JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 2, July 2005

Signal Diagram in Support of System Requirements Analysis

Marco W. Soijer*
Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629 HS Delft, the Netherlands

System development for airborne applications benefits from structural analysis as known from software engineering for describing the architecture of the system and analyzing the flow of information. Although the Unified Modeling Language is widely accepted as a standard for requirements analyses in information technology, its application to real-time signal processing design as encountered in avionics or flight test instrumentation design is often problematic. The cause is the absence of a diagram type that is tailored to signal processing instead of classical, operator-driven use cases. A new type of diagram, referred to as the signal diagram, is introduced as an alternative to use-case modeling. The signal diagram models the complete flow of information through the application, while focussing on the information itself rather than the components by which it is created or processed. The diagram combines existing symbology from electrical, electronic, and software engineering diagrams. It is intended as the very first diagram to be created upon system development. As such, it captures system requirements on the basis of functional requirements and serves as the starting point for deriving component requirements.

I. Introduction

There is a distinct difference between signal processing systems–or avionics and flight test instrumentation systems in particular–and real-time or distributed systems as they are normally dealt with in software engineering. Airborne signal processing applications are not dominated by contingent user input. Although an operator or a pilot in a closed-loop system might influence the way data are processed, the core of the system will generally process data in a consistent manner and will run quasi-continuously. The latter means that although processing in digital systems is time-discrete, there are no prolonged idle times that are only ended by user input. Requirements analysis for such a system therefore cannot be covered by the standard modeling of a concurrent system, in which the system reacts to discrete user inputs and reverts to an idle state when the event has been handled. However, this is exactly the behavior that is assumed for *use case modeling*, which is the starting point of system requirements analysis according to development methodologies like the *concurrent object modeling and architecture design method* (COMET) as presented by Gomaa,[1] or the object-oriented modeling and design methods by Rumbaugh et al.[2] or Jacobson et al.[3] Instead of use case modeling, requirements analysis for signal processing applications must be based on a flow-chart of signals and operators in the system. This paper proposes a new diagram type that combines existing symbology from electrical, electronic, and software engineering diagrams to create a flow chart that is suitable for requirements modeling. The diagram and the corresponding modeling activity were successfully applied to the development of a flight test instrumentation system for the Delft University Cessna Citation laboratory aircraft.[4]

Received 28 January 2004; revision received 12 March 2005; accepted for publication 25 May 2005. Copyright 2005 by Marco W. Soijer. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 1542-9423/04 $10.00 in correspondence with the CCC.
* Researcher; project manager Formation Keeping Management System with EADS Defence Electronics, Woerthstrasse 85, 89077 Ulm, Germany; Marco.Soijer@eads.com; AIAA Member.

A. Use Cases

In use case modeling, the system is treated as a black box. Its functional requirements are represented in terms of the tasks that the system will perform for an outside user. Such outside users are referred to as actors. Each use case describes a typical application of the system and the major actions that are performed in the process. During the use case, the system may interact with additional actors. All use cases are nevertheless initiated by a single actor, which is referred to as the primary actor. Thus, a user-initiated, event-driven character of the system's application is essential to use case modeling. As a second step in traditional use case modeling, each case description is detailed in terms of one or more scenarios that textually describe a particular development of the use case. For example, the use case for a client who wants to withdraw money using an automated teller machine, will have different scenarios in case the user enters the correct identification number for the bank card that is used, and in case an incorrect number is entered.

Clear-cut use cases and scenarios are rare in many essential parts of airborne signal processing applications. Use cases are applicable to the human-machine interface parts of a system, as the operating crew is the primary actor who drives the behavior of the system. In these event-based processes, the consecutive steps of a scenario are synchronized by interaction with the actor. In the real-time parts of the system that process streaming data, an actor can influence the way the data is processed, but neither does the actor initiate the use case, nor are the system's activities synchronized with the user. An example is the behavior of a digital flight control system for which the pilot's inputs are one of many received signals, or data acquisition and processing by distance measuring equipment (DME) or a hybrid navigation system for the single and simple use case that the pilot has switched it on.

