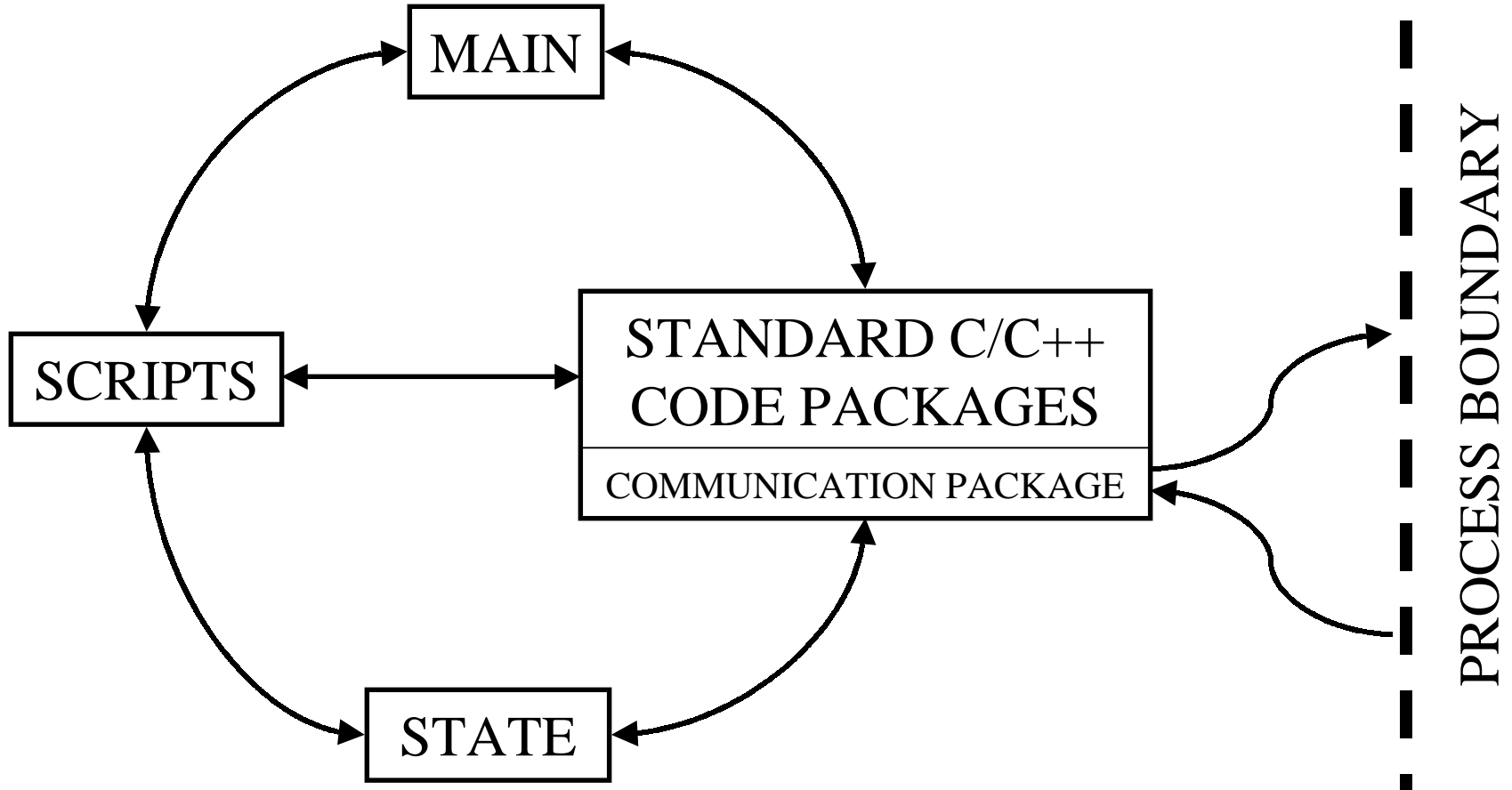


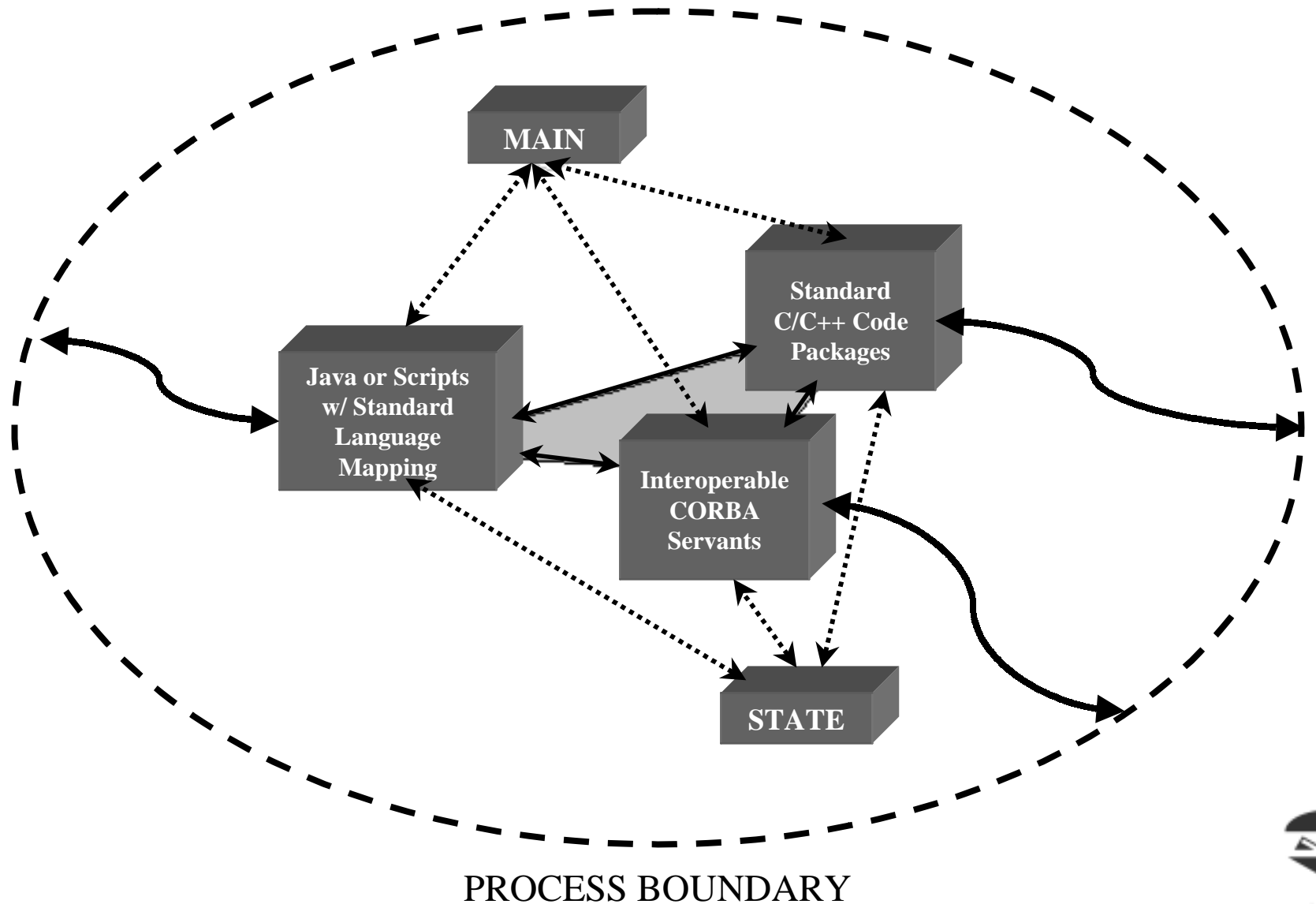
# Game Industry Characteristics

<b>LANGUAGES</b>	<b>CODE PACKAGES</b>	<b>DATA</b>	<b>DEVELOPMENT TEAM</b>	<b>USERS</b>
<b>C/C++</b>	<b>Rendering</b>	<b>3D Models</b>	<b>Game Concept Designers</b>	<b>Anyone</b>
<b>Java</b>	<b>Physics</b>	<b>Surface Bitmaps</b>	<b>Software Engineers</b>	
<b>Python</b>	<b>AI</b>	<b>AI Scripts</b>	<b>Artists &amp; Modellers</b>	
<b>Proprietary</b>	<b>Database</b>	<b>Player Statistics</b>		
	<b>Controller Input</b>	<b>Animation Sequences</b>		
	<b>Sound</b>	<b>Sound Effects</b>		

