ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 1, Issue 10, December 2012

Object Oriented to Aspect Oriented: A Case Study

Harpreet Kaur

Abstract - **This work is to present the object oriented software into the aspect oriented software. And also highlight the tangling and scattering problem in the object oriented software development. There is a problem of structural mismatch between the specification of requirements for software systems and the specification of object-oriented software systems. The problem is that it does not address cross-cutting concerns. Aspect Oriented Software development, a programming paradigm that aims to support the modularization of crosscutting concerns in software. It is emerging software that helps to untangle the tangling by crosscutting concerns. There are aspects that are used to untangle the problem.**

Keywords- Aspect Oriented , Object Oriented

I. INTRODUCTION

In this, Weather Monitoring System, the system selected for case study are developed using Object-Oriented paradigm. Scattering and tangling problems were identified in these systems and were removed by using Aspect-Oriented paradigm. The main intent of this chapter is to present an exemplary case showing how object-oriented (OO) design patterns can be redesigned into pure aspect-oriented (AO) design patterns.

The Weather Monitoring is a Data Acquisition System. This system provides automatic monitoring of various weather conditions such as wind speed and direction, temperature, biometric pressure and humidity. This system also provides derived measurements like wind chill, dew point temperature, temperature trend and barometric pressure trend.

The rest of the Chapter is organized as follows: Section II presents the system in Object Oriented Approach and presents the problem of Weather monitoring System. Section III presents the problem formulation of the work in Aspect Oriented.

II. WEATHER MONITORING SYSTEM IN OO

The Weather Monitoring Sensor System shall have a means of determining the current time and date, so that it can report the highest and lowest values of nay of the four primary measurements during 24-hour period. The system allows calibrating its sensors against known values and setting the current time and date. The object-oriented techniques is used to design a data acquisition system allows to isolate the hardware that measures and collects the data from the application that then analyzes the information. Robust architecture can be defined to allow for sensors and devices to be added or replaced without disturbing the architecture of the control application. Fig.1 shows the Deployment Diagram for Weather Monitoring System. This figure contains the devices keypad, clock, graphic display, sensors- Wind direction sensor, wind speed sensor, temperature sensor, humidity sensor and pressure sensor and processor as CPU.

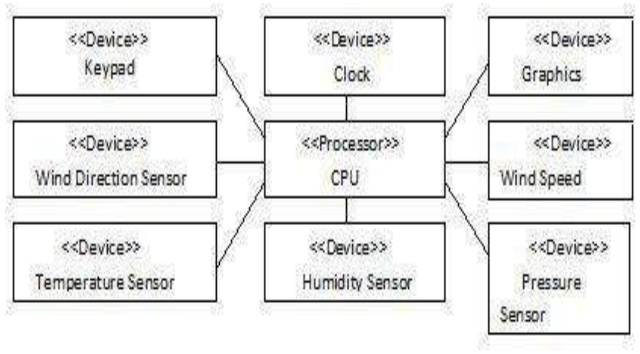
<<Device>> Keypad
<<Device>> Clock
<<Device>> Graphics
<<Device>> Wind Direction Sensor
<<Processor>> CPU
<<Device>> Wind Speed
<<Device>> Temperature Sensor
<<Device>> Humidity Sensor
<<Device>> Pressure Sensor

Fig.1. The Deployment Diagram for the Weather Monitoring System

All devices are connected to the processor. The details of hardware interfaces can be easily insulated from software abstractions by wrapping a class around each interface. For example, a simple class is created for accessing the current time and date. Thus, this class is responsible for keeping track of the current time in hours, minutes, and seconds, as well as the current month, day, and year.

The Sensor class, Historical class and Tender class act as the super class of other classes. In this, there are some common features in the super classes that are inherited by the subclasses.

Enumerated a number of primary use cases, as viewed from the point of view of the clients of the system:

1) Monitoring basic weather measurements, including wind speed and direction, temperature, barometric pressure, and humidity.
2) Monitoring derived measurements, including wind chill, dew point, temperature trend and barometric pressure trend.
3) Displaying the highest and lowest values of a selected measurement
4) Setting the time and date
5) Calibrating a selected sensor
6) Powering up the system

