Object-Oriented Design of a Windows™ Based Automated Telemetry System

Lance P. L. Self
Computer Scientist
NAWC-Weapons Division
Pt. Mugu, CA 93042

ABSTRACT

This paper illustrates a Windows computer application program which uses the object-oriented paradigm as a basis. The objective of the application program is to control the setup of equipment involved in routing a telemetry signal. This design uses abstract classes as high level building blocks from which site specific classes are derived. It is the next generation to the software portion of a system described by Eugene L. Law[1]. The object-oriented design method, as presented by Grady Booch in his book Object-Oriented Analysis and Design with Applications[2], is the design tool.

KEY WORDS

Object-oriented, Abstraction, Encapsulate, Class

DESIGN OBJECTIVES

The logical design captures the essence of the NAWC-Weapons Division (Pt. Mugu) automated telemetry signal routing system. This design has three objectives: software portability, software adaptability, and user friendliness.

The first objective describes the desire to transport this application to another site. This approach abstracts general and common operations, and places them into appropriate high level classes. The highest level classes which contain these operations become site transparent.

The second objective is software adaptability. This program is designed in a modular fashion so component replacement can be accomplished with less trauma to the system. Using encapsulation and inheritance principles a

programmer can integrate a new piece of equipment without major disruption to other parts of the program.

Graphical user interface (GUI) methods improve user friendliness. Using the mouse to implement dialog box elements such as list box, combination box, check box, radio button, etc., eliminates or reduces the need for operators to type commands. An on-line Help system also improves user friendliness by allowing users to search for information while the application is running.

ABSTRACT VIEW

The high level view of the system (Figure 1) shows the core classes, their relationships with one another, and their responsibility within the system (see Appendix for notation definitions). As previously stated, this logical design is intended to capture the essence of the automated telemetry signal routing system. Certain system components such as file management, operator interface, and the help system have not been included.

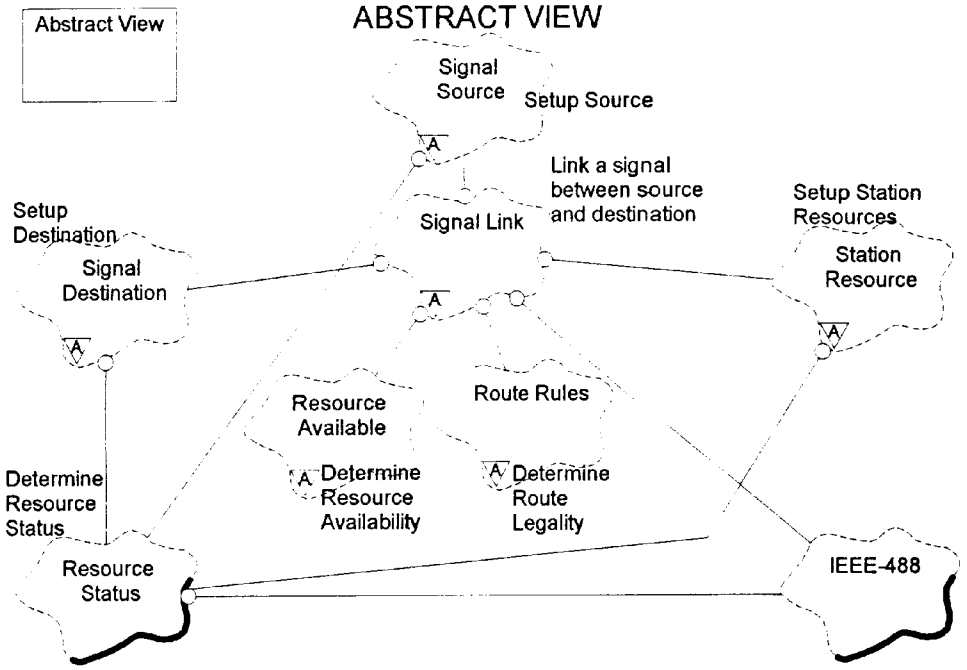

Figure 1

The cornerstone abstract classes of the system are Signal Destination, Signal Source, and Station Resource. These classes are designed to work with one another and are loosely coupled to common classes. All three use the services of Resource Status class, which determines the status of components. All three are

used by Signal Link, which links signals from source to destination. Both Resource Status and Signal Link use IEEE-488 to interrogate components.