# Typical Collaboration for Client Architecture

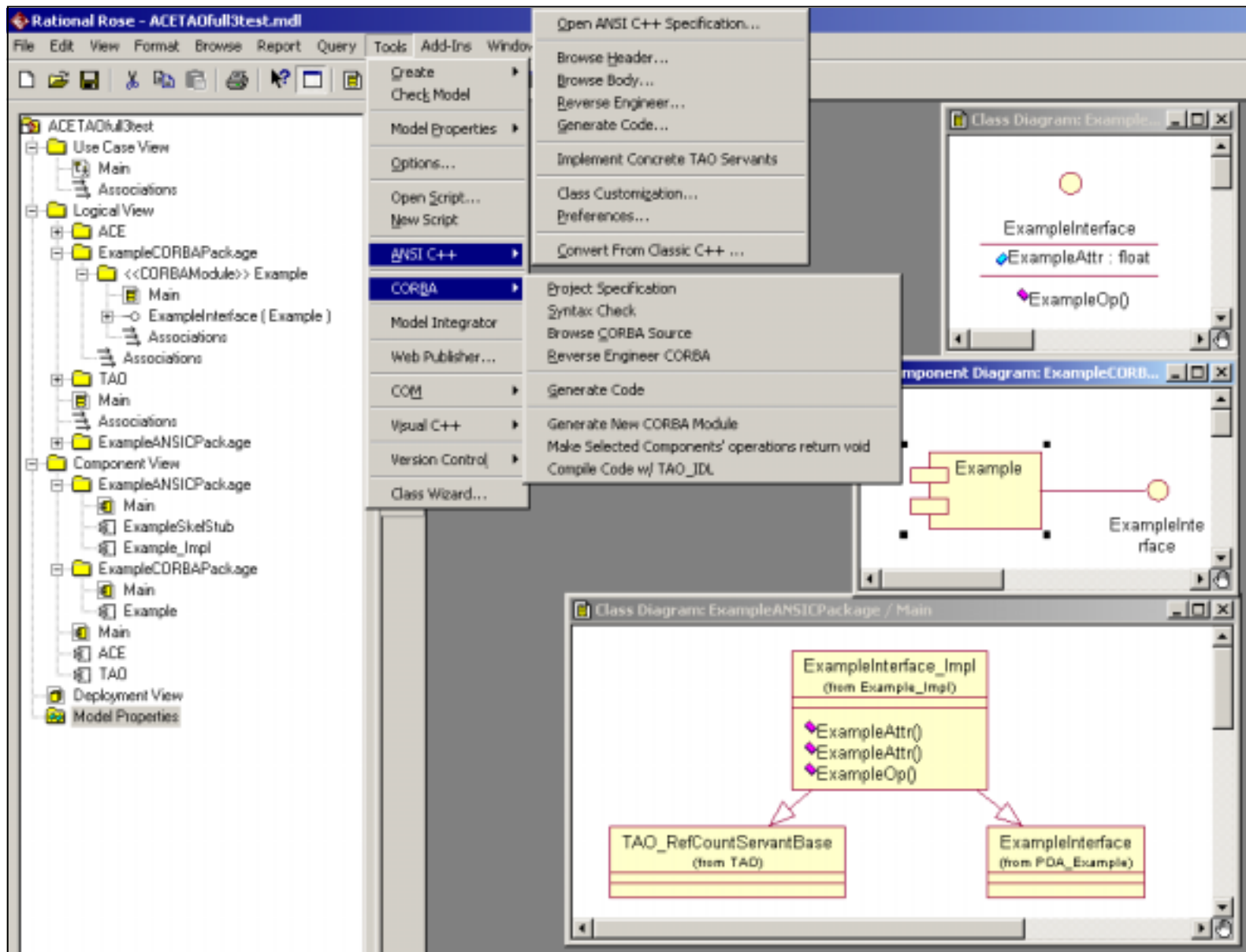


# Alternative Collaboration Using CORBA Architecture



# Developer Prototyping Process

1. Start by running multiplayer sample code.
2. Examine UML and extend samples.
3. Generate code for customized samples.
4. Examine critical metrics with diagnostic tools.
5. Round-trip engineering iterations refine code.



The Ninja Networking Developer Acceleration Kit streamlines the process of designing CORBA servants. Automated