B. Unified Modeling Language

The quasi-continuous application as a signal processor suggest signal modeling as a counterpart of use case modeling which meets the specific requirements of airborne real-time signal processing systems. Signal modeling can thus be seen as an alternative starting point for the system's development process. In software engineering, development methodologies have matured during the last decades from a series of individual approaches[1−3] to an industry standard which is more or less summarized by the Unified Software Development Process.[5] This development methodology relies on the Unified Modeling Language (UML)[6,7] for the recording of its analyses and designs in a series of diagrams. As signal modeling is an alternative to use case modeling, the flow-chart-like diagram that is introduced in this paper is regarded as an alternative to the UML's use case diagram. It is named the *signal diagram* and combines elements from the class, use case, and sequence diagrams as used in UML with elements from traditional signal flow diagrams as used in electrical and electronics engineering.

II. Signal Modeling

Like a use case model is meant to serve as a basis for static modeling of the system's components,[1] the annotated diagrams from signal modeling should be such, that the signal descriptions allow for a detailed task analysis of the system components that are responsible for transforming one signal into another. The signal diagram thus closes the gap between functional requirements analysis as performed before the system development process is started, and detailed component and/or software requirements analysis. The signal diagram can be a support in mapping the detailed requirements to the functional requirements as necessary during a formal development process according to DO-178B.[8]

A. Modeling Process

Signal modeling typically starts at the back end of the system, by defining the signals that are produced as output. In this context, the term signal has a broader scope than it typically has in signal processing. It covers not only electrical signals that are used to transmit information, but also any other type of time-dependent physical quantity that conveys information or excites an adjacent system. For example, actuator positions or forces and moments that act on a body are considered signals as well, although they do not transmit information in the traditional sense. The output signals for the system follow directly from the functional requirements of the application for which the system is developed. The requirements that are specified for the output signals in the signal model typically include the accuracy and any real-time or reliability requirements. The complete signal model is then constructed by analyzing the signal requirements from the system's back end towards the front. For each signal, the operator from which it originates is identified and indicated in the signal diagram. Each operator must have at least one outgoing signal; multiple output signals are possible. When an operator does not have an output signal, it is useless for the application and should be removed from the signal model. The operator and its output signals, in combination with the previously specified output requirements, determine the input signals to the operator and the corresponding input requirements.

This procedure ensures the completeness of the signal model with respect to the primary signals that contribute to the system's outputs. Additionally, it prevents the model from containing obsolete or redundant signal paths. The resulting diagram contains all internal signals that contribute to the system's output—hence to the fulfillment of the functional requirements—but nothing more than these. This is an aspect that is important for using the signal model in a system certification process.

B. The Signal Diagram

Where existing types of flow charts and block diagrams are mainly used during the design of an electrical or electronical system, the signal diagram is a product of the requirements analysis activity of a signal processing system. Consequently, the focus of the diagram is on the specification of signal requirements. For this purpose, the diagram is extensively annotated with many characteristics of the signals that are depicted. The components that make up the system are shown in the diagram as well, but merely as black boxes that transform one or more input signals into one or more output signals. The detailed characteristics of the operators is of no importance in the signal diagram. It is the objective of the analysis and design activities afterwards to derive the operator requirements from the input and output signal requirements.

Figure 1 shows the suggested elements for the signal diagram. They are divided into three groups: the signals themselves, the annotations to the signals that specify their requirements, and the operators that process the signals. The signal symbology covers the primary signal characteristics. There are differently labeled arcs for analog and for digital signals; additional arrows at the start point of an arc indicate continuous or discrete signals. Most signals in airborne equipment will be discrete and digital; they are depicted with an open triangle at the beginning of the arc and a square doublet somewhere along it. Continuous, analog signals are shown with a filled triangle and a sine doublet along the arc. The other combinations are possible as well—for example, the output of a sampler will be a discrete, analog signal, depicted with an open triangle and a sine doublet—but are less common. To avoid clutter in the signal diagram, it is suggested that the triangles and doublets for depicting continuous or discrete and analog or digital signals are only shown when the characteristics are changed by an operator. For large areas of a diagram where all signals are of the same type, they can thus be depicted as simple arrows from one operator to the next.

