The Service Configurator Framework

An Extensible Architecture for Configuring and Controlling Network Services

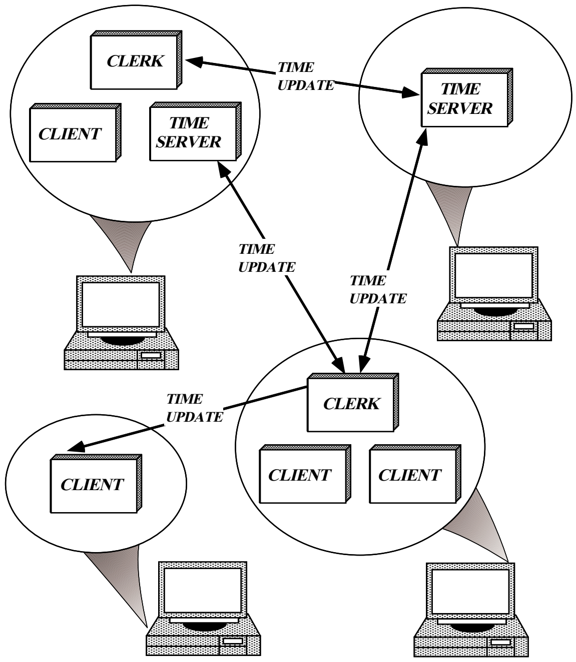
Douglas C. Schmidt

Washington University, St. Louis

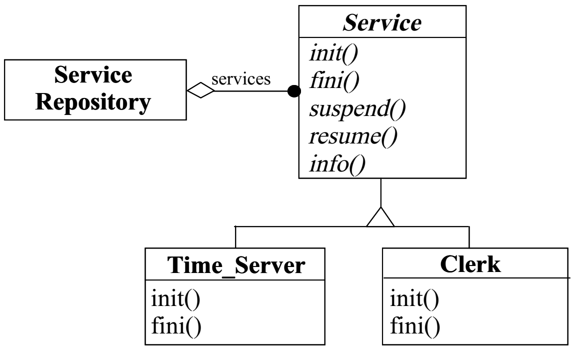
http://www.cs.wustl.edu/~schmidt/

schmidt@cs.wustl.edu

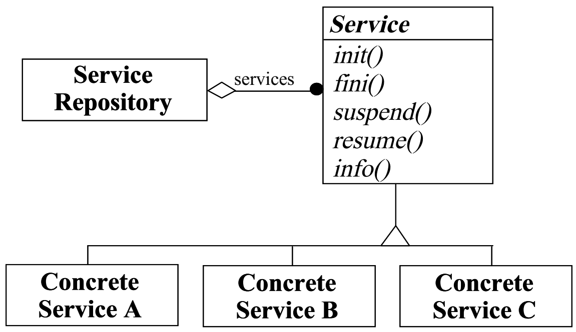
A Distributed Time Service



Structure of a Distributed Time Service



Structure of the Service Configurator Pattern



ace Service_Object Class

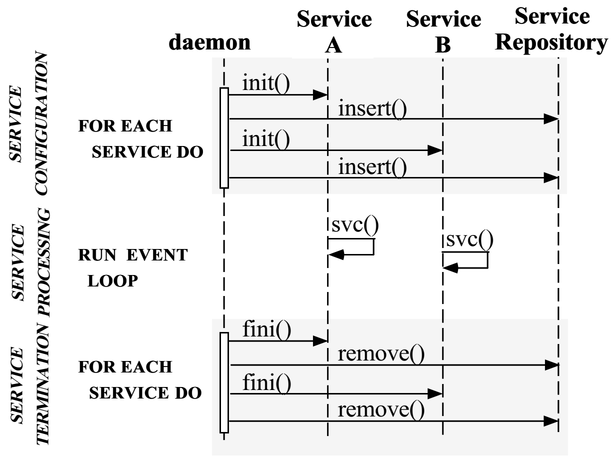
- Provides a hook for dynamic configuration

```
class ACE_Service_Object
{
public:
  // = Initialization and termination hooks.
  virtual int init (int argc, char *argv[]) = 0;
  virtual int fini (void) = 0;

  // = Informational hook.
  virtual int info (char **, size_t) = 0;

  // = Also contains Event_Handler
  // hooks for demultiplexing.
};
```


Interaction Diagram for the Service Configurator Pattern



The Time Server Class

```
class Time_Server : public ACE_Service_Object
{
public:
  Time_Server (u_short port) { acceptor_.open (port); }

  // Initialize the service when linked dynamically.
  virtual int init (int argc, char *argv[]) {
    // Parse command line arguments to get
    // port number to listen on...

    // Set the connection acceptor endpoint into
    // listen mode (using the Reactor pattern)
    acceptor_.open (port_);
  }

  // Terminate the service when dynamically unlinked.
  virtual int fini (void) {
    acceptor_->close (); // Close down the connection.
  }

  // info(), suspend(), and resume() methods omitted.

private:
  // Acceptor used to accept all connections.
  ACE_Acceptor<Time_Handler, ACE_SOCK_Stream> acceptor_;

  // Port the Time Server listens on.
  int port_;
};
```