tools support roundtrip engineering, taking even non-CORBA developers from modelling the interfaces to generating the servant code.

# Automatically Generated Implementation of the Object Interface

## HEADER

```
#ifndef
EXAMPLEINTERFACE_IMPL_H_HEADER_INCLUDED_C2FAFEB7
#define
EXAMPLEINTERFACE_IMPL_H_HEADER_INCLUDED_C2FAFEB7

#include "ExampleS.h"
#include "tao/corbafwd.h"
#include "tao/PortableServer/Servant_Base.h"

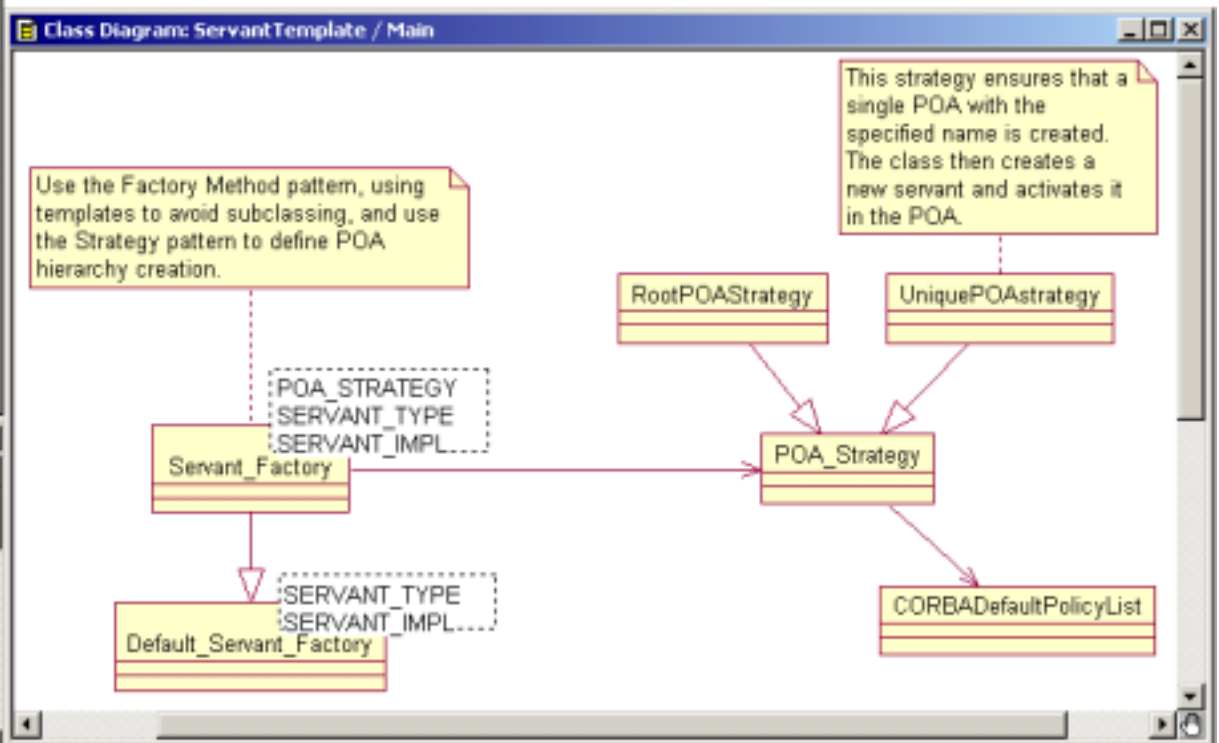
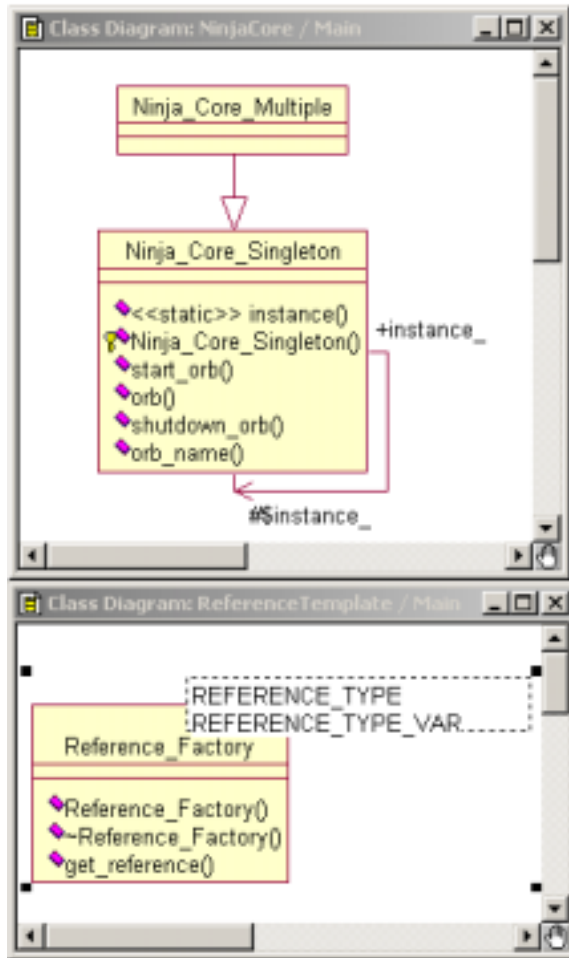
///
```

