



C++ provides through abstract base classes and virtual functions the means for programs to call interfaces without knowing the class that is implementing that interface. Similarly, at a binary, rather than a source level, Symbian OS has used extensively polymorphic DLLs, which implements a specified interface, and so allows new functionality to be added to existing programming frameworks. Polymorphic DLLs, however, required each framework that uses them to provide its own mechanisms for clients to discover and instantiate the available implementations. **ECom** removes this duplication of functionality by introducing a generic framework that provides single mechanisms to:

- register and discover interface implementations
- select an appropriate implementation to use
- provide version control for plug-ins

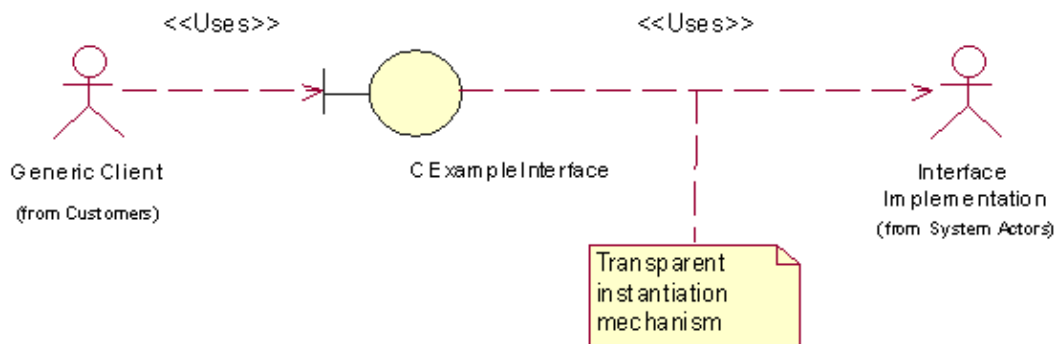
To show how **ECom** does this, let's recap the essentials of any plug-in system.

A client wishes to access an object to perform some processing. The specifics of this object are not known until run-time. The general characteristics of the processing are known, and are defined in an interface, but several variants of required processing could exist, which are provided by implementations that support the interface.

There are four clearly-defined roles in such a system.

- the client that wishes to access services
- the interface that defines how to request services
- the interface implementation that provides the required processing
- the framework that provides clients with access to the implementations

The relationships are shown in a UML diagram below:



The instantiation mechanism forms the backbone of such a system, and is responsible for providing the services that identify, and load the correct interface implementation at run-time. **ECom** is such a framework.

The full guide to ECOM can be found in Symbian's Developer Library documentation [here](#)