NOAA Technical Memorandum ERL AOML-89

OBJECT-ORIENTED DESIGN OF A NEAR REAL-TIME MARINE ENVIRONMENTAL DATA ACQUISITION AND REPORTING SYSTEM

J.C. Hendee

Atlantic Oceanographic and Meteorological Laboratory
Miami, Florida
September 1996

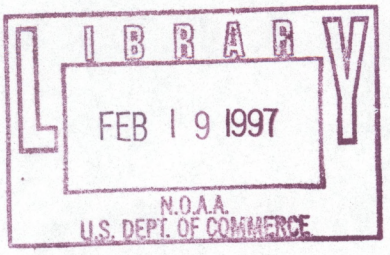
noaa NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION / Environmental Research Laboratories

NOAA Technical Memorandum ERL AOML-89

OBJECT-ORIENTED DESIGN OF A NEAR REAL-TIME MARINE ENVIRONMENTAL DATA ACQUISITION AND REPORTING SYSTEM

James C. Hendee

Ocean Chemistry Division



Atlantic Oceanographic and Meteorological Laboratory
Miami, Florida
September 1996

NOTICE

Mention of a commercial company or product does not constitute an endorsement by the NOAA Environmental Research Laboratories. Use of information from this publication concerning proprietary products or the tests of such products for publicity or advertising pruposes is not authorized.

Table of Contents

Object-Oriented Design of a Near Real-Time Marine Environmental Data Acquisition and Reporting System

by

James C. Hendee
Ocean Chemistry Division
Atlantic Oceanographic and Meteorlogical Laboratory
National Oceanic and Atmospheric Administration
4301 Rickenbacker Causeway
Miami, FL 33149-1026

Abstract--The National Oceanic and Atmospheric Administration's Coral Health and Monitoring Program in Miami, Florida has for the last several years worked cooperatively with the Florida Institute of Oceanography in monitoring meteorological and oceangraphic events at selected Coastal-Marine Automated Network sites in the Florida Straits. In a previous report, an Object-Oriented Analysis (OOA) was conducted of the existing system with an eye toward redesigning the system. This reports builds on the OOA results from the previous study and utilizes the lastest in Object-Oriented Design techniques to design a new system.

Keywords--Object-oriented analysis, object-oriented design, data management, oceanography, Florida Keys, coral, Coastal-Marine Automated Network, monitoring.

1.0 Introduction

In a previous paper (Hendee 1996) an object-oriented analysis (OOA) was conducted of a near real-time marine environmental data acquisition and reporting system, called the CoralFax system, that exists as part of the Coral Health and Monitoring Program (CHAMP) at the Ocean Chemistry Division, Atlantic Oceanographic and Meteorological Laboratory (OCD/AOML), National Oceanic and Atmopspheric Administration (NOAA) in Miami, FL. The method chosen for that study was that of Coad and Yourdon (1991a, 1991b), hereinafter referred to as C&Y, and one of the reasons mentioned was that there is a continuity in graphical notation between their OOA and object-oriented design (OOD) methodologies. Therefore, that method is also chosen here for the re-design of the new system. For a good description of how this notation differs from other popular notations, see Losavio et al (1994) and Wang (1995).

The system design described here is one which captures data from a satellite archiving facility in Wallups Island, VA, reformats it, then displays it in near real-time where environmental researchers have easy access to it. The data are collected from meteorological and oceanographic sensors housed upon lighthouses and buoys maintained by the National Data Buoy Center (NDBC, part of NOAA) and the Florida Institute of Oceanograpy (FIO). The network of these data collecting stations is called the Coastal-Marine Automated Network (C-MAN). For more information see Hendee (1996).

One of the differences between the C&Y OOA and standard systems analysis is that requirements gathering for a new system is absent in the C&Y OOA--it is apparently left for OOD. A redesign of the CoralFax system became desirable after the explosive growth of Internet and World-Wide Web (WWW) usage, as it became easier and more convenient for people to acquire environmental data via electronic mail (e-mail), file transfer protocol (FTP) or the WWW. Hence, the new design includes details for the broadcasting of data via the WWW and e-mail.

1.1 Object-Oriented Design

There have been many approaches to OOD, even in so few as the last six years. Some authors offer complete life-cycle approaches. Capretz and Lee (1993), for example offer Methodology for Object-Oriented Design (MOOD) which they describe as an interactive refinement process which sits between systems analysis and implementation. They mention four types of diagrams: composition, class hierarchy, object and operation. Henderson-Sellers and Edwards (1990) devise their own graphical notation and approach, based upon an amalgam of approaches by other authors. They describe the design stage as, "perhaps the most loosely defined since it is a phase of progressive decomposition toward more and more detail...and is

essentially a creative, not a mechanistic, process." They provide a very good overview on differences perceived by many authors among functional decomposition, Jackson structured development and object-oriented development. Their seven-step summary, even though an early one in the literature, probably best sums up most approaches today; however, what seems to be missing is a modeling of whatever current system is in use:

1) Undertake object-oriented system requirements specification

2) Indentify the objects (entities) and the services each can provide (interface)

3) Establish interactions between objects in terms of services required and services rendered

4) Analysis stage merges into design stage: use of lower-level EDFDs/IFDs (Entity Data Flow Diagrams/Information Flow Diagrams)

5) Bottom-up concerns --> use of library classes

6) Introduce hierarchical inheritance relationships as required

7) Aggregation and/or generalization of classes

In review of the literature, it appears that the most popular methodologies for OOD are Booch (1991), Rumbaugh et al (1991) and Coad and Yourdon (1991a, 1991b).

One interesting paper which offers good design heuristics is Lierberherr and Holland (1989), although their advice is more for actual programming than analysis and design strategies. They discuss the "Law of Demeter", a style guide for object-oriented programs which they claim makes for easier program maintenance. Other good heuristics may be found in Coad et al (1995), Nerson (1992), Coad (1992) and Wegner (1992).

Meyer (1992) has proposed an interesting concept, arrived at by revising material of other authors, called "design by contract" that is intended to make software more reliable. The "contract" is between two software modules which act in concert to perform a task. The client in the contract specifies how much should be done, and the contractor's task is to do as little as possible but still get the job done. McKim (1995) expands on Meyer's concept and provides some additional Eiffel code to Meyer's.

There have been several reviews of the different OOD methodologies. Some of the more notable and comprehensive ones include Eckert and Golder (1994, although they concentrate more on analysis than design), de Champeaux and Faure (1992), Fichman and Kemerer (1992), Graham (1991), Henderson-Sellers and Edwards (1990), Monarchi and Puhr (1992), and Wirfs-Brock and Johnson (1990),

2.0 Methods and Materials

Interestingly, on p. 38 of their text, Coad and Yourdon (1991b) say, "Some modifications are likely to occur that are modifications of the requirements specified during OOA." However, there is no place in the C&Y OOA that calls for a requirements analysis! Since the system under consideration was originally programmed and designed by the present author, interviews were not necessary to arrive at a list of requirements, at least for the present version. A list of requirements deemed desirable for the new system is included below.

The tool used to construct OOD diagrams using the C&Y notation is a tool produced by Peter Coad's group at Objects International, Inc. It is available as shareware over the WWW at Uniform Resource Locator http://www.oi.com. A summary of the notation is shown in Figure 1, while an explanation of the five layers described by this notation is found in Appendix 1. The current approach, however, adopts that of Wang's (1995) approach, namely, that cardinalities between systems are omitted because they are sometimes difficult to determine.