The actual signal type is described by its name. Similar to mathematical convention, regular characters indicate a scalar signal and bold characters indicate a vector signal. Scalars are the classical interpretation of a signal. However, because the signal diagram considers any coherent transmission of information from one operator to the next as a signal, the three-dimensional position of an object, or its spatial velocity are signals too. These are clear examples of signals that are of a vector type: they contain three components. More exotic signal types follow the UML convention for displaying objects: the underlined object name is followed by a colon and the class name. The class name can be used to refer to any user-defined type. If appropriate, it can be defined somewhere else in the signal diagram using the UML notation from the class diagram. For example, one might define the composite of a spatial position and a time tag as a four-dimensional "position" signal that is outputted by a time-tagging navigation device.

Signal names and types can be followed by any combination of requirement specifiers. An at sign (@) and a colon precede the frequency for a periodic, discrete signal and the accuracy of any type of signal respectively. For example, "temperature @ 5 Hz : 1 K" denotes a scalar, discrete signal of the name "temperature", which is updated periodically at 5 Hz and has an accuracy of 1 Kelvin. Angle brackets are only used with digital signals; they embrace the resolution or precision, the range, or both the range and the resolution. For example, "temperature : 5 K $<$ 200 K; 1 K ; 2000 K $>$" denotes a digital temperature signal that ranges from 200 to 2000 Kelvin and that has a resolution of 1 and an accuracy of 5 Kelvin. It is suggested that accuracies and precisions are specified by the size of the single-sided interval about the true value that contains 95% of the signal's disturbed values, which corresponds to twice the standard deviation for an assumed Gaussian distribution. The use of different definitions—for example 65% or 99% intervals—should explicitly be mentioned in the diagram by means of a UML note.

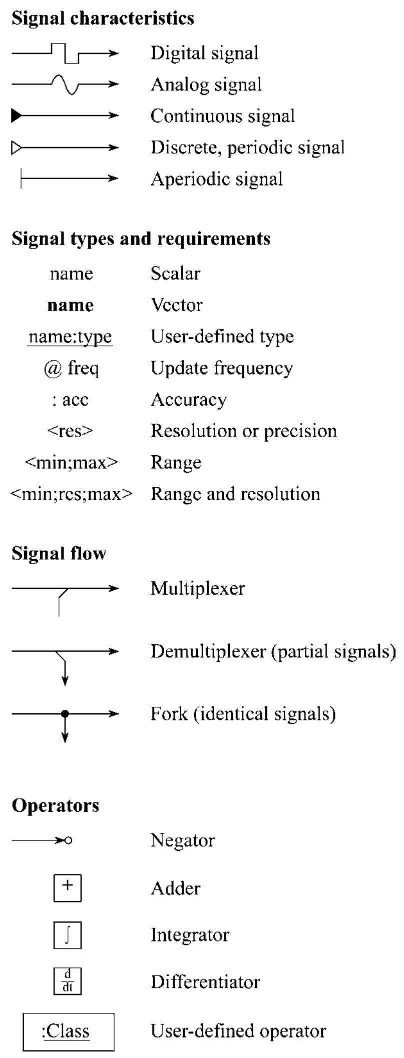

Fig. 1 Elements of a signal diagram.

The operators in a signal diagram are classes. They are depicted as usual in the UML class diagram. In addition to any user-defined operator, some common operators are predefined for the signal diagram. For an adder, an integrator, or a differentiator, the standard symbol as used in traditional flow chart diagrams can be used. From computer logic diagrams, the circle is copied as shorthand for a negator. It is used at the end of a signal, before connecting to another operator. In particular, it is often used in combination with an adder to indicate a subtraction.

