Real-Time Environment Modeling
Using Object Oriented Techniques

Carl Burmeister
January 23, 1993

Summary

Much has been made of recent advances in software development as regards software reusability as a means of making systems easier to develop and maintain and making (as the name implies) it possible to reuse significant portions in other related systems development, further decreasing development and maintenance efforts. Real-time systems pose both unusual problems and unique opportunities in this area. This paper will explore specific modeling approaches for tying together the real world we are controlling, the devices we are using to monitor and control that environment, the internal representation of the environment and temporal relationships between discrete events within the environment and related control functions.

Part of this study will be the application of the proposed techniques to an hypothetical air traffic control system; we will provide examples of traditional process control environments as well.

1.0 Introduction

Reusability comes in two different forms, component reuse (Object Orientation) and generational reuse (encapsulating knowledge of the problem domain in a program which generates a wide variety of similar applications). What this work attempts to examine is the application of component reuse techniques toward modeling real-time environments.

Reusability is often spoken of in terms of Object Oriented Design (OOD) and Object Oriented Programming (OOP). Object Orientation usually refers to the organization of software into discrete elements or "objects" which encapsulate both the data structure defining the element and behavior of such elements once created. Reusability and Object Oriented design involve the application of notions of classification, abstraction and encapsulation.

An object is said to be some distinct and accessible software element that can be found in the storage resources of a system. The external view or access of these elements are known as its specification while the objects internal representation is called its implementation.

Unfortunately, there has been a great deal of misconception about what sort of "objects" are the concern of the application developer. This is largely a result of the nature of the early practitioners of this technology. Object Orientation has been applied extensively to the areas of operating systems and graphical user interface managers. As a result, the "how to" literature is full of examples of such things as memory objects, mouse objects, window objects and process objects. For the application developer however these are, in fact, implementation details and are properly encapsulated in the classification hierarchy built around the real problem domain.

In order to start developing an object classification scheme, one needs to ask about what the highest level (most abstract) entities the application delivers and what the most general abstractions of its input stimuli are. From the standpoint of a process control application, the highest level abstractions might be data gathering entities and control entities. From the standpoint of an air-traffic control application, the highest level abstraction of the problem domain would be the volume of airspace under control and, of course, those objects and weather conditions which are present in that airspace.

Classification and refinement enable us to take advantage of large portions of our knowledge of the problem domain in future development efforts, while the encapsulation methodology enables us to easily incorporate activities which we anticipate will be provided by future, time constraint capable operating systems but which currently must be done at the application level. Object orientation also encourages the atomic break-
down of overall process execution which will make using currently available timing analysis tools more practical.

### 2.0 Real-Time Environment Modelling

The most important step in developing a useful model is understanding the so-called problem domain. At the highest level this means understanding what the common denominator of all the desired concrete results of the system might be. For example, in a process control environment where we are trying to control an oil field production system, our desired results are a maximized flow of crude together with minimized risk to life and property. From that we can conceive of our system as merely a black box that creates desired oil flow into a pipeline from one or more sources in the ground. The system provides control for oil flow and safety purposes.

Generally speaking we may look at any real-time system as being made up of

1. the real world environment we wish to control
2. the devices which acquire data about the real world and which provide the ability to control certain aspects of that environment
3. the internal representation or the real world environment as digital data in structures in both the permanent and temporary storage of the digital processing components of our system.

Each of these represent a separate but interrelated problem domains and object classification hierarchies. At each level of the object classifications in one hierarchy, there are associations with objects at the same level within each of the other two hierarchies. The data acquisition and control hierarchy provides the interface by which we get information about the real world which is then saved in the internal digital representation. The internal representation deduces the behavior of the real world environment based on that information and communicates actions to the data acquisition and control hierarchy in order to control the behavior of the real world environment. In the following diagram, we show in general terms the associations which exist at each level.

![General Relationships Between Object Hierarchies](image)

**FIGURE 1.** General Relationships Between Object Hierarchies

In the above figure, we show how each level of each hierarchy forms associations (represented by the thick, shaded lines) with corresponding levels in the other hierarchies. We will explore the advantages and disadvantages of establishing this problem domain breakdown. In terms its advantages, we count

1. completely application independent and reusable device access and control functions
2. clearer establishment of temporal relationships between the data processing, device and real world entities.
3. easier “firewalling” of dependencies.
4. clearer understanding and manipulation of the problem domain by keeping our entire real world model distinct from its internal representation.

Whereas among disadvantages
1. redundancy of data structures
2. do-nothing functions in the device control hierarchy which simply pass through data.

In our example prototype applications we will clearly show that the disadvantages are outweighed by the benefits in terms of clarity and reusability this approach affords.

3.0 Temporal Constraints

One of the principal roadblocks to open systems in real-time, meaning not only reusability but also portability and interoperability, is the issue of establishing a model of the systems timing behavior completely independent of the hardware and operating system being used for the data processing component. While research into the areas of schedulability and alternate scheduling methods offers promise, as a practical matter, little of that work can be counted on for systems under development today. What we will show is a sampling of methods for encapsulating required temporal behavior in the real world class hierarchies and for encapsulating the latencies in translating environmental data and in making control decisions. We will describe the firewalling of such timing regimens, where there are system and architectural dependencies, in lower levels of the device and internal representation hierarchies.

4.0 Conclusions

What we will show is how, through the use of Object Oriented techniques, we can model the real-time behavior of a system. The particular properties that make the system real-time i.e. that it has temporal constraints and even that it has interactions with a real world environment. That some part of the real world environment is part of any real-time system is part of the quality that causes us to think of that system as being “real-time”. It is important to the idea of making real-time systems portable and their components reusable that encourages us to consider developing object oriented modelling techniques.