3.0 Results

The notational design of the new system is shown in Figure 2. This should be considered a provisional design, as the system has not been completely coded and at least some changes in the design should be expected.

In addition to determining the five layers outlined in Appendix 1, OOD entails four other activities, namely, designing:

- The Problem Domain
- The Human Interaction Component
- The Task Management Component
- The Data Management Component

In this study, the Problem Domain component needed a requirements specification to add to Classes that were still of use from the previously mentioned OOA. The Human Interaction Component was virtually non-existent, as the entire system is designed to be run automatically, that is, in a non-interactive mode. The Task Management Component forms an important portion of the system in that a particular sequence of operations must be followed, and the Data Management Component constitutes the largest element.

3.1 The Problem Domain Component

The C&Y OOD methodology offers this strategy towards designing the Problem Domain component:

- o Apply OOA
- o Use OOA results and *improve* them during OOD
- o Use OOA results and *add* to them during OOD

Some of the Classes discovered in the OOA of the present system (Hendee 1996) were eligible for use in the new design. Following is a description of requirements the new system is to contain.

3.1.1 Requirements Analysis

There are several levels of quality control (QC) for the C-MAN data as they arrive. The first level is range-checking. If an instrument is malfunctioning, there is a good probability that numbers that are recorded are way out of range. A simple range check would flag numbers that are grossly out of range for the parameter being measured. The next problem deals with accuracy. For instance, a salinometer instrument may measure a salinity of 35.5 practical salinity units (psu), but a "ground-truth" measurement, that is, a true measurement of salinity using either wet chemistry or an instrument calibrated immediately beforehand in the laboratory by wet chemistry, may yield a value of 35.0. This means that the measurement by the instrument in the field drifted 0.5 psu over time. The oceanographic instruments used in the field by FIO are calibrated every one to two weeks (sometimes four) to compensate for this drift, and the drift corrections are sent to NDBC for their final QC of the data. Another problem deals with precision of the instruments, but that is more of a cost issue than one of field logistics or data collection.

There is a chance, then that the near real-time data that are received are inaccurate, if slightly so. It would be desirable to have a system in which the data

that are posted are as near correct as possible, and to have the these data in a time frame of less than several months, as some time-sensitive features of the ecosystem in which direct sampling or witness by investigators might be missed. Another desired function of a new, near real- time data processing system would be the dissemination of these data via the Internet. Such a system would automatically transmit the C-MAN data via e-mail, and post them to a World-Wide Web Home Page.

It would also be helpful if there was an automated system for making sense out of the huge amount of data that are continually processed; that is, the provision of a data summary.

Finally, it would be highly desirable if the new system were written in a portable object-oriented programming (OOP) language that would allow easier development of new capabilities (something OOP languages are noted for) and for deployment under other platforms.

A new system is herewith proposed wherein the FIO Field Technician who is responsible for maintenance of the oceanographic instrumentation uploads a data file to a special FTP site which has the most recent or probable amount of drift corrections for each oceanographic instrument, as soon as possible after the corrections are determined. The new system reads those corrections each day when new data are distributed, and applies them to the incoming data. Those data are then processed through an Expert System (ES), such as used in Hendee (1995), to determine:

(a) if the data are within acceptable ranges for that time of year, at that station and for that time of day,
(b) what certain interrelationships of the data mean at that particular time,
(c) what real-time features might be occurring at that particular time that would be of immediate interest to environmental managers and research scientists.

To illustrate these eventualities, consider these cases. A sensor may read salinity within an acceptable range (say 32 to 36 psu), but during the rainy season historical data have shown that salinities over 34 near the surface are highly unlikely. Therefore, a sensor which reads 35 or 36 may be in error, and the data should be flagged as suspicious. In another case, a photosynthetically active radiation (PAR) sensor at 3 meters may read that the usual amount of sunlight available for phytoplankton is reaching that depth; on the other hand, a transmissometer, which measures the amount of light that passes through the water, might indicate highly turbid waters. In such a case, one of the instruments is probably malfunctioning, even though both are reading within acceptable ranges for the instrument. Finally, a number may be out of the typical range for that station, at that time of year, at that time of day, but it might indicate a real event. In fact, this happened during the summer of 1993 when flooding of the Mississippi River lowered salinities as far away as the Florida Straits! That is, salinities were abnormally low, but a real event was occurring. At

that particular time, first hand witnessing of the effects of the Mississippi River on the biota could not be be witnessed, because the lag between data collection and review were too great.

Critically reviewing daily data from six stations which are collecting numbers every hour is a difficult and time-consuming task. An ES that could be constructed to provide English- language constructs of data summaries would be most valuable. For instance, a 24 hour summary might read:

```
Molasses Reef:
PAR at surface remained high during the day.
PAR at 3 meters remained low.
=>
PAR at 3 meters:
     malfunction, biofouling, high turbidity, or
          Murphy Factor
```

In summary then, what is required in the new system is:

1) A near real-time ground-truth adjustment system

2) An Expert System which contains

    a) range checker

    b) data parameter cross-checker

    c) data summarizing facility

3) A portable programming language

3.2 The Human Interaction Component

The C&Y OOD strategy for designing this component consists of the following:

- o Classify the humans
- o Describe the humans and their task scenarios
- o Design the command hierarchy
- o Design the detailed interaction
- o Continue to prototype

- o Design the HIC Classes
- o Design, accounting for Graphical User Interfaces

3.2.1 Classify the Humans, Describe their Tasks

There are at least four groups of people who will use the system:

The FIO Field Technicians. Two field technicians maintain all six C-MAN stations. They will be responsible for conducting ground-truthing of salinity and sea temperatures at each of the stations. After they have determined the instrument reading adjustments necessary, they will upload a file (called TRUTH.DAT, see Appendix 2) to an FTP site accessible by the new system.

The System Operator. This person oversees operation of the system and initiates the program, which continues to run without interruption except for times of maintenance, breakdown, etc.

The Field Scientist. Scientists review the data over the WWW or by reviewing their e-mail. If they have questions or suggestions for improvement of the system, they forward them to the System Operator.

The Casual Browser. WWW users visit the CHAMP WWW site quite often. Some of them simply browse around to see what it is all about. This part of the complete system, however, actually belongs under a different problem domain, that of the WWW data management.

3.2.2 Describe the Command Hierarchy

The current system design has no interactive mode for the user of the data acquisition software. It is anticipated that the entire system will be automated and that once it is set into motion, no interaction by the operator will be necessary. Because of this, the remaining tasks under the Human Interaction Component are rendered moot.

3.3 The Task Management Component

The following strategy is outlined for this component:

- o Identify event-driven tasks
- o Identify clock-driven tasks
- o Identify priority tasks and critical tasks

- o Identify a coordinator
- o Challenge each task
- o Define each task

All of the tasks are managed within the Class-Object called Task_Manager in the order in which they are listed as Services.

3.3.1 Identify Event-Driven Tasks

There are no event-driven tasks under this design.

3.3.2 Identify Clock-Driven Tasks

All of the events under the Class-Object Task_Manager are initiated by a batch file called TIMER.BAT. This batch file starts with a shareware utility called WAITUNTL.EXE which takes a command line argument of a time to wait until, after which it quits. After a time of 04:10 hrs, WAITUNTL.EXE quits and the next line in the batch file TIMER.BAT executes. The command line arguments within TIMER.BAT are represented by the Services under Task_Manager.

