Übungsaufgabe #6: Callback-Mechanismus

14.06.2004

In Aufgabe 5 wurde eine *Call-by-Value-Result*-Semantik für primitive Datentypen implementiert. In dieser Aufgabe soll nun eine echte *Call-by-Reference*-Semantik für Objektreferenzen implementiert werden.

Call-by-Reference bei Objekten

Wenn man auf den Zustand eines Objekts nur mit Methodenaufrufen zugreift (und nicht mit direkten Zugriffen auf Membervariablen), lässt sich eine *Call-by-Reference*-Semantik mittels Fernaufrufe realisieren. Hierzu ist grundsätzliche folgendes zu tun:

- Auf Clientseite wird bei einem Fernaufruf, der eine Objektreferenz als Parameter besitzt, ein Skeleton für das entsprechende Objekt erzeugt und eine geeignete *Remotereferenz* (z.B. eigener Rechnername, Portnummer, Objekt-ID) zum Server geschickt.
- Empfängt der Server eine solche Referenz, so wird dort ein entsprechender Stub erzeugt und mit der Remotereferenz initialisiert.

Lösungsmöglichkeit: Erstellen Sie sich ein Objekt `RemoteReference`, welches die Remote-Adresse eines Objekts darstellt. Um die Adresse für ein Objekt zu erstellen, kann ein Konstruktor vorgesehen werden. Mit Hilfe dieses Konstruktors kann der Client bei einem Fernaufruf Adressen zu den übergebenen Parametern erzeugen. Um diese zu übertragen kann die Klasse `Message` um eine Methode erweitert werden:

```
bool writeReference(RemoteReference r);
```

Auf Seite des Empfängers muss ein `RemoteReference`-Objekt erzeugt werden. Auch hierzu kann eine neue Methode der Klasse `Message` vorgesehen werden:

```
bool readReference(RemoteReference &r);
```

Sie bekommt als Parameter eine Referenz auf ein `RemoteReference` Objekt, welches mit den empfangenen Daten initialisiert werden soll. Ein mögliches Interface für `RemoteReference` könnte wie folgt aussehen:

```
class RemoteReference {
    public:
            RemoteReference(Address addr, int oid);
            RemoteReference();
            setReference(Address addr, int oid);
            int getOID();
            Address getAddress();
};
```

Einfache Beispielanwendung
Der implementierte Callback-Mechanismus soll mit einer einfachen Musteranwendung getestet werden. Hierbei sollen der eigenen Kreativität keine Grenzen gesetzt werden. Eine Möglichkeit stellt folgende IDL dar:

```
typedef char message[256];
interface Printer {
    void write(in message msg);
};
interface HelloServer {
    void sayHello(in Printer prt, in message msg);
};
```

Der Client soll lokal ein Objekt, welches das Interface Printer implementiert, erzeugen. Danach soll er einen Stub für den Remote-Zugriff auf ein HelloServer-Objekt instantiieren. An diesem ruft er die sayHello-Methode mit einer Printer-Objektreferenz als Parameter auf. Der Server soll lokal ein HelloServer-Objekt erzeugen. Der Skeleton übergibt bei sayHello-Aufrufen eine Referenz auf ein Stub-Objekt, das ebenfalls das Interface Printer implementiert. Als Ergebnis soll beim Client durch Rückruf vom Server ein Text ausgegeben werden. Ein Deadlock soll dabei natürlich nicht entstehen können.

Hinweis:
Ihr System muss in der Lage sein sowohl als Client als auch als Server zu dienen. D.h. während ein Fernaufruf (als Client) blockiert ist, da auf die Antwort gewartet wird, muss das System in der Lage sein eine neue Anfrage (als Server) zu bearbeiten. Erstellen Sie dazu einen Dispatcher, der alle eingehenden Nachrichten entgegennimmt und die folgenden 2 Fälle unterscheidet:

- Antwort auf einen laufenden Fernaufruf: In diesem Fall (als Client) muss die Nachricht an den entsprechenden Fernaufruf zugestellt werden. (Zur Vereinfachung kann davon ausgegangen werden, dass die Anwendung (welche die RPC-Schicht nutzt) nur einen Aktivitätsträger hat und daher maximal ein Fernaufruf auf eine Antwort wartet)
- neue Anforderung: In diesem Fall (als Server) muss die Anfrage bearbeitet und eine entsprechende Antwort zum Client übermittelt werden. (Zur Vereinfachung kann davon ausgegangen werden, dass während eine Anfrage bearbeitet wird keine Anfrage oder Antwort eintrifft.)

<u>Abgabe</u>: bis 25.06.2004 12:00 Uhr
Abgabe mittels /proj/i4vs/pub/abgabe