Failure Notifications: A Useful Extension to the Object Programming Model

Vicraj Thomas
Honeywell Technology Center
Minneapolis, MN 55418.
thomas@htc.honeywell.com

Abstract
This position paper argues for extending the object programming model with failure notifications that are generated when an object fails to meet its specification.

1. Introduction
Object-oriented system design and development methods have proven useful in building large and complex systems. Since real-time, dependable systems tend to be large and complex systems it is natural to use object-oriented methods for building such systems. Unfortunately, the object-oriented programming paradigm and the languages that support it were not designed with real-time or fault-tolerant system in mind.

It is therefore important to look at ways in which the object model can be extended to simplify the construction of real-time, fault-tolerant systems. This paper proposes extending the object model to include failure notifications. A case is made for the need for such failure notifications and the advantages of having such notifications.

2. Failure Notifications
Designers of real-time dependable systems have to make assumptions about the number and kinds of failures the system can suffer. These assumptions determine the techniques used to build the system. For example, if a component can suffer at most $N$ failures, a fault-tolerance technique such as the replicated state machine approach may be used with $N + 1$ replicas of the component. This assumption may however not hold during system operation, resulting in the failure of the component. If other components in the system are not prepared to deal with this failure, the entire system is liable to fail.

A failure notification is generated if a component or object fails, i.e., it is unable to meet it specification because assumptions made during system design no longer hold. For example, the system may have suffered unanticipated failures or may be overloaded due to excessive demands for service. Other objects in the system may ask to be notified if an object fails and may take corrective action on receipt of the notification.

3. Usefulness of Failure Notifications
The ability to detect the failure of a object can greatly simplify the construction of a real-time dependable system. Ideally, a real-time dependable system as a whole behaves as an object that executes commands correctly and in a timely fashion, or a failure-notification is generated. This assures users that, unless a failure notification is generated, their commands have been correctly processed and any results produced can be relied upon. Such a system is much easier to design and implement if each of its component objects in turn produce correct results or a failure notification is generated. Since the failure of any object is detectable, other object do not have to implement complicated failure detection schemes. These objects may in turn be implemented by other such objects, and this process continued until the simplest objects of the system are reached. At each level, the guarantees made by these objects simplify their composition to form more complex objects.

Failure notifications is even more important in systems built out of commercial-off-the-shelf (COTS) components since any assumptions regarding the run time behavior of the system are more easily violated.

A programming model that provides for such failure notification is described in [1]. This model was originally developed for building fault tolerant software and is currently being extended to handle timing failures.

References