3.3.3 Identify Priority Tasks and Critical Tasks

Since each task is sequential, and each depends on the one before it, every task of equal priority.

3.3.4 Identify a Coordinator

The coordinating task is TIMER.BAT.

3.3.5 Challenge Each Task

Although it appears that the number of tasks have been kept to a minimum, experience with the system may come to prove that some tasks may be consolidated.

3.3.6 Define Each Task

Some of the tasks below are exactly as described in the OOA of the previous system (Hendee 1996); hence, reference is made to that document where appropriate.

Run_ScriptGenerator. Same as in the OOA.

Run_ProcommScript. Same as in the OOA.

Run_DataRenamer. Same as in the OOA.

Run_SandKey.ST, Run_LongKey.ST, Run_Sombrero.ST, Run_Molasses.ST, Run_Fowey.ST, Run_Dry_Tortugas.ST. Unlike the system described previously in the OOA, the present system will run individual programs for each station, since each station has different data profiles. The .ST suffix represents that the code for the program is Smalltalk code.

Run_Expert_System. The ES will be run against each of the C-MAN stations (six total). One of the outputs for each station is intended to be a data file that has suspicious data points flagged; another output for each station is envisioned to be a data summary. Since this portion of the system has not yet been prototyped, it is not known whether just one ES will be best, or more.

Run_WebDocMaker. This service will run a small program which just makes a Hypertext Markup Language (HTML) document from the document from immediately above so that the data can be viewable over the WWW.

Run_E-MailSender. This service will run a command line e-mail program which will send the data bulletins to whomever has requested them.

3.4 The Data Management Component

The Data Management Component consists of four activities:

a) Data acquisition

b) Data extraction (parsing)

c) Data review

d) Data transmittal

3.4.1 Data Acquisition

All the data acquisition Classes described in the OOA were imported into the new design, and this serves to underscore the validity of the OOA/OOD approach. Even though the programming languages will be different, the Classes, Attributes and Services will not change. These Classes include ScriptGenerator, Procomm-mScript, ZyXEL_Modem, Procomm, RawDataFile, DataRenamer and ReNamedFile. See the OOA for a complete description of these Classes. For a look at a small part of the data stream in RawDataFile, see Appendix 3.

3.4.2 Data Extraction (Parsing)

The elegance of OOP is clearly illustrated in this portion of the Data Management Component. As suggested in the C&Y OOD methodology, a protocol was established by adding a generalization Class. As previously mentioned, the programming language chosen for this project was Smalltalk, due mainly to its portability but also to its pure object-oriented nature. Since all C-MAN stations measure the same base meteorological parameters, and some oceanographic ones, a basic Class named CmanBasic was constructed (see Appendix 4). The Services (methods) in this Class could be used by all C-MAN stations that inherited from it, and of course, Services that needed modifying for a particular C-MAN station could be redefined for that station. For instance, if you look at Figure 2 and note the Services defined for CmanBasic, then look at the Services for an inherited Class, Sand_Key, you will note that the Services windDirection and windSpeed are redefined in the latter Class. (Please note that Services have not been listed for all Classes in order to save space on the figure.) This is because those parameters are located at different positions in the data stream (Class RawDataFile, or file DAPS.DAT) for Sand Key than they are for most other C-MAN stations. For comparison, see the CmanBasic Class code and the Sand_Key Class code listed in Appendix 5.

In Figure 2 you will note that each of the six C-MAN stations which are represented with Smalltalk code (*.ST), also have an instance connection to an ASCII file resulting from the invoking of the program. For an example of the SandKey.ST Smalltalk program see Appendix 6; for an example of the resulting ASCII bulletin, see Appendix 7. To see how the reports now differ (as compared to those presented in the previous OOA, in which all reports had the same format), compare the Sand Key report with the Molasses Reef report shown in Appendix 8. Also notice that in the Molasses Reef report, garbage in the raw data file is now replaced with "nil".

3.4.3 Data Review

Note that in the CmanBasic Class (Appendix 4) a preliminary range check may be invoked if the programmer wishes. The methods rangeCheck (two versions) and fuzzyRangeCheck can provide a first level check of data ranges without the use of an ES. However, it is anticipated that in the final design of the system, an ES will handle all range checking and data summarization, as mentioned above. Unfortunately, no code has been developed specifically for this Class; however, it is anticipated that an approach like that of Hendee (1995) can be used.

3.4.4 Data Transmittal

Data will be transmitted to the user via the WWW in Class WebDocMaker, and via e-mail in Class E-MailSender. See the Services Run_WebDocMaker and Run_E-MailSender under the Class Task_Manager for a fuller explanation.

4.0 Discussion and Conclusion

Because the actual coding (that is, the Implementation Stage) of the current system is not complete, it is highly likely that certain details in the Design phase will change. For instance, the ES will likely evolve to a complex system, and in fact may actually become more than one ES. It is also probable that a Human Interaction Component will be desirable, so that users of the system can review older data. Finally, the current Data Management Component handles only flat file data. As object- oriented database management system (OODBMS) technology is learned, a shift to its use is probably in the future, and this will again require a change in the OOD. However, because of the strength in the notational approach inherent in OOD, and because OODBMS technology basically uses the existing OOP language (see Loomis 1995) in its implementation, it is felt that representation of new Classes will not be that difficult.

Much has been written in the literature about the chasm that exists between analysis and design. Experience gained in this study and the one before it for OOA (Hendee 1996) leave this author with the impression that the most difficult task in bridging this gap is in constructing a requirements analysis ahead of time. Constructing an OOA of an existing system is straight-forward , yet conducting a requirements analysis and an OOD require iterations until the system is built. It is interesting that Coad and Yourdon (1991a and 1991b) dance around this issue without firmly pinning it down. Since the same notation is used in C&Y OOA and OOD with good transition, it would seem that a useful methodology could be adopted and described by these authors to address the need of a clear approach to requirements analysis. Bailin (1989) believes that Entity Relationship Diagrams should be used in requirements analysis and gathering, and though this departs from the more modern approaches, at least he discusses the importance of the subject and identifies it as a separate activity from "analysis" and "design".

5.0 Bibliography

Bailin, S. 1989. An object-oriented requirements specification method. Comm. ACM 32(5): 608-623.

Booch, G. 1991. Object-oriented design: with applications. Benjamin/Cummings, Redwood City, CA.

Capretz, L. and Lee, P. 1993. Object-oriented design: guidelines and techniques. Information and Software Technology 35(4): 195-206.

Coad, P. 1992. Object-oriented patterns. Comm. ACM 35(9): 152-159.

Coad, P.; North, D and Mayfield, M. 1995. Object models. Strategies, patterns, and applications. Yourdon Press, Englewood Cliffs, NJ.

Coad, P. and Yourdon, E. 1991. Object-oriented analysis. Prentice-Hall, Englewood Cliffs, NJ.

de Champeaux, D. and Faure, P. 1992. A comparative study of object-oriented analysis methods. J. Object-Oriented Programming (April): 21-33.

Eckert, G. and Golder, P. 1994. Improving object-oriented analysis. Information and Software Technology 36(2): 67-86.

Fichman, R. and Kemerer, C. 1992. Object-oriented and conventional analysis and design methodologies. IEEE Computer, October: 22-39.

Graham, I. 1991. Object-oriented methods. Addison-Wesley, NY.

Hendee, J. 1995. PELAGOS: An expert system for quality control and feature recognition of oceanographic data from the open ocean. NOAA Tech. Memorandum ERL AOML-87.