The Clerk Class

```
class Clerk : public ACE_Service_Object
{
public:
  // Initialize the service when linked dynamically.
  virtual int init (int argc, char *argv[]) {
    // Use the Iterator pattern to set up
    // connections.
    Clerk_Handler **handler = 0;

    for (ITERATOR iterator (handler_set_);
         iterator.next (handler) != 0;
         iterator.advance ()) {
      connector_.connect (*handler, handler->addr ());
  }

  // Terminate the service when dynamically unlinked.
  virtual int fini (void) {
    // Disconnect all time servers
    Clerk_Handler **handler = 0;
    for (ITERATOR iterator (handler_set_);
         iterator.next (handler) != 0;
         iterator.advance ())
      (*handler)->close ();
  }
```

```
  // info(), suspend(), and resume() methods omitted.

private:
  typedef ACE_Unbounded_Set <Clerk_Handler *>
          HANDLER_SET;
  typedef ACE_Unbounded_Set_Iterator <Clerk_Handler *>
          ITERATOR;

  // Set of Clerks and iterator over the set.
  HANDLER_SET handler_set_;

  // Connector used to set up connections
  // to all servers.
  ACE_Connector<Clerk_Handler, ACE_SOCK_Connector>
    connector_;
};
```


Non-Generic Main Program

```
int main (int argc, char *argv[])
{
  ACE_Service_Config daemon;

  // Ignore error values (for now).
  daemon.open (argc, argv);

  // Set up the port number for the local server socket...

  // Create an acceptor.
  Time_Server time_server ((ACE_INET_Addr) port_number);

  // Register the Acceptor with the Reactor Singleton
  ACE_Service_Config::reactor ()->register_handler
    (&time_server, ACE_Event_Handler::READ_MASK);

  ACE_Sig_Adapter shutdown_handler
    ((ACE_Sig_Handler_Ex)
     ACE_Service_Config::end_reactor_event_loop);

  // Register a SIGINT handler with the Reactor Singleton.
  ACE_Service_Config::reactor ()->register_handler
    (SIGINT, &shutdown_handler);

  // Loop ``forever'' handling HTTP client requests.
  daemon.run_reactor_event_loop ();
}
```

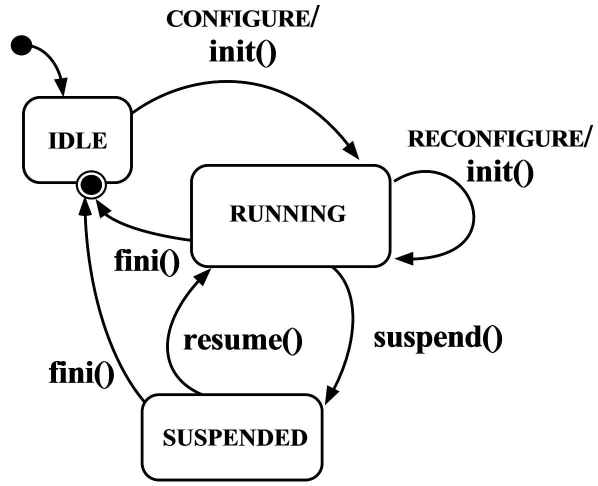

Generic Main Program

```
int
main (int argc, char *argv[])
{
  // Configure the daemon.
  ACE_Service_Config daemon;

  daemon.open (argc, argv);

  // Perform daemon services updates.
  daemon.run_reactor_event_loop ();
}
```


State Diagram of the Service Lifecycle



The svc.conf Service Configuration File

- Used to configure services dynamically and/or statically

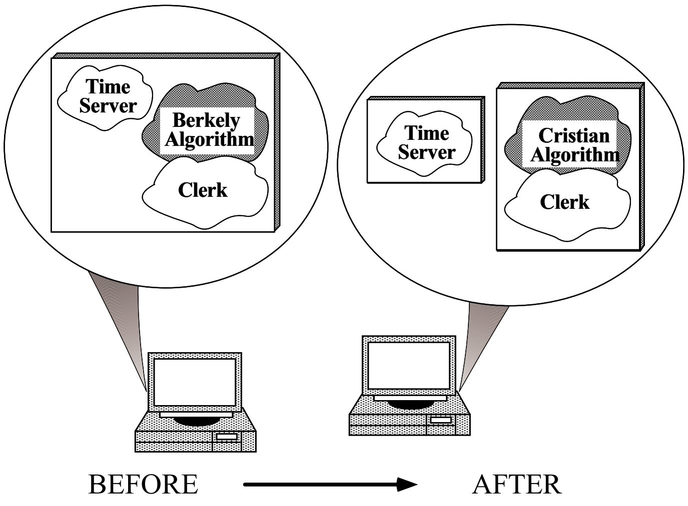
```
% cat svc.conf
# Configure a Time Server.
dynamic Time_Server Service_Object *
   libnet_svcs_a.dll:make_Time_Server()
     "-p $TIME_SERVER_PORT"

# Configure a Clerk.
dynamic Active_Time_Server_Clerk Service_Object *
   libnet_svcs_a.dll:make_Clerk()
     "-h tango.cs:$TIME_SERVER_PORT"
     "-h perdita.wuerl:$TIME_SERVER_PORT"
     "-h atomic-clock.lanl.gov:$TIME_SERVER_PORT"
```

- Factory function

```
extern "C" ACE_Service_Object *make_Time_Server(void)
{
  return new Time_Server;
}
```


Reconfiguring a Time Server and a Clerk



Reconfiguring a Timer Server Clerk

- Terminating a clerk

```
remove Time_Server_Clerk
```

- Reconfigure a new Clerk

```
dynamic Passive_Time_Server_Clerk Service *
   libnet_svcs_p.dll:make_Clerk()
     "-h tango.cs:$TIME_SERVER_PORT"
     "-h perdita.wuerl:$TIME_SERVER_PORT"
     "-h atomic-clock.lanl.gov:$TIME_SERVER_PORT"
```