Signal Link also uses Route Rules abstract class to determine if a component can legally be linked to another component. Components assume different roles within the system. A bit synchronizer, for example, is at times a source, and other times a destination. The Route Rules class is responsible for determining what role a component is assuming.

Resource Available is also used by Signal Link. It is responsible for determining which components have been assigned to an operation. Because simultaneous operations are conducted, Resource Available is used to decide which components are available and more important, prevent "de-assigning" a component while an operation is in progress.

SIGNAL SOURCE VIEW

Input to the system is received through antennas at Pt. Mugu. The Signal Source class (Figure 2) is responsible for setting up these signal source devices. Antenna class forms one aggregate of Signal Source and Global Source forms the other. Components which would fall under the Global Source class are San Nicolas Island antennas, fiber optics, best source selection, etc.. The reason for the class split is because the Global Source class is not under computer control. Typically, global sources are manually patched directly to some destination. They are included, however, to illustrate the modularity of the design.

The Antenna class itself is made up of general functions. These functions (Clear, Load, Prior, Save/Next) provide the operator with the ability to quickly setup, save, and review individual antenna frequency band assignments.

The single child class of Antenna is GKR-11. This class inherits properties from Antenna, yet retains the ability to over-ride functions when necessary. The design appears redundant since there is only one type of antenna (i.e., there could have been a single antenna class which would also have been GKR-11 class). The reasons for this design choice is first, to be consistent with other components, all of which have an abstract parent class, second, future expansion, and third, software portability.

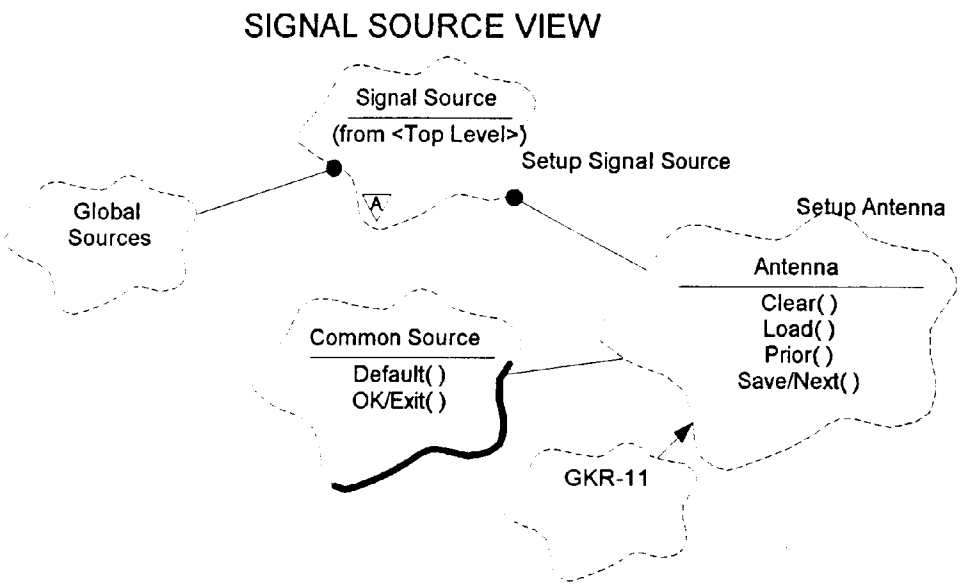

Figure 2

STATION RESOURCE VIEW

The Station Resource abstract class (Figure 3) is responsible for setting up station components. These components are: MR1200 receivers, PC3200 combiners, and two types of bit synchronizers (DSI7700, DSI7735). Each component class is associated with the utility Common Station, and each has at least one child class which inherits properties from it. Consistent with all component classes, these classes are designed to allow the operator to quickly and easily setup, save, and review component settings.

Each component uses a structure (not depicted) of component settings. Within the structure is the name of each parameter for that component. Using the structure, each component class has the ability to save, load, and review settings from previous operations.

Bit Sync has two child classes: DSI7700 and DSI7735. Both child classes inherit common functions from Bit Sync (Clear, Load, etc.). Each class over-rides functions according to it's needs. DSI7735, for example, has a "Equalizer" parameter which DSI7700 does not. DSI7735 over-rides common functions allowing use of this parameter at the lowest class level.