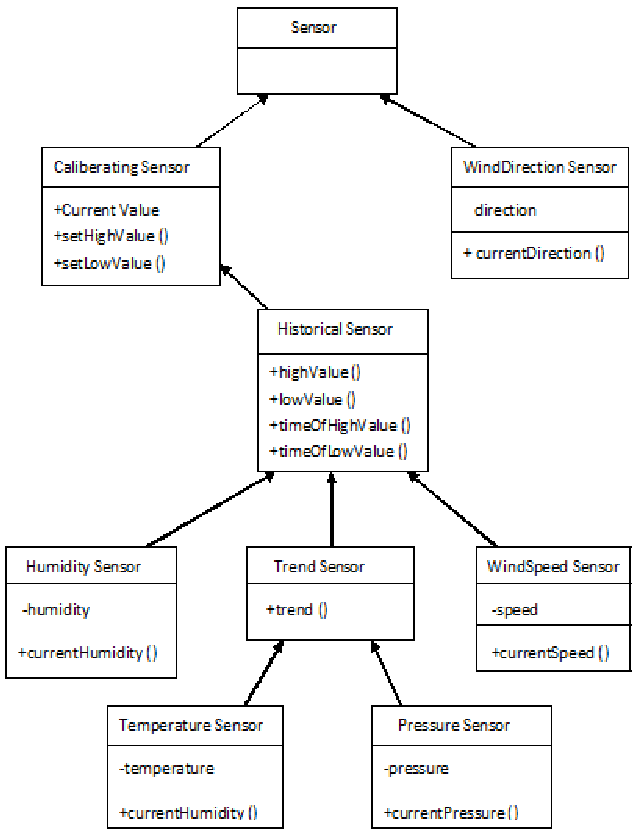

Fig.2. The Hierarchy of the Sensor Class

Enumerated a number of primary use cases, as viewed from the point of view of the clients of the system:

7) Monitoring basic weather measurements, including wind speed and direction, temperature, barometric pressure, and humidity.
8) Monitoring derived measurements, including wind chill, dew point, temperature trend and barometric pressure trend.
9) Displaying the highest and lowest values of a selected measurement
10) Setting the time and date
11) Calibrating a selected sensor
12) Powering up the system

The Data Acquisition System is essentially monitoring software and therefore, is a collection of modules that work together to provide the desired functionality defined by the set of requirements. ‘Separation of concerns’ and ‘modularity’ are the fundamental principles that derive software evolution.

Using Object-Oriented Programming (OOP) methodology, goals of modular programming and separation of concerns are seldom realized due to presence of crosscutting concerns. This leads to problems of code scattering and tangling which increases the complexity of the software and degrade its quality.

Aspect oriented Programming (AOP) is a newly emerging methodology that aims to improve the modularity and quality of software by achieving better separation of concern. AOP define a new program construct- ‘aspect’, which is a software entity that implements crosscutting functionality in a modular way. This provide most promising solution for elimination of code scattering and tangling, thus overcome the limitations of OOP.

III. WEATHER MONITORING SYSTEM IN AO

Aspect Oriented Programming is a newly emerging methodology that aims to improve the modularity and quality of software by achieving better separation of concern. AOP define a new program construct- ‘aspect’, which is a software entity that implements crosscutting functionality in a modular way.This provide most promising solution for elimination of code scattering and tangling, thus overcome the limitations of OOP. The most successful AOP language: AspectJ. It is a general purpose AOP extension to Java. It adds to Java a few constructs: pointcuts, advice, intertype declarations and aspects. AspectJ is very useful in design and implementation; provide enhanced IDE support for programming in AspectJ. AOP methodology improves the modularity, decrease in complexity and enhancement to the reusability of software. It is claimed that AOP methodology is very powerful and using it through AspectJ can enable development of concise, modular, efficient, flexible and cost-effective source code in shorter span of time. Increase in understandability, adaptability and maintainability are also reported.

Separation of concerns and modularity are the heart of programming process. These help in breaking down complex problem into smaller parts and solved them individually. Following this approach, the system is structured into units of function and behavior which can be put together to produce a complete software system.

From the study and analysis of the system fig.2, it is noticed that some functionalities cannot be localized into single modular units using OOP methodology. They represent crosscutting concerns. Value, timer, sampler, monitoring, display manager are such functionalities. Apart from these there are three more infrastructure concerns in the system, which are crosscutting: Failure checking, Authorization, Logging. Thus there are eight crosscutting concerns in the data acquisition system.

Identification of Aspects

In section II crosscutting concerns have been identified, which cause code scattering and tangling and increase the complexity of the system. They can be best modularized as aspects using AOP methodology.

A. Value Aspect

Any sensor nodes value is changes according to some modifications, the values must be set after each and every modifications call setvalue function and whenever any sensor needs historical values for further implementations they call highvalue and lowvalue function. So, instead of calling these functions repeatedly in different modules: Value aspect is created. This eliminates the code scattering and reduces complexity.

