A View of the GDMO Interface Definition Language

Robert A. Fraley

Hewlett-Packard
Networked Systems Architecture
fraley@nsa.hp.com

Abstract

Large distributed systems require increased facilities for distributed management. Common interfaces for accessing remote resources are critical for such facilities. GDMO is a management interface definition language that is an international standard for defining management interfaces.

Distributed systems management allows an administrator to run an application that can carry out management tasks in a distributed environment. To achieve this goal, management applications access various system components through management interfaces. A managed resource is any system component accessible through a management interface.

Systems management in a heterogeneous environment requires that system components developed by many companies support common interfaces. Creating these common interfaces is a major hurdle that must be overcome by the systems management community. Adoption of a common interface definition language is a major step towards achieving this goal.

Developers must be encouraged to produce manageable components and management applications. This will be economically viable only if an implementation can be leveraged over many machines, operating systems, and communications protocols. Emerging technologies for management should take this need into account.

OSI has been developing standards for the network management area. GDMO [1] is the interface definition language defined by this effort. It is used to specify management interfaces supported by resources and available for management applications. Specifications contain elements that include attributes and actions. The type structure of these elements is written in ASN.1 [2].

GDMO is object oriented. Interfaces are defined as classes, which may be inherited by other class definitions. Multiple inheritance is supported, and is useful for constructing resources that can be managed under several interfaces. GDMO elements are organized into "packages"; inclusion of a package is equivalent to inheriting a class. A package may be conditionally contained in a class, allowing the definition of optional extended features.

Notifications are another type of element that may be defined in a GDMO class. Notifications are asynchronous events raised by a resource and received by interested applications. They contain data describing the event. Applications receive the event information, and may reply to the event with response information. Events are important for management applications, and their inclusion in the class structure is important. Event replies are not currently in widespread use, but will have increasing importance.

Many management applications need to perform a single operation over many resources. The multi-cast facility of GDMO is called Scoping and Filtering [3]. A scope specifies some subset of resources that support a specified interface. A filter is a condition that must be met by the resource to process the message. For example, a message could be sent to all mail servers in the Chicago offices of a company, with filtering restricting the request to servers having more than 10 mega-bytes of accumulated mail. The GDMO model has limitations, but is a step in the right direction.

GDMO specifies the logical content of client-server messages, but actual communication requires a message protocol. OSI has defined the CMIP protocol for management communications [4]. Other protocols, such as RPC, could be used, and may be preferable for some platforms, such as OSF's DCE [5].

Each element in a GDMO class is assigned a unique identifier. This is used in the CMIP protocol to identify data elements. Unfortunately, this makes GDMO specifications protocol dependent. Interface definition languages should be independent of protocol, to help developers leverage their software into different environments. Without an agreement on a single standard for universal identifiers, this will be difficult to achieve.

The management community would like specifications to be reused whenever possible. Inheritance and package
inclusion lead to reuse, but a desire to reuse individual ele-
ments and error specifications has influenced the structure
of the language. Current cataloging practices do not distin-
guish elements that have semantic dependencies on other
elements of a class, so identifying reuse candidates is diffi-
cult.

As distributed systems become established, standard
management interfaces will be installed on more and more
system resources. This will lead to a problem for system
evolution: how to allow changes to interface standards.
Existing resources and applications cannot all support the
new interface at a single instant, nor can the network be
brought down and all components replaced simultaneously.
There must be an evolution, over several years, to new
interface standards. During this time the old interface must
be equally supported.

GDMO makes an initial cut at this problem with a con-
cept called "allomorphism". This allows applications to
present a list of acceptable interfaces when connecting to a
resource. Likewise, resources present a set of interfaces
through which they may be accessed. Resource access will
use the new standard when it is available, but can still use
an old interface when necessary.

Allomorphism is not yet a mature concept. We don’t
know yet how difficult it will be to support multiple man-
agement interfaces, or what interface design rules will
make the evolution simpler. Evolution will be expensive
for software developers. Interface designers must take
steps to ensure that changes beyond a simple extension of
the interface will seldom occur.

GDMO itself does not specify a mapping from the inter-
face specification to source code structures. XMP [6] is a
standard API that may be used with GDMO, providing the
procedures used to access objects through a GDMO inter-
face. XMP is defined in conjunction with XOM [7], which
specifies the mapping from ASN.1 data types to C lan-
guage specifications. Only if all of these standards are fol-
lowed will there be portability of software. Even with these
interfaces, applications cannot be totally independent of
the communications protocol.

The success of GDMO hinges on its acceptance in the
developer community. It faces competition from other con-
tenders, such as OMG’s IDL. Additional development of
the standard to streamline development using GDMO is an
essential ingredient for gaining acceptance.

References

[1] ISO/IEC 10165-4 Information Technology - Open Sys-
tems Interconnection - Structure of Management Infor-
mation - Part 4: Guidelines for the Definition of Managed

[2] ISO 8824 Information processing systems - Open Sys-
tems Interconnection - Specification of Abstract Syntax
Notation One (ASN.1), 1989

[3] ISO/IEC 10165-1 Information Technology - Open Sys-
tems Interconnection - Structure of Management Infor-

Interconnection - Common Management Information

[5] Introduction to OSP™ DCE, Open Software Foundation,

[6] X/Open Preliminary Specification, Systems Manage-
ment: Management Protocols API (XMP), XO/Prelim/91/

[7] X/Open CAE Specification, OSI-Abstract-Data Manipu-
lation API (XOM), XO/CAE/91/080, X/Open Company

89