Prototyp für ein LVOOP basiertes mobiles Agentensystem

Frederik Berck und Dr. Holger Brand

GSI Helmholtzzentrum für Schwerionenforschung GmbH, Planckstraße 1, 64291 Darmstadt

Kurzfassung

Ein mobiler Agent ist eine Software, die eigenständig eine ihr zugewiesene Aufgabe ausführt, mit ihrer Umgebung interagieren kann und zusätzlich in der Lage ist, mitsamt ihren Daten zwischen verschiedenen Hosts zu migrieren.
Im Gegensatz zu konventionellen objektorientierten Programmiersprachen wie C++ oder JAVA werden Objekte in LabVIEW nicht als Entitäten behandelt, sondern als passive Datenobjekte, die dem Datenflusskonzept folgen. Die HGF Basisklassenbibliothek [1] stellt LVOOP Klassen bereit, die wichtige Entwurfsmuster [2] datenflusskonform implementieren, und die Behandlung von LabVIEW-Objekten als Entitäten erlaubt. Dabei wird streng zwischen Klassen unterschieden, die passive Datenobjekte (*HGF_Visitable*) repräsentieren und solchen, die jene aktivieren (*HGF_ThreadPool*). Als Beispiel wurde ein stationärer Geräteagent implementiert, der die Machbarkeit demonstriert. In diesem Beitrag wird beschrieben, wie sich dieses Konzept auf mobile Agenten erweitern lässt.
Das Main-VI, der sogenannte *Host*, stellt für jeden Agenten eine Arbeitsumgebung bereit, d.h. für jedes passive Agenten-Objekt wird je ein Task im ThreadPool gestartet, der eine Zustandsmaschine treibt, in der der Agent aktiviert wird. Die Kapselung eines Agentenobjektes innerhalb der privaten Zustandsdaten einer Zustandsmaschine bewahrt vor unerwünschtem Zugriff auf das Agentenobjekt oder Erzeugung eines Duplikates durch einen Drahtabzweig. Agenten können untereinander, mit den Hosts oder mit einer Anwenderapplikation ereignisgesteuert (*HGF_Event*) mittels passiver Nachrichten-Objekte (*Message*) kommunizieren. Diese werden als Strings („Flatten From/To String.vi“) in der Kommunikationsschicht übertragen, die mit Hilfe von Umgebungsvariablen implementiert ist. Ein Nachrichten-Objekt kann einfache Zeichenketten und ein LabVIEW Objekt, in diesem Zusammenhang speziell Besucher- oder Agenten-Objekt, enthalten. Besucher-Objekte ermöglichen das Aufrufen von Agenten-Methoden innerhalb der Agenten-Zustandsmaschine. Eine Migration von Agenten von einem Host zu einem anderen ist möglich, indem innerhalb der Zustandsmaschine eine Kopie des Agenten an einen Zielhost gesendet wird. Direkt anschließend wird die Aktivität der Zustandsmaschine und damit der zugehörige Task beendet und das ursprüngliche Agenten-Objekt zerstört.
Die nötige Sicherheit bei der Übertragung durch die öffentliche Kommunikationsschicht wird durch Verschlüsselung mittels GnuPG-Interface [3] erreicht. Jeder Host und Agent besitzt ein GPG-Objekt, das sich während der Initialisierung programmatisch ein Schlüsselpaar mit eindeutiger ID erzeugt, um Nachrichten-Objekte asymmetrisch zu verschlüsseln und mögliche Manipulationen zu erkennen. Mit Hilfe von Positiv- und Negativlisten von Bibliotheksnamen wird das dynamische Laden von Agenten- und zugehörigen Besucher-Klassen erlaubt oder verboten.
Das allgemeine Agentensystem wird durch konkrete Agenten erweitert, die von der Agenten-Basisklasse erben und SubVI’s implementieren, die von zugehörigen Agenten-Besucher-Kindklassen aufgerufen werden können.

[1] Dr. Holger Brand, Dr. Dietrich Beck, Frederik Berck, “Machbarkeitsstudie: LVOOP basiertes Agenten-System“, Virtuelle Instrumente in der Praxis, Begleitband zum Kongress VIP 2009, Seite 447, Hüthig Verlag, ISBN 978-3-7785-4057-2
[2] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, “Design Patterns - Elements of Reusable Object-Oriented Software”, Addison-Wesley, 1994
[3] The GNU Privacy Guard, http://gnupg.org/