the use of a SENSORPERIOD constant supplied at compile time to different instances of the sensor reader modules. Each instance could be implemented as a process. Each sensor reader module is then responsible for walling up at the specified period, reading its sensor, and storing its reading in the data banker for later retrieval. The sensor monitor module then becomes a module that runs at or before compile time.

A2: Yes. The decisions are not as independent as they appear. Because of timing constraints, adding sensors means that the program may not have time to read sensors as often. In this structure we would have a sensor monitor module and a sensor reader module. (There could be one instance of the sensor reader module per sensor.) The sensor monitor module's responsibility is to obtain sensor data from the sensor reader modules. The sensor reader module's responsibility is to provide sensor data when it is invoked.

Resolution:

Alternative Al: would give us more independence in changing the values for SensorCount and SensorPeriod, except that we would still have to recalculate the system timing when we changed either. It might require slightly less run-time code, and would probably increase the amount of process switching overhead. Since SensorCount and SensorPeriod are not as independent of each other as we would like because of their joint effect on system timing, and since alternative A2 may require less process switching, we will choose A2.

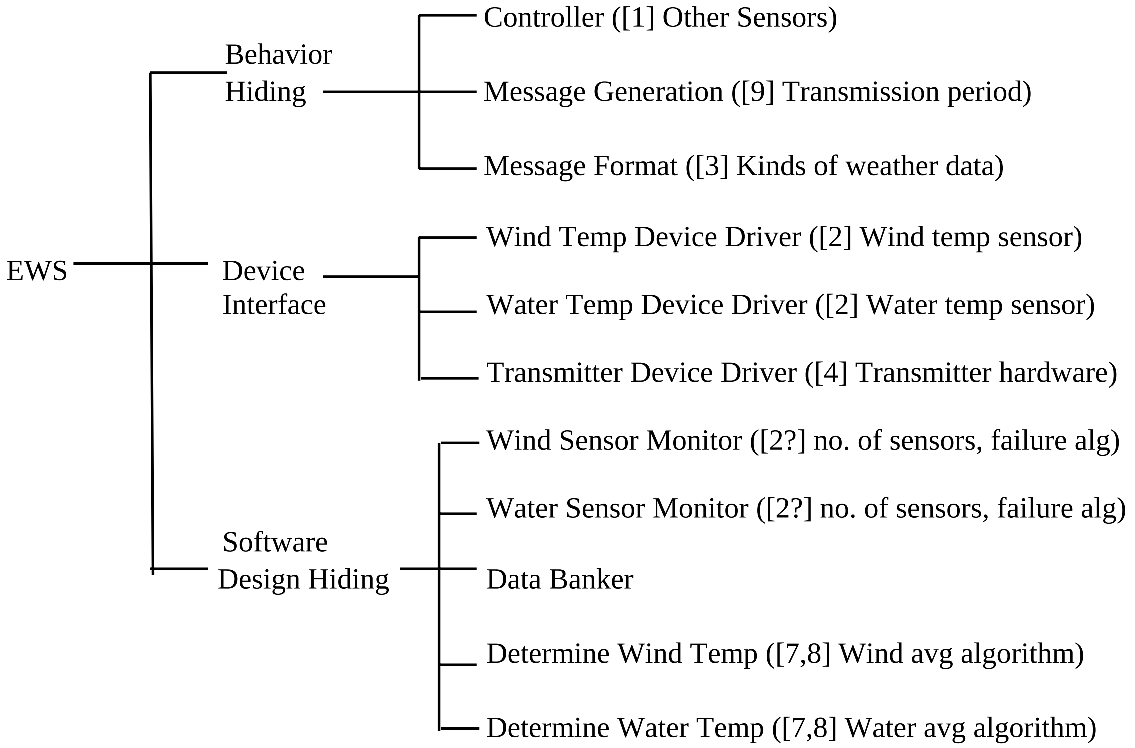
EWS
Behavior Hiding
Controller ([1] Other Sensors)
Message Generation ([9] Transmission period)
Message Format ([3] Kinds of weather data)
Device Interface
Wind Temp Device Driver ([2] Wind temp sensor)
Water Temp Device Driver ([2] Water temp sensor)
Transmitter Device Driver ([4] Transmitter hardware)
Software Design Hiding
Wind Sensor Monitor ([2?] no. of sensors, failure alg)
Water Sensor Monitor ([2?] no. of sensors, failure alg)
Data Banker
Determine Wind Temp ([7,8] Wind avg algorithm)
Determine Water Temp ([7,8] Water avg algorithm)

FIGURE 1. Floating Weather Station Module Hierarchy with Mapping to Changes