Although wired together, the Receiver and Combiner classes are not directly linked to one another. One test for a class is responsibility, i.e., what task(s)

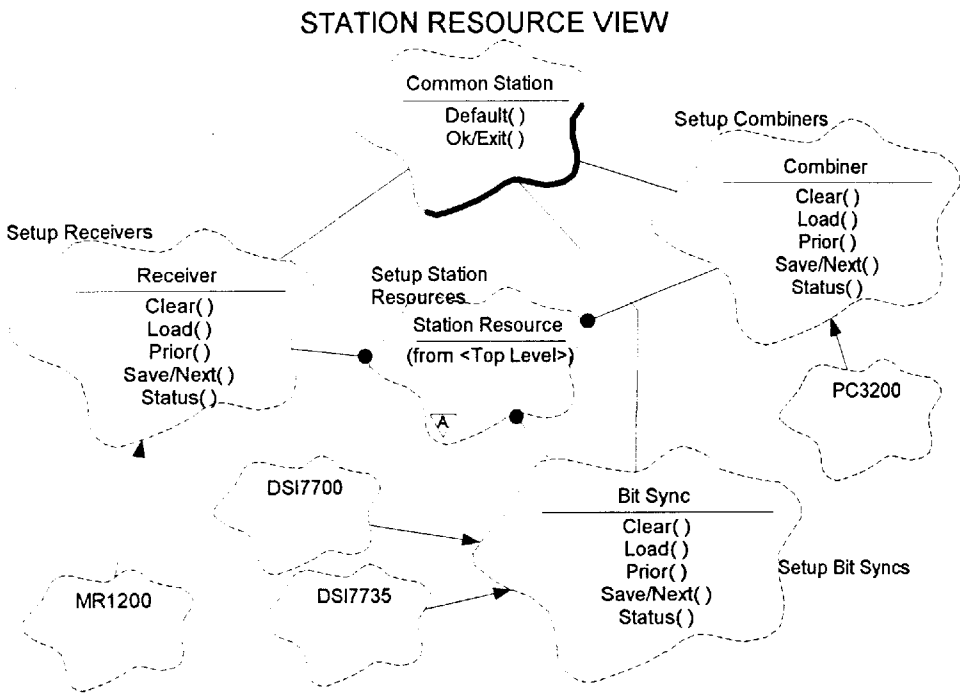

Figure 3

must this class accomplish. These classes clearly have different responsibilities within the system and so separate classes are warranted.

SIGNAL DESTINATION VIEW

The Signal Destination abstract class (Figure 4) is responsible for setting up the tape recorder. Currently there is one type of recorder, the Sabre 10. In keeping with the modular design, the Sabre 10 class is derived and inherits properties from the Tape Recorder class. When other tape recorder types become available they will form other classes.

SIGNAL LINK VIEW

Typically, a antenna signal is routed through receivers, combiners, and bit synchronizers. Along the data path, switches are used to deliver the signal to the destination(s). The Signal Link class (Figure 5) is responsible for providing the correct switch card address, slot, and channel with which to link those components. It works in conjunction with Route Rules and Resource Available to determine if a component is available, and if a connection is allowed.