C. Example Signal Diagram

Figure 2 shows an example of a small signal diagram for a simple hybrid navigation system, in which GPS position and velocity observations are fused with inertial measurements in a Kalman filter. At the top left, the vehicle is shown as the UML symbol for an actor, because it is external to the system and operates independently. The continuous, analog signal it produces is the vehicle's trajectory; the underlined name indicates that "trajectory" is neither a scalar nor a normal vector, but rather a compound type that combines position, orientation, velocity, and acceleration information.

The various operators at the top of the diagram convert the trajectory signal into several discrete and digital signals with varying characteristics. The GPS receiver transforms the vehicle's continuous and analog trajectory into

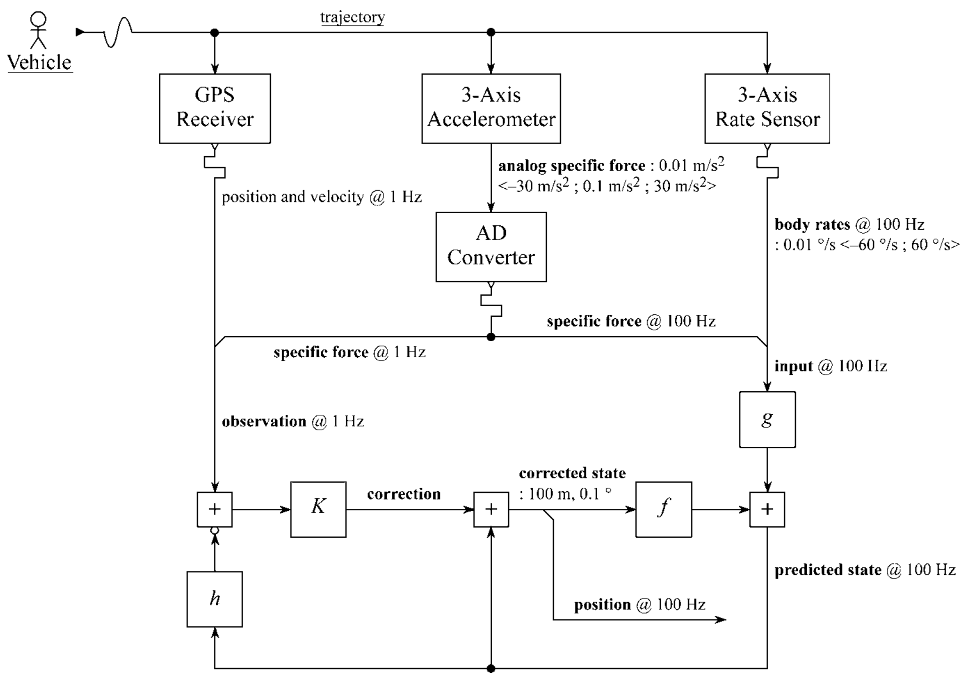

Fig. 2 Example signal diagram for a hybrid navigation system with Kalman filter.

a discrete and digital signal of position and velocity at a rate of 1 Hz. This is an illustration of the way of thinking during signal modeling. The vehicle trajectory is regarded as an input signal to the receiver, which is "transmitted" trough the strapdown installation of the receiver—or actually of its antenna—to the vehicle. In most other models of such an application, it is more likely that the radio signals from the GPS space segment would be shown as the only input signals to the receiver. Although these may be shown in the signal diagram of Fig. 2 as well, they are not sufficient. From a requirements-analysis point-of-view, the GPS receiver provides a digital measurement of the vehicle position and velocity, and thus requires the true, analog trajectory of the vehicle as its input.

Similar to the GPS receiver, the three-axis accelerometer and rate sensor convert the analog and continuous vehicle trajectory into electrical measurements of the specific force and the body rates respectively. The example shows an analog accelerometer that produces a continuous signal—known from the fact that the trajectory input signal is continuous and analog, and the characteristics are not changed—and a digital rate sensor that produces a discrete signal, as shown by the open triangle and the square doublet. The accelerometer signal is explicitly made digital and discrete by the analog/digital (AD) converter as shown.

