

Object-oriented software development for computer-aided, mobile data transmission

Object-oriented software development is in part based on concepts already known for some time; it is however only in the past five years that it has become a topic of general interest. There are various definitions as to when an approach can be regarded as object-oriented. In contrast to the commonly employed function- or data-oriented approaches, object-oriented software development combines data and functions in such a way that they reflect as closely as possible the concrete and abstract objects of the real world.

Rohde & Schwarz has, for a customer-specific project, systematically and consistently implemented an object-oriented concept. The result is a flexible, easy-to-service **ship communication system** for the exchange of radar data, electronic mail and files of virtually any size [1]. The whole system is extremely immune to interference for two reasons:

1. use of high-quality Tx/Rx components such as VHF-UHF Transceiver XT452 and HF Transceiver XK855 with integrated FSK modem and ALIS processor [2],

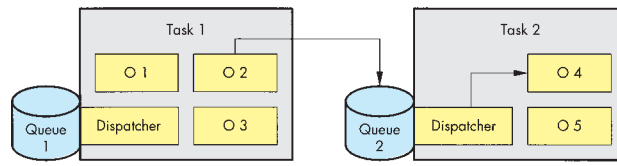


FIG 2 Typical inter-task communication via message dispatcher (O2 sends message to O4)

2. use of reliable system software that automatically responds to disturbance or interference from the environment not only by changing the frequency but also by effecting handover between the HF and VHF Tx/Rx components. The software design is based on **object-oriented software engineering** according to Jacobson [3], since this method enables a smooth transition from design to implementation.

Design steps

Object-oriented software development is implemented in several steps.

Step 1: Definition of functions

Based on the customer's requirements and the resulting specifications, all desired system functions – referred to as use cases – are recorded. The objective is to describe as completely as possible all functions required in a system. Fol-

lowing are four examples from the ship communication system:

“StartSupercycle” starts the super cycle, ie the central sequencing control for radio data transmission.

“StopSupercycle” stops the super cycle and allows radiotelephone communication.

“MakeLink” starts the procedure for a X.25 link setup.

“SendMessage” starts radio data transmission.

Step 2: Finding objects

In this step you determine what objects are necessary for system operation and what relationships exist between these objects. As a result the software architecture is obtained, ie the organization of the software into tasks and components (FIG 1). For example, the message dispatcher developed by Rohde & Schwarz plays an important role in the system because it extends the multitasking capability of the operating system (multi-user DOS) by simulated multitasking at the application level. While the major tasks are handled by the multitasking function of the operating system, the message dispatcher manages the objects within the tasks. For this the objects must register with all their methods (messages) at the message dispatcher during runtime. This procedure is employed not only for inter- but also for intra-task communication and for communication with device drivers (FIG 2).

Step 3: Description of interaction between objects

Object-oriented programs are based on the exchange of messages between ob-

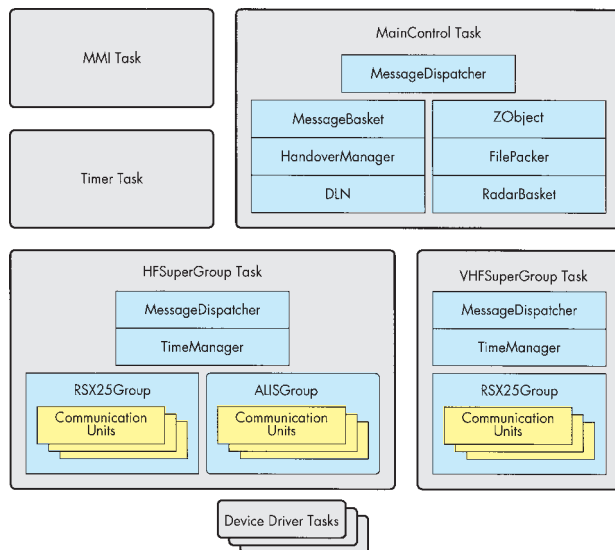


FIG 1 Software architecture for communication system

jects. A commonly employed method of illustrating the message flow is by means of message sequence charts (interaction diagrams). The message sequence charts enable the functional interfaces of the objects to be defined (FIG 3).

Step 4: Specification of objects

The majority of the objects used in the ship communication system were implemented in the form of status automaton as is common practice in communications engineering. State transition diagrams are used for the specification of the objects. The diagrams allow a smooth transition from design to implementation (FIG 4).