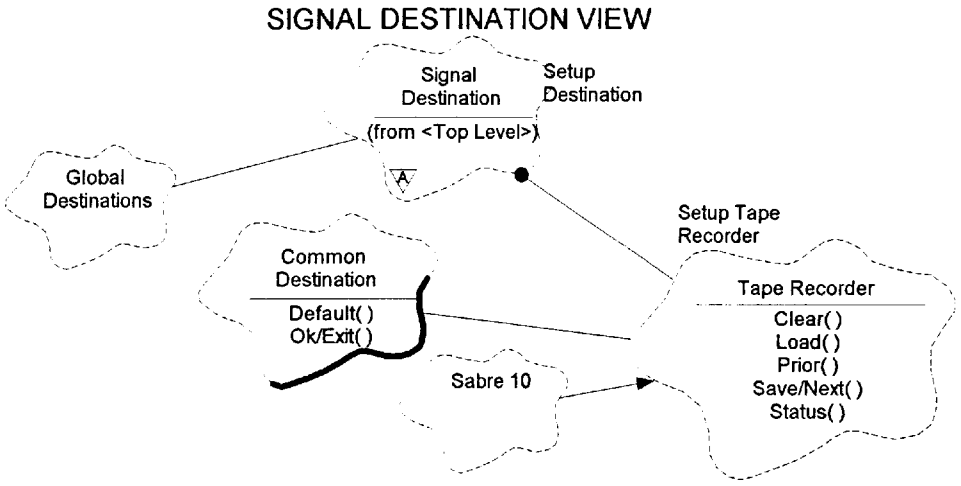

Figure 4

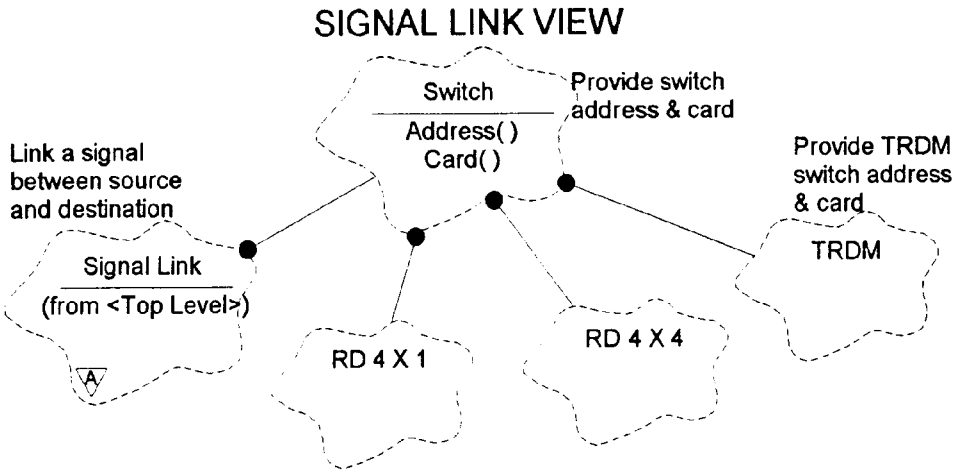

Figure 5

Two types of switches are represented by the RD 4 X 1 and RD 4 X 4 classes. TRDM is a switch matrix for distribution to the tape recorder.

All component classes are graphically represented by dialog boxes. The Signal Link class dialog box is presented in Figure 6. This dialog box allows users to connect components by using the mouse and "clicking" on them. This process draws a line between components to graphically show the connection. A connection is shown between the LoS band frequency of antenna 1 and receiver 4. Components that are off-line for any reason are shown as solid black. These components cannot be connected.

As shown in Figure 6, antenna 1 has two active frequency bands (the remaining 2 are covered). Within station 1: receivers 1, 4, and 5 have been setup (2, 3, and 6 are covered). Bit synchronizers (DSI7700) 1 and 4 have also been setup (2 and

3 are covered). Notice in station 2 the different colored icon for bit synchronizers 3 and 4. These are DSI7735 bit synchronizers. Finally, both tape recorders in station 1 have been setup.