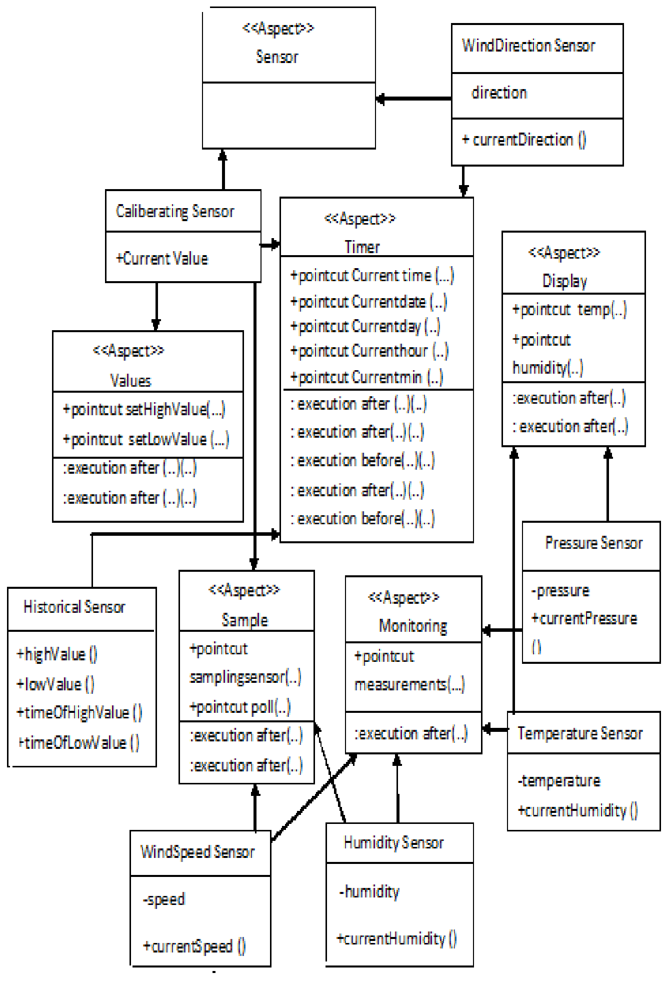

Fig.3. Shows the class diagram

B. *Timer Aspect*

The starting and finishing time and date of any sensor node is checked before and after execution of the modules. These classes keeping track of current time in hours, minutes and seconds, as well as current day and year. The operation current time and date perform these functions. The modifications are part of other module: time and date. In order to eliminate code tangling between these modules Timer aspect is created.

C. *Sample Aspect*

When the agent begins sampling, it polls each sensor inturn but intentionally skip certain sensors. This aspect contains the sample of each sensor and sampling rate, and polls each sensor.

D. *Monitoring Aspect*

Monitoring Aspect is used to monitor the weather measurements and derived measurements.

E. *Display _Manager Aspect*

Used to manage the layout of items on the LCD device like display time, date, temperature, humidity. There are various display functions in each module which create complexity, to reduce the complexity, the Display_Manager aspect is created.

F. *Logging Aspect*

Whenever the values are updated, the log file is maintained for each of them. These files record the details to values, time and date. This is handled by Logging Aspect.

G. *Failure Aspect*

There are two types of failure Power failure and sensor failure. These failures are handled by Failure Aspect.

H. *Scheduler Aspect*

It traces the execution of schedule module within the system. This is useful in understanding the system.

I. *Authorization_checking Aspect*

Only authorized persons should be allowed to log in, log out or perform transactions or database modifications etc. This crosscutting concern is best handled by adding Authorization_checking Aspect to programming mode.

Crosscutting concerns have been modeled as aspects using the join point model of AspectJ. Each aspect has its well defined join points, pointcuts and advice. Fig.3 shows the class diagram that showing classes and functional aspect.

IV. CONCLUSION

In this, the Weather Monitoring System as the object oriented software development and redesigned it in pure aspect oriented design. In the Weather Monitoring System there are problems of crosscutting concerns that have been identified, which cause code scattering and tangling and increase the complexity of the system in the Object Oriented paradigm. They can be best modularized as aspects using AOP methodology. In this paper, various aspects are introduced in Weather Monitoring like Value Aspect that shows that any sensor nodes value is changes according to some modifications, the values must be set after each and every modifications call setvalue function and whenever any sensor needs historical values for further implementations they call highvalue and lowvalue function. So, instead of calling these functions repeatedly in different modules: Value aspect is created. This eliminates the code scattering and reduces complexity. The another one is Timer Aspect that checked starting and finishing time and date of any sensor node is before and after execution of the modules. Monitoring Aspect is used to monitor the weather measurements and derived measurements. Authorization_checking Aspect, in this only authorized persons should be allowed to log in, log out or perform transactions or database modifications etc. he values are updated, the log file is maintained for each of them. These files record the details to values, time and date. This is handled by Logging Aspect.

V. FUTURE WORK

Future work will involve actual implementation of our case study. The Weather Monitoring case will now move into actual coding using Java Framework.

REFERENCES

[1] Pedro J. Clemente and Juan Hernandez,"Aspect Component based Software Engineering", Preceding the Sec. Int. Conf. on Aspect-Oriented Software Development (AOSD 2003), pp. 39-42, 2003.

[2] J.Whittle and J.Araujo "Scenario modeling with aspects", IEE Proc. Softw. , Vol. 151, No.4, August 2004pp. 157-171, 2004.

[3] Grady Booch, Robert A. Maksimchuk, Michael W. Engle, Bobbi J. Young, Ph.D. Jim Conallen and Kelli A.Houston, "Object-Oriented Analysis and Design with Applications", Third Edition , 2010.

[4] Amit Sharma "Event Managemnet System: Design and Implementation using AOP Methodology in Eclipse-AJDT Enviornment", International Journal of Engineering Science and Technology(IJEST), pp. 139-149, 2011.

Author Harpreet Kaur has completed M.tech (CSE) from Department of Computer Science and Application from Kurukshetra University, Kurukshetra, India in june 2012.