Coding and quality

About 20 new objects (classes) were defined for the above communication system. Part of the objects was later implemented from C++ using inheritance. Most of the objects can be re-used, eg the message dispatcher, as well as classes employed for system configuration and a timer class for time monitoring, to name just the most important ones. But not only have new classes been generated, old ones were used as well. The latter mainly included container classes and classes for the configuration of the user interface and were taken from the Borland library of classes.

High-quality software is characterized, among other features, by stability, maintainability and testability. Rohde & Schwarz created high quality in its ship communication system by following a strategy based on three mainstays:

Precondition and postcondition checks

Each object ensures as far as possible that its data are consistent by checking them during runtime. Checking is performed twice, ie on entering a method (precondition check) and on leaving a method (postcondition check). The concept is simple [4] but it markedly enhances quality and productivity as it requires developers to specify classes accurately to be able to define the conditions (invariants) for these checks. This makes codes maintainable and also extends component testing, which substantially reduces the probability of runtime errors.

Exception handling

Exception handling causes the program to go to a recovery routine in the event of an error. This feature of C++ is another powerful tool for the determination of runtime errors. It is used for cushioning protocol errors, for example.

Trace mechanism

Each object of a task writes messages into a file during runtime. The messages are classified according to contents and can be filtered as required. System operation is recorded in this way, and any malfunctions are traceable. This feature can be disabled for reasons of system performance.

Through the use of object-oriented techniques in conjunction with measures aimed at the improvement of software quality, Rohde & Schwarz has developed a product that meets even the most exacting requirements. Not with-

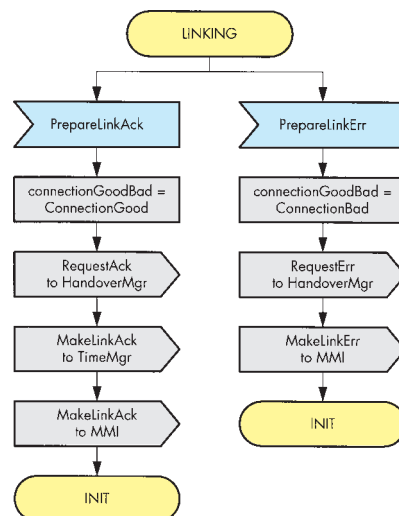


FIG 4 Detail of state transition diagram (object: RSX25Group; state transition: linking → init)

out reason is the object-oriented approach regarded as a quantum leap in terms of productivity and maintainability in professional software development. Exception handling as well as pre- and postcondition checks have been incorporated into modern programming languages [5] and make for considerably increased reliability, stability and testability.

Jürgen Rauch; Karl Scalet

REFERENCES

- [1] Jungherz, M.; Maurer, P.: Radiocommunication system helps Italian customs authorities in coastal monitoring. News from Rohde & Schwarz (1996) No. 150, pp 46–47
- [2] Greiner, G.: Reliable shortwave with ALIS. News from Rohde & Schwarz (1988) No. 116, pp 47–50
- [3] Jacobson, I.: Object-Oriented Software Engineering – A Use Case Driven Approach. Addison-Wesley (1993)
- [4] Porat, S.; Fertig, P.: Class Assertions in C++. Journal of Object-Oriented Programming. SIGS Publications (May 1995) volume 8, No. 2, pp 30–37
- [5] Meyer, B.: Eiffel: The Language. Prentice Hall (1992)

Reader service card 151/17

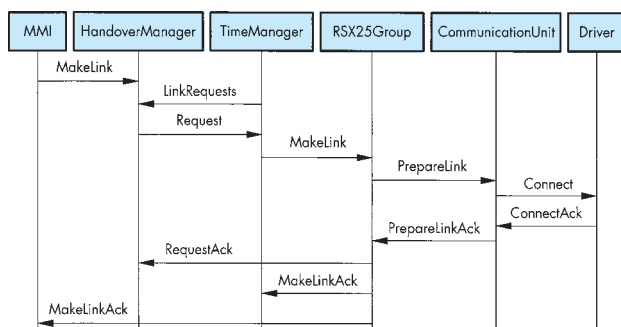


FIG 3 Message sequence chart for MakeLink system function