Hendee, J. 1996. Object-oriented analysis of a near real-time marine environmental data acquisition and reporting system. NOAA Technical Memorandum ERL AOML-?? [to be submitted].

Henderson-Sellers, B. and Edwards, J. 1990. The object-oriented systems life cycle. Comm. ACM 33(9): 142-159.

Lieberherr, K. and Holland, I. 1989. Assuring good style for object-oriented programs. IEEE Software (September): 38-48.

Loomis, M.E.S. 1995. Object databases. The essentials. Addison-Wesley.

Losavio, F., Matteo, A. and Schlienger, F. 1994. Object-oriented methodologies of Coad and Yourdon and Booch: comparison of graphical notations. Information and Software Technology 36(8): 503-514.

McKim, J.; Jr. 1995. Class interface design and programming by contract. Proc. TOOLS Pacific. Prentice-all; Englewood Cliffs, NJ.

Meyer, B. 1992. Applying "Design by contract". IEEE Computer (October): 40-51.

Monarchi, D. and Puhr, G. 1992. A research typology for object-oriented analysis and design. Comm. ACM 35(9): 35-47.

Nerson, J-M. 1992. Applying object-oriented analysis and design. Comm. ACM 35(9): 63-74.

Rubin, K. and Goldberg, A. 1992. Object-behavior analysis. Comm. ACM 35(9): 48-62.

Rumbaugh, J. Blaha, M. Premerlani, W. Eddy, F. and Lorensen, W. 1991. Object-oriented modeling and design. Prentice Hall, Englewood Cliffs, N.J.

Shlaer, S. and Mellor, S.J. 1988. Object-oriented systems analysis: Modeling the world in data. Prentice Hall, Englewood Cliffs, NJ.

Wang, S. 1995. Object-oriented task analysis. Information & Management 29: 331-341.

Wegner, P. 1992. Dimensions of object-oriented modeling. IEEE Computer (October): 12-20.

Whitten, J.; Bentley, L.; and Barlow, V. 1994. Systems analysis and design methods, 3rd edition. Richard D. Irwin, Inc. Burr Ridge, IL.

Wirfs-Brock, J. and Johnson, R. 1990. Surveying current research in object-oriented design. Comm. ACM 3(9): 104-124.

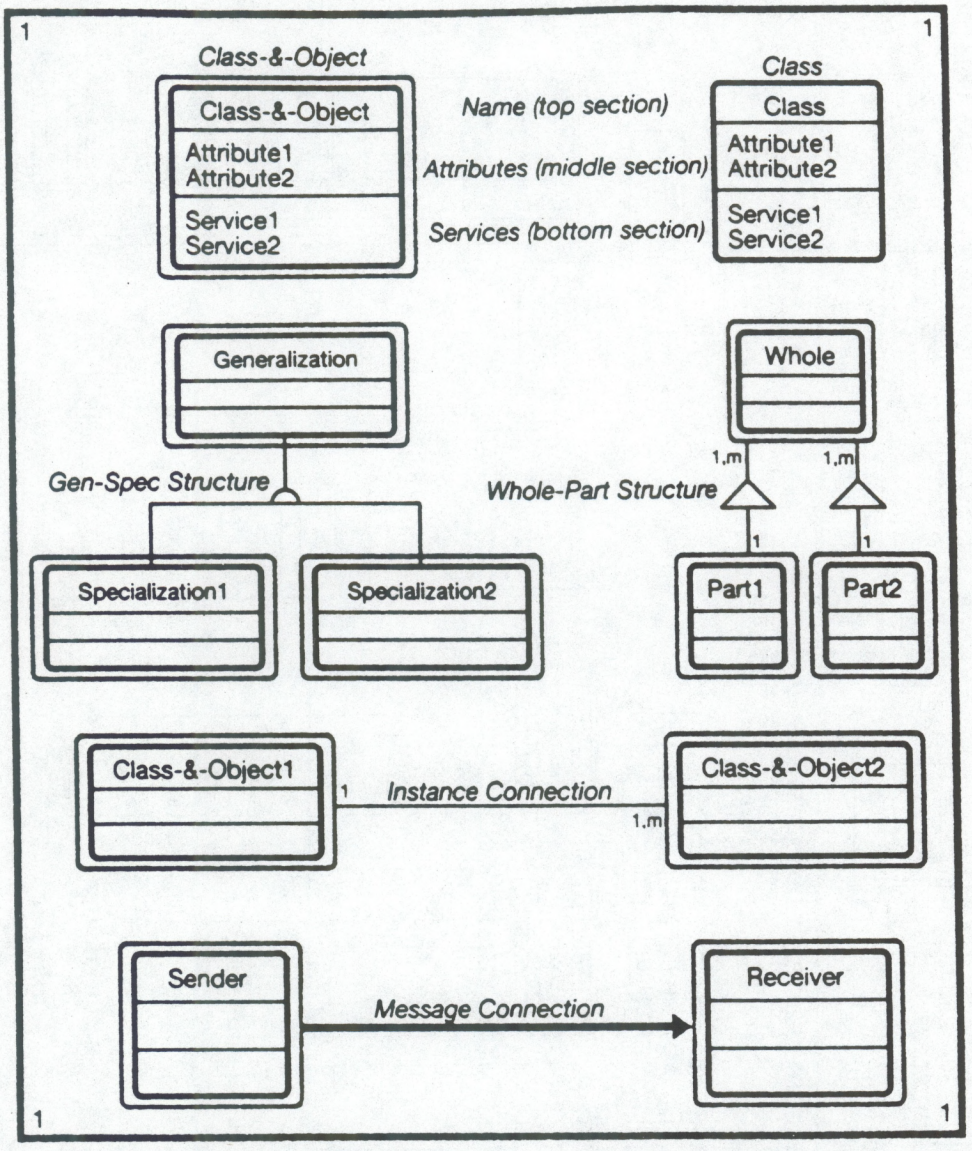


Figure 1. OOA notations. (From Coad and Yourdon 1991a, 1991b).