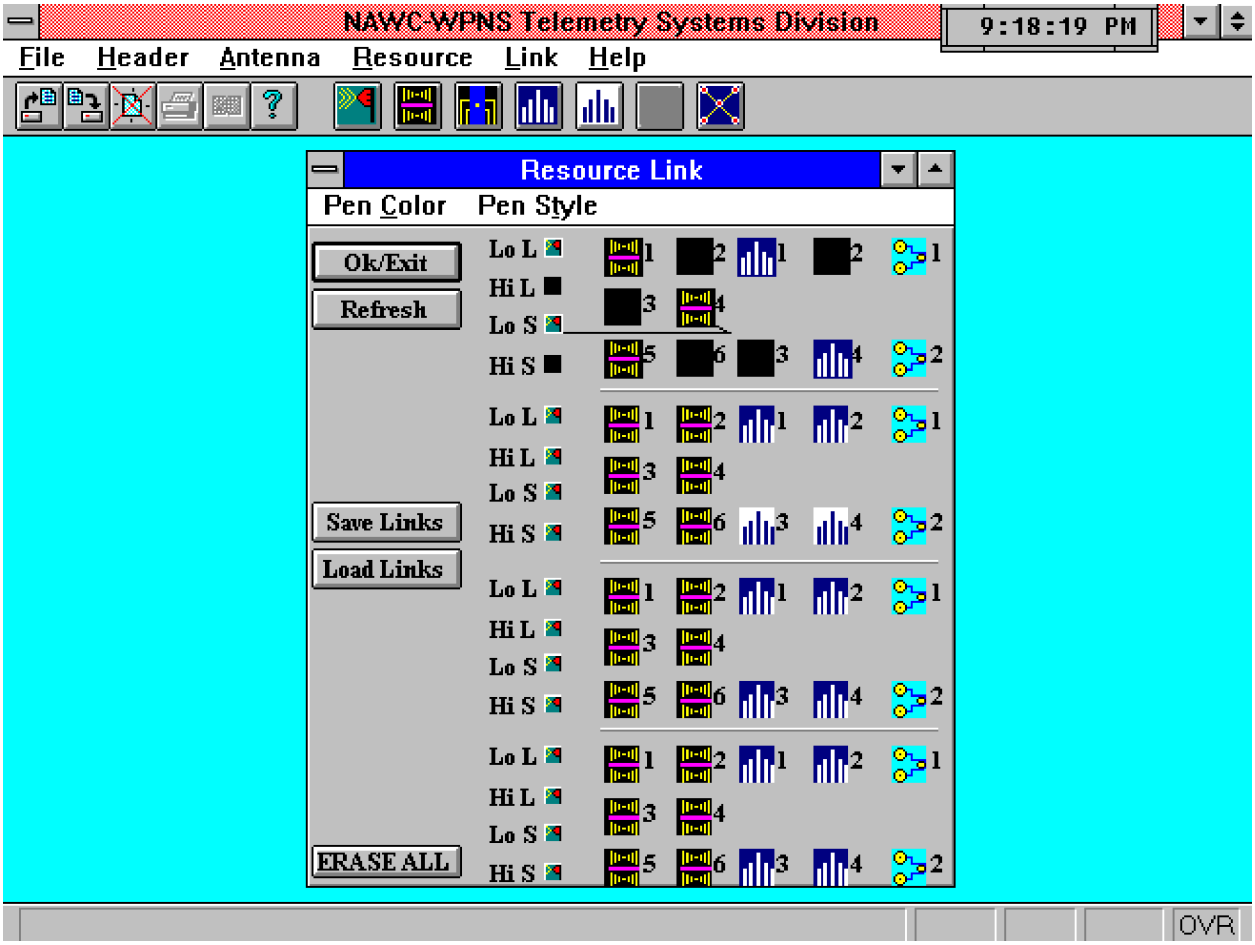

Figure 6

SUMMARY

Designing in abstraction is important for software portability and adaptability. Abstraction allows the system to filter the essentials of a class at a high level. The specifics of the class, those operations and attributes specific to the application, are implemented at a lower class level. Careful use of abstraction can produce high level generic classes that are transparent to a specific site. Low level classes may then be built to implement the site specifics. This design uses object-oriented principles to take advantage of abstraction.

I wish to thank William Karr, Melvin Kilpatrick, and Kenneth Smith for their efforts reviewing this paper. Their comments and suggestions are greatly appreciated.

REFERENCES

[1] Law, Eugene L. "Computer Controlled Telemetry Receive and Record System." Proceedings of the International Telemetering Conference, Vol.XXVI, Instrument Society of America, November 1990, pp. 655-662.

[2] Booch, Grady. Object-Oriented Analysis and Design with Applications. 2nd ed. New York: Benjamin/Cummings, 1994.

[3] White, Iseult. Using the Booch Method. New York: Benjamin/Cummings, 1994. p. 180.

[4] White, Iseult. Using the Booch Method. New York: Benjamin/ Cummings, 1994. pp. 179-180.

[5] White, Iseult. Using the Booch Method. New York: Benjamin/ Cummings, 1994. p. 181.

[6] White, Iseult. Using the Booch Method. New York: Benjamin/ Cummings, 1994. p. 178.

APPENDIX

This section briefly describes the Booch design notations used in figures 1-5.

- “cloud” icon - Abstract class. Used at all levels and serves as design building blocks.

- “shaded” icon - Utility class. Used to depict those classes that contain operations which provide additional functions for classes.

- A line with an open circle is a Uses relationship. It shows "that the source class (class with the circle end) depends on the target class to provide certain services[3]."

- A line with a solid circle is a Has (aggregate) relationship. It shows a "whole-part relationship between classes[4]."

- A line with an arrow is a Inherits relationship. It shows the "subclass shares the structure or behavior defined in one or more superclasses[5]."

- A simple line between classes is an Association relationship, which is described as a "semantic relationship between two classes[6]."