MMN Architecture

Nat Pryce

1.0 Terminology

There has been some confusion due to conflicting uses of terminology. In particular, the word "binding" was used differently by Imperial and Lancaster. We decided to standardise on the following meanings:

Binding. An association between a client and a service created by the acquisition by the client of a reference to the service.

Communication Session. The state required for communication to take place. This includes state in the network, such as ATM VC's, and state at each end of the connection, such as protocol stacks.

2.0 Architecture

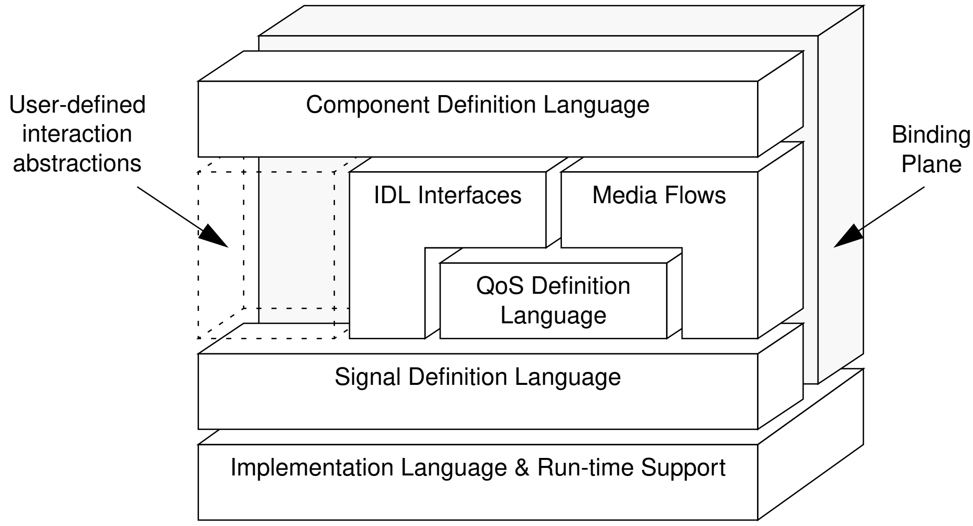
CORBA IDL is used to specify object interfaces in terms of operations, attributes, one-way messages and user-defined data types. An important issue discussed was whether IDL interfaces should also include media flows or whether "vanilla" IDL interfaces could be defined through which media flows could be set-up and controlled. That is, should media flows be in the ORB or external to the ORB.

Lancaster wanted to extend the IDL with media flows and signals. Signals are typed asynchronous messages, primitive enough to describe any other communication abstraction. Imperial wanted to leave the IDL as it is, since persuading the OMG to change the IDL would be politically impossible. Their suggestion was to define IDL interfaces which would be used to set-up and control streams from within a CORBA environment. However, Lancaster pointed out that since IDL has no notion of real-time, QoS-controlled invocations, it would be impossible to control a real-time media flow using IDL interfaces.

We decided to leave the IDL as it is. However, since IDL and media flows can be defined in terms of signals, we would create a "signal definition language" (SDL) which could be used to describe communication abstractions. IDL would therefore be defined in terms of the SDL and could be compiled into it. The SDL would also be used to describe media flows and would allow programmers to define new communication abstractions if necessary.

Signals would be used to monitor and maintain the QoS of a communication session; QoS requirements would be defined in some "QoS definition language" (QDL). QoS requirements for IDL interfaces and flows would be specified by associating QDL specifications with interfaces and bindings.

Layered above the communication abstractions is the concept of a *component*. Components are compound objects which export a number of interfaces. We agreed that a "component definition language" should be used to describe components and the bindings between them, making use of the IDL, QDL, SDL and other languages layered beneath it.



3.0 QoS Definition Language

We decided that it was best to specify QoS using a combination of temporal logic and name-value pairs. Sets of name-value pairs could be used to specify most QoS requirements but would not be expressive enough to express everything. Certain specifications would require the use of temporal logic, such as the QL or SQTL languages developed at Lancaster. Specifications written in such a temporal logic could be compiled into protocol layers which could themselves be "fine-tuned" by passing them sets of name-value pairs. Alternatively, a temporal logic statement might be obtained from a service provider, treated as an opaque value of a name-value pair, and passed to an interpreter in a QoS management agent. The real-time performance of such an approach might not be adequate, however.

QoS would be requested by associating QDL specifications with particular IDL interfaces and with bindings, either first party bindings or bindings specified in the CDL.