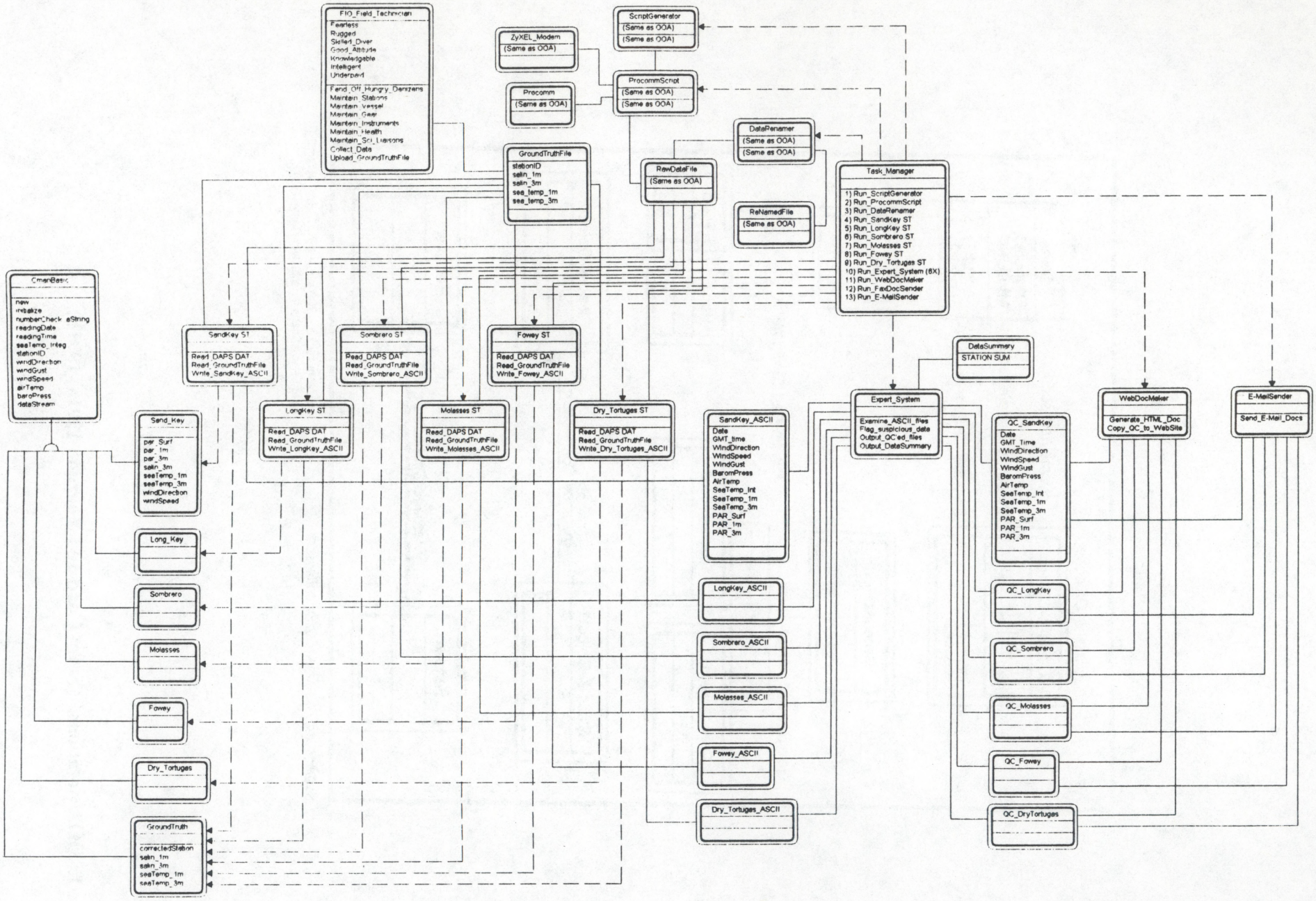


Figure 2. Notation representing an object-oriented design of a new NOAA/AOML/OCD near real-time marine environmental data acquisition and reporting system.

Appendix 1

Object-oriented Analysis
Notations and Strategies
(from Coad and Yourdon 1991a)

Object. An abstraction of something in a problem domain, reflecting the capabilities of a system to keep information about or interact with it; an encapsulation of Attribute values and their exclusive Services.

Class. A description of one or more Objects with a uniform set of Attributes and Services, including a description of how to create new objects in the Class.

Structure. Structure is an expression of problem-domain complexity, pertinent to the system's responsibilities. The term "Structure" is used as an overall term, describing both Generalization-Specialization (Gen-Spec) and Whole-Part Structures.

Gen-Spec Structures. For potential generalizations and specializations, ask:

Is it in the problem domain?
Is it within the system's responsibilities?
Will there be inheritance?
Will the specializations meet the criteria for Class & Objects?

Whole-Part Structures. Consider these variations:

Assembly-Parts
Container-Contents
Collection-Members

Consider each Object as a whole. For its potential parts, ask:

Is it in the problem domain?
Is it within the system's responsibilities?
Does it capture just a status value?
Does it provide a useful abstraction in dealing with the problem domain?

Subject. A Subject is a mechanism for guiding a reader (analyst, problem domain expert, manager, client) through a large, complex model. Subjects are also helpful for organizing work packages on larger projects, based upon initial OOA investigations.

Attribute. An Attribute is some data (state information) for which each Object in a Class has its own value.

Instance Connection. An Instance Connection is a model of problem domain mapping(s) that one Object needs with other Objects, in order to fulfill its responsibilities.

Service. A Service is a specific behavior that an Object is responsible for exhibiting.

Message Connection. A Message Connection models the processing dependency of an Object, indicating a need for Services in order to fulfill its responsibilities.

Appendix 2

Example Ground-Truth Data File,
TRUTH.DAT

Sta	Sal_1m	Sal_3m	Tem_1m	Tem_3m
FWYF1	+00.20	+00.00	+00.00	+00.00
MLRF1	+00.10	+00.00	+00.00	+00.00
LONF1	+00.20	+00.00	+00.00	+00.00
SMKF1	IOOW	+00.00	+00.00	+00.00
SANF1	+00.40	+00.00	+00.00	+00.00
DRYF1	+01.20	+00.70	+00.00	+00.00

Appendix 3

Portion of File DAPS.DAT
(Class RawDataFile)

2240C45C96068090615G53-0NN015EFF0032530858 0900 MLRF1 46/// /237096(/195114) 10241 30106(3////) 40119(4////)
222// 00228 333 921102(921121) WWW233093(192110)W231079(186099)W235065(1880
84)W239053(186076)W234057(183080)W238063(189 085)W23610657(19612355)W235068(187089)W015(0 13) S1111
A10019 A20020 FIO000034 22.88 53.3 35.2 15.5
////////////////

2240C45C96068080615G54-0NN015EFF0032528858 0800 MLRF1 46/// /239067(/190087) 10241 30113(3////) 40127(4////)
222// 00229 333 921068(921089) WWW233068(188086)W222060(180075)W215060(1740
78)W214059(172074)W211051(169063)W199057(158 070)W18907731(18809257)W216059(174074)W007(0 08) S1111
A10019 A20020 FIO000034 23.01 53.6 35.4 15.6
////////////////

2240C45C96068070615G54-0NN015EFF0032530858 0700 MLRF1 46/// /192062(/154074) 10240 30111(3////) 40124(4////)
222// 00227 333 921065(921079) WWW196059(157072)W205056(165068)W210055(1710
66)W216054(177064)W226050(185060)W228049(186 059)W19406559(15507959)W214054(174065)W004(0 05) S1111
A10019 A20020 FIO000034 22.89 53.7 35.5 15.6
////////////////