The bottom part of the example shows a Kalman filter. It contains of a predictor and a corrector, informally described by the equations $\text{state}_{\text{predicted}} = f(\text{state}_{\text{corrected}}) + g(\text{input})$ and $\text{state}_{\text{corrected}} = \text{state}_{\text{predicted}} + K(\text{observation} - h(\text{state}_{\text{predicted}}))$ respectively. The specific force signal from the analog/digital converter is used for both prediction and correction. The filter's predictor uses the vector to integrate the vehicle's acceleration into a velocity and then into a position vector; the corrector uses the same vector to allign the vertical of the strapdown navigation system with the long-term gravity vector.

The diagram clearly shows the different bandwidth requirements for the two applications of the specific force vector: the predictor at the right-hand side requires a sample rate of 100 Hz, whereas the corrector at the left-hand side only requires one update per second. If the system should be changed in a way that the Kalman filter predictor no longer needs specific force information—for example because it directly acquires high-bandwidth velocity information from an alternative source —the signal diagram immediately reveals that the three-axis accelerometer and the corresponding analog/digital converter are still essential, but that these components now need to provide the specific force vector at a rate of only one Hertz. The example thus demonstrates the value of the signal diagram for the requirements modeling of a signal processing system that is to be changed, extended, or reused in a different context.

III. Conclusion

The signal diagram that is presented in this paper fills the gap between functional requirements modeling for an airborne signal processing application and the detailed requirements modeling of the system components. This is achieved by the process of signal modeling, which starts at the system's back end with the functional requirements on the outputs. Working from the back to the front, the process identifies the operators in the system without specifying them, and details the characteristics of all the signals in the system. The result is a model that can be used to derive the component requirements in the subsequent detailed analysis and design phases with three important advantages with respect to a procedure that does not use signal modeling. First, the focus on signals rather than operators allows a functional analysis through the system that is not obscured by system or component requirements. This way, the connection between the functional requirements for the application and the detailed requirements to the components (operators) is easily established. Second, the process ensures a clean system design without obsolete or redundant signals, which is important during system certification. Third, the signal diagram simplifies the analysis of the impact of a change to the functional requirements.

References

[1]Gomaa, H., *Designing Concurrent, Distributed, and Real-Time Applications with UML*, Addison Wesley, Boston, MA, 2000, pp. 110, 119–136.

[2]Rumbaugh, J., Blaha M.R., Lorensen, W., Eddy, F., and Premerlani, W., *Object-Oriented Modeling and Design*, Prentice Hall, Upper Saddle River, NJ, 1991, pp. 75–141.

[3]Jacobson, I., Christerson, M., Jonsson, P., and Overgaard, G., *Object-Oriented Software Engineering: A Use Case Driven Approach*, Addison-Wesley, Boston, MA, 1992, pp. 109–200.

[4]Soijer, M.W., *Software-Enabled Modular Instrumentation Systems*, PhD Thesis, Delft University of Technology, the Netherlands, 2003, pp. 159–186.

[5]Jacobson, I., Booch, G., and Rumbaugh, J., *The Unified Software Development Process*, Addison-Wesley, Boston, MA, 1999, pp. 33–58.

[6]Booch, G., Rumbaugh, J., and Jacobson, I., *The Unified Modeling Language User Guide*, Addison-Wesley, Boston, MA, 1999, pp. 105–116, 233–241, 275–340.

[7]Rumbaugh, J., Jacobson, I., and Booch, G., *The Unified Modeling Language Reference Manual*, Addison-Wesley, Boston, MA, 1999, pp. 63–88.

[8]RTCA, *Software Considerations in Airborne Systems and Equipment Certification*, DO-178B, RTCA Inc., Washington, DC, 1992.