## SOURCE

```
#include "ExampleInterface_Impl.h"

///
```

# CORBA Object Factories Simplify Development



# Developer Servant and Client Main Blocks

## SERVANT

```
#include "ace\ace.h"
#include "Default_Servant_Factory.h"
#include "Ninja_Core_Singleton.h"
#include "tao\corbafwd.h"
#include "ExampleInterface_Impl.h"
#include "ExampleS.h"

INSTANTIATE_DEFAULT_SERVANT_FACTORY(ExampleInterface,
    ExampleInterface_Impl);

Default_Servant_Factory<ExampleInterface, ExampleInterface_Impl>
    example_factory;

int
main (int argc, char* argv[])
{
    ACE::init(); //Should be the first statement executed in this process.

    Ninja_Core_Singleton::instance()->start_orb(__argc, __argv);

    {
        ExampleInterface_var example_servant =
            example_factory.create_servant();

        //Leave the servant active until no longer needed
        some_event.wait();

    } //ExampleInterface_var example_servant goes out of scope.

    Ninja_Core_Singleton::instance()->shutdown_orb(TRUE);

    ACE::fini();
    return 0;
}
```

## CLIENT

```
#include "ace\ace.h"
#include "Reference_Factory.h"
#include "Ninja_Core_Singleton.h"
#include "tao\corbafwd.h"
#include "ExampleC.h"

INSTANTIATE_REFERENCE_FACTORY(ExampleInterface,
    ExampleInterface_var);

int main (int argc, char* argv[])
{
    ACE::init(); //Should be the first statement executed in this process.

    Ninja_Core_Singleton::instance()->start_orb(__argc, __argv);

    Reference_Factory<ExampleInterface, ExampleInterface_var>
        example_factory(__argc,
            __argv,
            Ninja_Core_Singleton::instance()->orb_name());

    {
        ExampleInterface_var example =
            example_factory.get_reference();

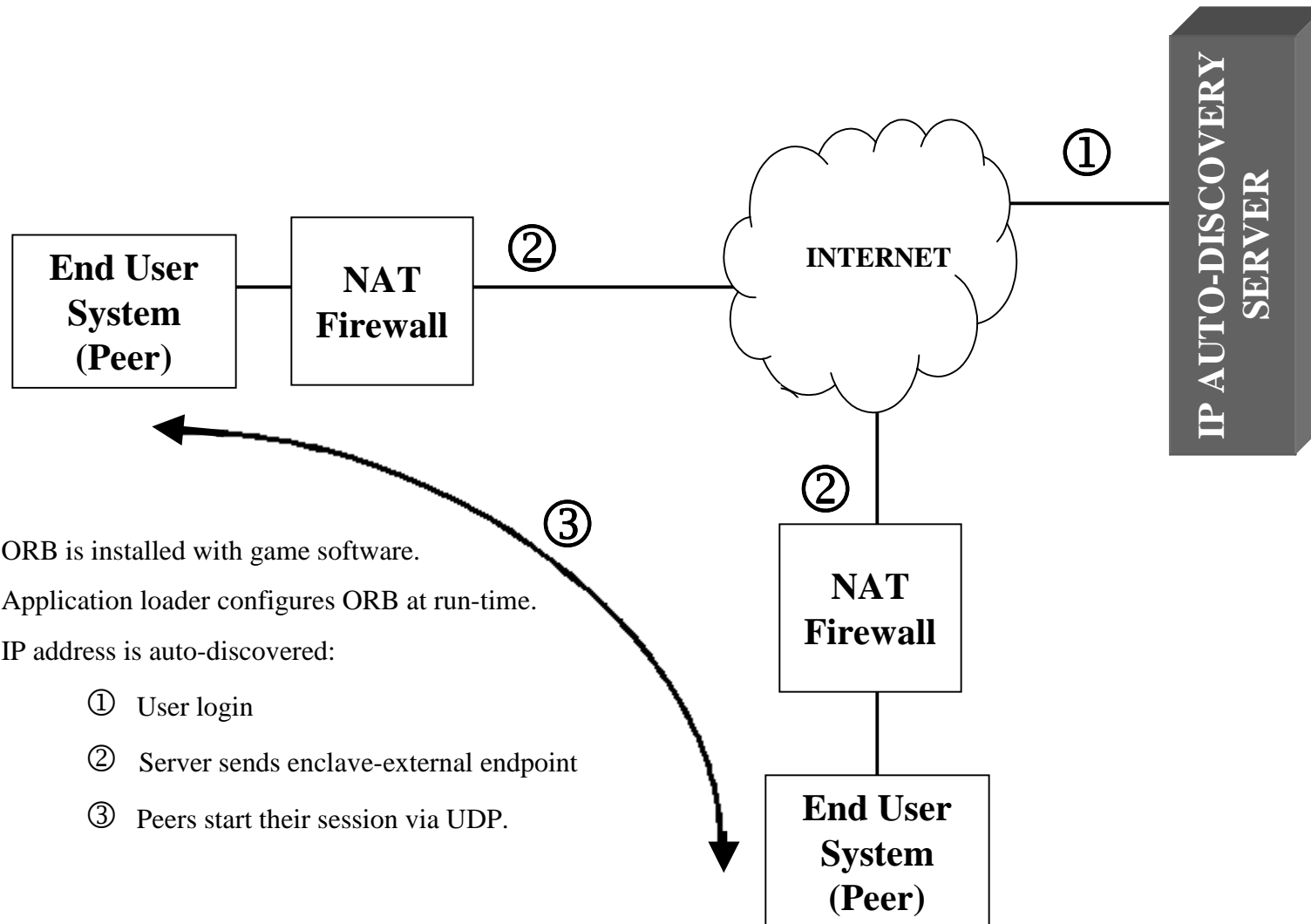
        //Leave the servant active until no longer needed
        some_event.wait();

    } //ExampleInterface_var example_servant goes out of scope.

    Ninja_Core_Singleton::instance()->shutdown_orb(TRUE);

    ACE::fini();
    return 0;
}
```

# Peer-to-Peer End-User System



# Client-Server End-User System