Appendix 4

Source Code for the Digitalk /V Smalltalk
Class CmanBasic

```
Object subclass: #CmanBasic
 instanceVariableNames:
  ' dataStream '
 classVariableNames: ''
 poolDictionaries: ''  !

!CmanBasic class methods !

new: aStringOfData
      "initialize"
  ^super new initialize: aStringOfData! !

!CmanBasic methods !

airTemp
      "put parsed stuff together"
   | theNumber major minor |

   major := dataStream copyFrom: 80 to: 81.
   minor := dataStream copyFrom: 82 to: 82.
   theNumber := major, '.', minor.
   theNumber = '   .  '
      ifTrue: [ theNumber := '------' ].

   ^theNumber!

baroPress
      "barometric pressure"
   | theNumber major minor gale |

major := dataStream copyFrom: 85 to: 87.
   minor := dataStream copyFrom: 88 to: 88.
gale := dataStream copyFrom: 85 to: 85.
gale = '0'
ifTrue:          "no low pressure"
[^theNumber := '1',major, '.', minor]
ifFalse:         "low barometric pressure"
[^theNumber := ' ',major,'.',minor]
!

dataStream
      "Not needed here but often useful accessor method"
   ^dataStream!

fuzzyRangeCheck: aString from: aNumber1 to: aNumber2 from: aNumber3
     to: aNumber4 from: aNumber5 to: aNumber6
   "Change aString to number, then return whether number is low, medium,
high, or none of those, in expected range."
     | stringNum result |
result := '? ',aString.
stringNum := aString asFloat.
```

```
( (aNumber1 <= stringNum) & ( stringNum <= aNumber2) )
ifTrue:  [ result := 'L  ',aString ].
( (aNumber3 <= stringNum) & ( stringNum <= aNumber4) )
ifTrue:  [ result := '   ',aString ].
( (aNumber5 <= stringNum) & ( stringNum <= aNumber6) )
ifTrue:  [ result := 'H  ',aString ].
^result!

initialize: aStringOfData
      "assign argument"

   dataStream := aStringOfData.!

numberCheck: aString
   "If any characters in the string, *except* a decimal,
return a flag indicating bad data stream."
| input flag |
   input := aString.
   input do: [ :char |
     ( (char asciiValue >= 48) & (char asciiValue <= 57) )
        ifFalse:[
     char asciiValue = 46  "in case there is a decimal place"
ifFalse:[
     flag := 'nil  '.
     ^flag ]
                ]
        ifTrue:[ flag := aString]
              ].
   ^flag.
!

rangeCheck: aString from: aNumber1 to: aNumber2
   "Change aString to number, then see if it is the range aNumber1 to aNumber2"

| stringNum |
stringNum := aString asFloat.

( (aNumber1 <= stringNum) & ( stringNum <= aNumber2) )

ifTrue:  [ ^'  ',aString ]
ifFalse: [ ^'* ',aString]!

rangeCheck: aString from: aNumber1 to: aNumber2 exclude: aNumber3
   "Change aString to number, then see if it is the range aNumber1 to aNumber2"

| stringNum |
stringNum := aString asFloat.
((aNumber1 <= stringNum) & ( stringNum <= aNumber2))
ifTrue:  [ ^'  ',aString ]
ifFalse: [
stringNum = aNumber3
              ifTrue: [ ^'  nil ']
ifFalse: [ ^'* ',aString]].!

readingDate
"return Gregorian date from Year-Julian date string, e.g.,
96060, which would equal 02/29/96.  I do not have the brains to tell
if this will screw up during 12/31/1999 to 01/01/2000"

| year julianDay yearPrefix |
```

```
year        := dataStream copyFrom: 9 to: 10.
julianDay  := dataStream copyFrom: 11 to: 13.

"Notice that in the next line I'm assuming that the reading date was
made in the same millenium as the current one (today)."
yearPrefix := (Date today year asString) copyFrom: 1 to: 2.

^(Date newDay: (julianDay asNumber)  year: ((yearPrefix,year) asNumber)) asString.
!

readingTime
      "put parsed stuff together"
   | theNumber |

   ^theNumber := (dataStream copyFrom: 14 to: 15),'00'
!

seaTemp_Integ
      "parse out sea temperature integrated through the water column"
   | major minor |

     major := dataStream copyFrom: 118 to: 119.
minor := dataStream copyFrom: 120 to: 120.

   ^major,'.',minor!

stationID
      "Find 5-character station identifer"
   | station |

   station := dataStream copyFrom: 49 to: 53.

   ^station!

windDirection
      "parse wind direction from data stream"
 ^dataStream copyFrom: 65 to: 67.
!

windGust
      "put parsed stuff together"
   | theNumber major minor gale |

gale := dataStream copyFrom: 129 to: 131.
gale = '999'
                ifFalse:
[major := dataStream copyFrom: 129 to: 130.
                minor := dataStream copyFrom: 131 to: 131.
                theNumber := major, '.', minor]
ifTrue:         "bummer for South Florida..."
[major := dataStream copyFrom: 136 to: 137.
                minor := dataStream copyFrom: 138 to: 138.
                theNumber := major, '.', minor].

   ^theNumber!

windSpeed
      "put parsed stuff together"
   | theNumber major minor |
```

```
major := dataStream copyFrom: 65 to: 66.
minor := dataStream copyFrom: 67 to: 67.
theNumber := major, '.', minor.
theNumber = '   .  '
   ifTrue: [ theNumber := '------' ].

^theNumber! !
```

Appendix 5

Source Code for Digitalk /V Smalltalk
Class SandKey

```
CmanBasic subclass: #SandKey
 instanceVariableNames: ''
 classVariableNames: ''
 poolDictionaries: ''   !

!SandKey class methods ! !

!SandKey methods !

par_1m
      "parse out photosynthetically active radiation at surface"
   ^dataStream copyFrom: 292 to: 295
!

par_3m
      "parse out photosynthetically active radiation at 3 meters"
   ^dataStream copyFrom: 299 to: 302
!

par_Surf
      "parse out photosynthetically active radiation at surface"
   ^dataStream copyFrom: 285 to: 288
!

salin_3m
   "parse out salinity at 1 meter"
   ^dataStream copyFrom: 339 to: 342
!

seaTemp_1m
      "parse out sea temperature integrated through the water column"
   | major minor |

     major := dataStream copyFrom: 307 to: 308.
minor := dataStream copyFrom: 309 to: 309.

   ^major,'.',minor!

seaTemp_3m
      "parse out temperature at 3 meters"
   | major minor |

   ^dataStream copyFrom: 322 to: 326
!

windDirection
      "parse wind direction from data stream"

   ^dataStream copyFrom: 70 to: 72
!

windSpeed
```

```
    "put parsed stuff together"
  | theNumber major minor |

  major := dataStream copyFrom: 73 to: 74.
  minor := dataStream copyFrom: 75 to: 75.
  theNumber := major, '.', minor.
  theNumber = '  .  '
    ifTrue: [ theNumber := '------' ].

  ^theNumber
! !
```

