Über programmgestützten Unit–Test von CORBA–Komponenten

Ulrich Hoffmann
Heidelberger Druckmaschinen AG, Kiel
Ulrich.Hoffmann@heidelberg.com

7. April 2002

Zusammenfassung

Ein Weg, große Softwaresystem zu realisieren, ist es, sie in überschaubare Funktionseinheiten (Komponenten) zu zerlegen, Schnittstellen zwischen diesen Komponenten festzulegen, dann zunächst jede Komponente unabhängig von anderen Komponenten zu realisieren und diese schließlich in einem Integrationsschritt zum Gesamtsystem zusammenzusetzen. Kommunikation zwischen Komponenten wird häufig über sogenannte Middleware abgewickelt. Wir konzentrieren uns hier exemplarisch auf die Common Object Request Broker Architecture (CORBA) [www.corba.org] als typische objektorientierte Kommunikationsschicht, die mit der Interface Definition Language einen Formalismus zur Plattform– und Sprachübergreifenden Spezifikation von Komponentenschnittstellen anbietet.

Praktizierte Softwareentwicklungsmethoden sehen nach Analyse, Design, Implementierung und vor der Integration mit weiteren Komponenten einen sogenannten Unit–Test vor, bei dem die implementierte Komponente isoliert gegen ihre (meist informelle) Spezifikation getestet wird. Im skizzierten Fall des Unit–Tests von CORBA–Komponenten entsteht die Schwierigkeit, dass die Komponenten häufig als Server–Programme realisiert sind, die nach dem Programmstart ohne eigene aktive Kommunikation auf das Ansprechen über eine ihrer CORBA–Schnittstellen warten. Eine für diesen Fall geeignete, realistische Unit–Test–Umgebung muß es erlauben, auf fehlerunanfällige Weise die Komponenten an ihren Schnittstellen zu stimulieren und ihre Reaktionen und Antworten zu überwachen. Zudem müssen Abstraktionen späterer Kommunikationspartner der Komponenten simuliert werden, um die Funktionalität der Komponenten möglichst weitgehend untersuchen zu können.

Die objekt-orientierte Programmiersprache Python [www.python.org] bietet über Zusatzmodule (etwa Fnorb [www.fnorb.org]) die Möglichkeit auf bequeme Weise mit CORBA–Komponenten kommunizieren zu können oder selbst CORBA–Komponenten zu realisieren. Durch ihren interaktiven Charakter und Datenstrukturen auf hohem Abstraktionsgrad, lassen sich Unit-Test-Umgebungen schnell und fehlerunanfällig programmieren und einzelne Testfälle leicht formulieren.

Der Vortrag stellt die auf Python basierende Unit-Test Umgebung vor, die bei der Entwicklung des Kontrollsystems NexStation für die digitale Farbdruckmaschine NexPress 2100 seit drei Jahren eingesetzt wird, und berichtet über die bisher gesammelten Erfahrungen.