Appendix 6

Source Code for Digitalk /V Smalltalk
Program SandKey.ST

```
"Program to parse Sand Key C-MAN data out of DAPS.DAT file.
   by James C. Hendee
   03/23/96"

inputFile   := File pathName: 'd:\temp\daps.dat'.       "Raw data file."
" put in something to kill preexisting file, otherwise, you get a funky overwrite..."
outputFile := File pathName: 'd:\temp\sandkey.dat'.    "File to record results."
truthFile    := File pathName: 'd:\temp\truth.dat' .      "File with groundtruth data."

outputFile
   nextPutAll: '  Coastal-Marine Automated Network (C-MAN)';
   nextPut: Lf;
   nextPutAll: '          -- Sand Key (SANF1) --';
   nextPut: Lf;nextPut: Lf;
   nextPutAll: 'Date     GMT   WD   WS    WG    BAROM   AT    ST-I ST-1  ST-3    SAL3 PARs PAR1 PAR3'; nextPut: Lf.

groundTruth := GroundTruth new.

[truthFile atEnd]   "Assign drift corrections (ground truth data) to variables
                     so you can correct data read from instrument (inputFile)."
whileFalse:
   [
   aLine := truthFile nextLine.    "include test for aLine size = 0?"
   groundTruth initialize: aLine.
   truthStation := groundTruth correctedStation.
   truthStation = 'SANF1'
      ifTrue:
         [sanf1Salin_3m := groundTruth salin_3m.].
   ].

truthFile close.

[inputFile atEnd]
whileFalse:
   [
   aLine    := inputFile nextLine.
   station  := CmanBasic new.      "Don't know what the station is yet, so assume basic."
   station initialize: aLine.

   aLine size >  180    "Don't use trashed lines?  What size to use!!?"
      ifTrue:
      [
      chosenStation := station stationID.
      chosenStation = 'SANF1'
         ifTrue:
            [
            station := SandKey new.
            station initialize: aLine.
            " *** SALINITY at 1m *** "
            " ~~~~~ First check to see if string is garbage. ~~~~~"
            (station numberCheck: (station salin_3m)) =  'nil  '
```

```
        ifFalse:[                       " Apply groundtruth adjustments to parameters here. "
          sanf1Salin_3m = 'IOOW'    " Instrument Out Of Water. "
          ifFalse:[
              salin_3m :=  (((station salin_3m) asFloat) + ((sanf1Salin_3m)  asFloat)) asString.

              "This is a simple range checker, but be careful, because the exclude
                   means that number (0.0) can never be a good number!"
              salin_3m := station rangeCheck: salin_3m from: 34.5 to: 36.5 exclude: 0.0.
              "This assigns subject range of ? (say what?), L (low), okay (no mark) or H (high):"
              "salin_1m := station fuzzyRangeCheck: salin_3m
                                  from: 33.0 to: 34.4
                                  from: 34.5 to: 36.0
                                  from: 36.1 to: 37.0."
                ]
          ifTrue: [ salin_3m := 'IOOW'].
          ]
        ifTrue:[ salin_3m := '  nil '].    " Namely, you got garbage. "

        outputFile
          "nextPutAll: chosenStation;"
          nextPutAll: (station readingDate);
          nextPutAll: (' ',(station readingTime));
          nextPutAll: (' ',(station windDirection));
          nextPutAll: (' ',(station windSpeed));
          nextPutAll: (' ',(station windGust));
          nextPutAll: (' ',(station baroPress));
          nextPutAll: (' ',(station airTemp));
          nextPutAll: (' ',(station seaTemp_Integ));
          nextPutAll: (' ',(station numberCheck: (station seaTemp_1m)));
          nextPutAll: (' ',(station numberCheck: (station seaTemp_3m))); " no groundtruth "
          nextPutAll: (' ',salin_3m);    " groundtruth checked "
          nextPutAll: (' ',(station par_Surf));
          nextPutAll: (' ',(station par_1m));
          nextPutAll: (' ',(station par_3m));
          nextPut: Lf.   "end of record"
        ].
    ].
  ].

inputFile close.
outputFile close.
```

Appendix 7

Sample Report from SandKey.ST

Coastal-Marine Automated Network (C-MAN)
-- Sand Key (SANF1) --

Date	GMT	WD	WS	WG	BAROM	AT	ST-I	ST-1	ST-3	SAL3	PARs	PAR1	PAR3
03/08/96	0900	268	10.7	11.9	1011.4	24.4	23.6	23.8	23.78	36.2	0022	0100	0048
03/08/96	0800	312	05.9	07.2	1011.5	24.0	23.7	23.9	23.91	36.2	0020	0100	0048
03/08/96	0700	246	13.2	14.8	1011.9	24.5	23.7	23.9	23.89	36.2	0020	0101	0048
03/08/96	0600	222	07.5	08.9	1012.4	24.8	23.8	24.0	23.95	36.2	0020	0101	0049
03/08/96	0500	179	06.2	06.8	1012.9	24.8	23.8	24.1	24.00	36.2	0020	0101	0048
03/08/96	0400	160	05.6	06.5	1012.3	24.5	23.9	24.2	24.02	36.2	0020	0101	0049
03/08/96	0300	135	07.8	08.5	1012.4	24.5	23.9	24.2	24.13	36.2	0020	0101	0049
03/08/96	0200	115	03.4	03.7	1012.9	24.4	24.0	24.2	24.18	36.2	0021	0100	0049
03/08/96	0100	128	03.8	04.2	1012.2	24.5	23.9	24.1	24.20	36.2	0021	0101	0049
03/08/96	0000	155	02.8	03.1	1011.6	24.8	24.0	24.1	24.22	36.1	0021	0100	0049
03/07/96	2300	213	03.1	03.6	1011.3	25.2	24.1	24.4	24.29	36.1	0096	0136	0052
03/07/96	2200	210	03.9	04.5	1010.8	25.2	24.2	24.6	24.31	36.2	0333	0280	0183
03/07/96	2100	173	05.5	06.1	1010.5	25.2	24.1	24.7	24.31	36.1	0578	0428	0313
03/07/96	2000	188	06.5	07.4	1011.0	25.1	24.1	24.8	24.33	36.1	0779	0571	0413
03/07/96	1900	207	07.5	08.5	1011.9	25.0	24.1	24.6	24.27	36.1	0926	0725	0530
03/07/96	1800	198	08.5	08.8	1012.5	25.0	24.2	24.6	24.44	36.0	0998	0667	0538
03/07/96	1700	196	09.4	10.8	1013.3	25.0	23.7	24.1	24.06	35.8	0986	0611	0478
03/07/96	1600	176	10.7	12.4	1013.7	24.8	23.2	23.6	23.37	35.5	0889	0547	0383
03/07/96	1500	174	12.3	14.6	1013.9	24.6	23.1	23.5	23.33	35.6	0746	0446	0308
03/07/96	1400	159	13.5	15.3	1013.4	24.6	23.0	23.4	23.26	35.6	0531	0335	0201
03/07/96	1300	168	13.9	16.0	1013.0	24.5	23.0	23.3	23.23	35.6	0241	0194	0051
03/07/96	1200	162	14.6	16.8	1012.4	24.4	23.0	23.2	23.17	35.6	0045	0112	0051
03/07/96	1100	161	12.1	14.4	1012.0	24.4	23.1	23.3	23.30	35.7	0020	0101	0048
03/07/96	1000	149	13.8	16.0	1011.4	24.4	23.2	23.4	23.39	35.7	0020	0101	0048
03/07/96	0900	144	14.8	16.8	1011.4	24.4	23.1	23.3	23.24	35.6	0020	0101	0048
03/07/96	0800	130	13.2	14.9	1011.9	24.2	23.2	23.3	23.35	35.7	0020	0101	0048
03/07/96	0700	132	12.3	13.6	1012.7	24.2	23.4	23.7	23.80	36.0	0020	0099	0048
03/07/96	0600	131	13.7	15.2	1013.5	24.1	23.4	23.7	23.73	35.9	0020	0100	0048
03/07/96	0500	130	10.6	11.9	1014.3	24.2	23.0	23.3	23.24	35.7	0020	0100	0048
03/07/96	0400	127	10.1	11.3	1014.6	24.1	22.6	22.9	23.07	35.2	0020	0099	0048
03/07/96	0300	123	10.0	10.7	1014.7	23.8	22.5	22.7	22.73	34.8	0020	0098	0048
03/07/96	0200	094	07.7	09.1	1014.7	23.1	22.5	22.7	22.73	34.8	0020	0097	0048
03/07/96	0100	080	07.6	08.1	1014.2	23.1	22.5	22.8	22.73	34.8	0020	0093	0048
03/07/96	0000	071	08.5	08.5	1013.7	23.4	22.7	22.9	22.87	34.8	0021	0098	0048
03/06/96	2300	072	09.3	09.5	1013.5	23.8	22.8	23.0	22.99	34.9	0100	0137	0067
03/06/96	2200	088	07.2	07.6	1013.5	24.1	22.9	23.1	23.03	34.9	0345	0277	0136
03/06/96	2100	113	07.9	08.4	1013.6	24.3	22.9	23.2	23.08	35.0	0588	0441	0229
03/06/96	2000	128	08.2	09.1	1014.0	24.3	22.7	23.1	22.94	35.0	0788	0597	0315
03/06/96	1900	144	11.7	13.5	1014.3	24.3	22.6	22.9	22.78	34.9	0928	0653	0353
03/06/96	1800	149	13.1	15.1	1015.0	24.3	22.3	22.7	22.57	34.9	0999	0680	0353
03/06/96	1700	153	11.7	13.2	1015.9	24.3	22.2	22.5	22.41	35.0	1001	0638	0333
03/06/96	1600	135	13.7	15.2	1016.1	24.4	21.9	22.3	22.14	35.1	0908	0556	0284
03/06/96	1500	134	13.9	15.6	1016.0	24.0	21.7	22.0	21.84	35.2	0748	0436	0211
03/06/96	1400	135	14.4	16.8	1016.1	24.0	21.5	21.7	21.66	35.3	0529	0310	0139
03/06/96	1300	120	13.0	14.6	1016.1	24.1	21.4	21.7	21.62	35.4	0255	0189	0085
03/06/96	1200	121	14.2	15.8	1015.6	23.7	21.4	21.7	21.64	35.4	0048	0113	0052
03/06/96	1100	121	14.2	15.9	1015.3	23.6	21.4	21.6	21.63	35.4	0020	0101	0048
03/06/96	1000	125	14.5	16.7	1015.2	23.6	21.4	21.6	21.62	35.4	0020	0101	0047

Appendix 8

Report for Molasses Reef

Coastal-Marine Automated Network (C-MAN)
~~ Molasses Reef (MLRF1) ~~

Date	GMT	WD	WS	WG	BAROM	AT	ST-I	ST-1	SAL1	PARs	PAR1
03/08/96	0900	096	09.6	10.2	1010.6	24.1	22.8	22.88	35.2	0019	0020
03/08/96	0800	067	06.7	06.8	1011.3	24.1	22.9	23.01	35.4	0019	0020
03/08/96	0700	062	06.2	06.5	1011.1	24.0	22.7	22.89	35.5	0019	0020
03/08/96	0600	050	05.0	05.1	1012.3	24.2	22.7	22.81	35.5	0019	0020
03/08/96	0500	049	04.9	05.1	1012.2	24.1	22.6	nil	nil	0019	0020
03/08/96	0400	064	06.4	06.7	1012.2	24.0	22.6	22.67	35.6	0019	0020
03/08/96	0300	029	02.9	03.0	1012.8	24.3	22.5	22.55	35.6	0019	0020
03/08/96	0200	031	03.1	03.1	1013.0	24.1	22.6	22.60	35.6	0019	0020
03/08/96	0100	047	04.7	05.0	1012.4	24.2	22.9	22.89	35.6	0019	0020
03/08/96	0000	038	03.8	04.1	1011.7	24.3	23.1	23.17	35.2	0019	0020
03/07/96	2300	056	05.6	05.7	1011.3	24.4	23.1	23.19	35.6	0058	0020
03/07/96	2200	119	11.9	12.7	1011.0	24.5	23.2	23.17	35.6	0261	0020
03/07/96	2100	125	12.5	13.3	1011.1	24.6	23.4	23.45	35.4	0471	0020
03/07/96	2000	115	11.5	12.6	1011.2	24.5	23.4	nil	nil	0661	0020
03/07/96	1900	117	11.7	12.6	1011.9	24.5	23.6	23.62	35.5	0791	0020
03/07/96	1800	119	11.9	12.5	1012.8	24.5	24.0	23.98	35.6	0861	0020
03/07/96	1700	146	14.6	15.8	1013.3	24.4	23.7	23.78	35.6	0865	0019
03/07/96	1600	128	12.8	14.4	1014.1	24.3	23.6	23.67	35.6	0656	0019
03/07/96	1500	137	13.7	15.0	1014.2	24.3	23.5	23.51	35.7	0679	0019
03/07/96	1400	130	13.0	14.0	1014.0	24.2	23.4	23.42	35.6	0500	0019
03/07/96	1300	126	12.6	14.4	1013.5	24.1	23.3	23.39	35.6	0270	0019
03/07/96	1200	114	11.4	12.6	1012.9	23.9	23.3	23.35	35.7	0054	0020
03/07/96	1100	095	09.5	10.8	1012.4	23.8	23.2	23.27	35.6	0019	0020
03/07/96	1000	101	10.1	11.7	1012.0	23.8	23.1	23.17	35.4	0019	0020
03/07/96	0900	119	11.9	12.8	1012.1	23.8	23.1	23.15	35.7	0019	0020
03/07/96	0800	120	12.0	13.0	1012.7	24.0	23.1	23.15	35.7	0019	0020
03/07/96	0700	118	11.8	13.0	1013.4	23.9	23.3	23.33	35.5	0019	0020
03/07/96	0600	102	10.2	12.0	1014.1	23.8	23.3	23.38	35.6	0019	0020
03/07/96	0500	095	09.5	10.3	1014.5	23.7	23.2	nil	nil	0019	0020
03/07/96	0400	075	07.5	07.9	1014.9	23.6	23.2	nil	nil	0019	0020
03/07/96	0300	068	06.8	07.4	1015.1	23.7	23.2	nil	nil	0019	0020
03/07/96	0200	075	07.5	08.2	1015.1	23.9	23.2	23.22	35.6	0019	0020
03/07/96	0100	100	10.0	11.0	1014.6	23.9	23.2	nil	nil	0019	0020
03/07/96	0000	078	07.8	08.5	1014.4	23.9	23.3	23.33	35.9	0020	0020
03/06/96	2300	072	07.2	07.5	1014.3	24.1	23.6	23.59	35.7	0060	0020
03/06/96	2200	046	04.6	04.9	1014.3	24.2	23.8	23.79	35.6	0262	0020
03/06/96	2100	054	05.4	05.7	1014.2	24.2	23.9	nil	nil	0482	0020
03/06/96	2000	060	06.0	06.5	1014.8	24.2	23.9	23.94	35.7	0632	0020
03/06/96	1900	094	09.4	10.5	1015.0	24.1	23.8	23.82	35.8	0800	0020
03/06/96	1800	112	11.2	12.2	1015.6	24.1	23.7	23.78	35.7	0874	0020
03/06/96	1700	113	11.3	12.1	1016.5	24.0	23.6	nil	nil	0846	0020
03/06/96	1600	116	11.6	12.7	1016.8	23.9	23.5	23.56	35.7	0804	0020
03/06/96	1500	134	13.4	14.7	1016.8	23.8	23.4	23.45	35.7	0683	0019
03/06/96	1400	127	12.7	14.1	1016.8	23.8	23.2	nil	nil	0020	0020
03/06/96	1300	143	14.3	15.4	1016.6	23.7	22.9	23.01	35.9	0029	0020
03/06/96	1200	135	13.5	14.4	1016.3	23.6	22.5	nil	nil	0020	0020
03/06/96	1100	145	14.5	15.9	1016.1	23.5	22.1	nil	nil	0019	0020
03/06/96	1000	146	14.6	16.2	1015.9	23.5	22.0	nil	nil	0019	0020

*U.S.GOVERNMENT PRINTING OFFICE 1996-0-790-301/40023