Vom Makro-Recorder zum Software-Roboter

Wenn Software-Roboter dazu dienen mittels aufgezeichneter, menschlicher Eingaben eine Applikation maschinell und damit vollautomatisch zu steuern, was ist dann eigentlich der Unterschied zwischen einem Software-Roboter und einem Makro-Recorder?

Grundprinzip eines Makro-Recorders

In jeder Applikation mit Userinterface (Frontend) gibt es mindestens eine Funktionsschnittstelle über die alle Anwendereingaben laufen. Werden diese Funktionsaufrufe in einer Datei abgelegt, so kann der Inhalt der Datei wiederholt in die Schnittstelle eingespeist werden; mit demselben Ergebnis, als würde der Anwender wiederholt die Eingaben machen. Die Datei mit den gespeicherten Funktionsaufrufen nennt man ein „Makro“. Die Applikation bzw. den Teil der Applikation der die Funktionsaufrufe in die Datei ablegt bzw. von der Datei in die Schnittstelle einspeisen kann, nennt man den „Makro-Recorder“.

Erweiterter Makro-Recorder

Ein Makro-Recorder der nur nach dem beschriebenen Grundprinzip arbeiten würde, wäre in der Praxis fast nicht brauchbar, weil alle in einem Makro abgelegten Daten unveränderbar konstant sind und es im Alltag nur bei kleinsten Arbeitsschritten vorkommt, dass diese unverändert wiederholt werden müssen. Aus diesem Grunde sieht eine erste einfache Erweiterung des Makro-Recorders vor, dass in einen Makro nach dem Aufzeichnen konstante Werte durch Variablen ersetzt werden, wobei der Anwender den Wert einer Variablen kurzfristig, vor deren ersten Wiederverwendung festlegen

kann. Eine derartig einfache, aber doch unbedingt notwendige Erweiterung des Makro-Recorders lässt sich noch sehr leicht implementieren.

Skriptsprachen

Bei einen um Variablen erweiterter Makro-Recorder fehlt immer noch die Möglichkeit die Ausführung von Teilen des Makros von Bedingungen abhängig zu machen oder einen Teilschritt wiederholt in einer Schleife auszuführen. Die Hinzunahme dieser beiden Möglichkeiten verändert den Charakter eines Makros aber so stark, dass man hier nicht mehr von einen Makro sondern ein Skript spricht, wobei die Art und Weise wie die Anweisungen im Skript festgelegt werden, als Skriptsprache bezeichnet wird. Die Ausführung eines Skripts erfolgt mit einem sogenannten Interpreter, welcher deutlich aufwendiger zu implementieren ist als ein Makro-Recorder, insbesondere weil ein sogenannter „Parser“ notwendig ist, dessen Umsetzung tiefes fachlicher Informatik-Knowhow erfordert. Entgegen allgemeinen Programmiersprachen sind Skriptsprachen fast immer reduziert auf eine bestimmte Applikation oder Thematik in der sie eingesetzt werden. Eine typische Skriptsprache ist z.B. das in Microsoft Office Paketen verwendbare „Visual Basic for Applications“ (VBA).

Makros und Skripte

Während Makros „nebenbei“ maschinell aufgezeichnet werden, müssen Skripte händisch von einem Entwickler codiert werden. Das heißt, dass Erstellen eines Makros erfolgt auf komplett andere Weise als das Erstellen eines Skripts, denn das Aufzeichnen eines Makros entspricht im Vorgang dem Anlernen der Maschine durch „Vormachen“ des auszuführenden Arbeitsschrittes. Dieses „Vormachen“ entspricht deutlich mehr dem intuitiven Verhalten des Menschen einen Arbeitsschritt zu vermitteln, als ihn in abstrakter Form mit einer Skriptsprache zu formulieren. Damit befindet man sich in dem Dilemma, sich zwischen den geringen Möglichkeiten, aber intuitiv und einfach erstellbaren Makros oder den äußerst flexiblen, aber kompliziert zu erstellenden Skripten entscheiden zu müssen. Der in der Vergangenheit einzig praktizierbare und halbwegs brauchbare Ausweg aus diesem Dilemma bestand darin, die Makros in Form einer Skriptsprache so aufzuzeichnen, dass diese unmittelbar von einem Entwickler zu einem sinnvollen Skript erweitert werden können. Oft ist dies aber nur zur Einarbeitung in eine Skriptsprache von Nutzen, da dann die Skripte sehr simpel und deshalb nahe genug an einem Makro sind.

Intelligente Entwicklungsumgebungen

Noch bis vor wenigen Jahren wurden Skripte (oder allgemein Programme) mit simplen Texteditoren erstellt, während man heutzutage in sogenannten Entwicklungsumgebungen arbeitet. Der wesentliche Fortschritt besteht dabei darin, dass die verwendeten Editoren mit zusätzlicher Intelligenz ausgestattet sind, die den Entwickler beim Codieren unterstützen. Diese zusätzliche Intelligenz hat sehr viel Ähnlichkeit mit der, wie sie aus der Verwendung von Makro-Recordern resultiert. Das heißt konkret, der Anwender kann sich bestimmte, häufig auftretende Eingaben dadurch ersparen, dass ihn das System entsprechende verkürzte Eingaben („Shortcuts“) anbietet, ähnlich zu einem aufgezeichneten Makro. Ein typisches Beispiel dafür wäre die sogenannte IntelliSense-Funktionalität, die den Anwender beim Eintippen der ersten Buchstaben bereits den

kompletten Namen einer Variablen vorschlägt, den der Anwender nur noch mit einem Tastendruck bestätigen muss.

Code-Generatoren

Während die IntelliSense-Funktionalität auf einer Vielzahl von kleinen, fest implementierten Hilfen für den Entwickler basiert, stellen Code-Generatoren ein mächtiges Werkzeug dar, mit dem auf Knopfdruck Programmiercode in einem Umfang erstellt werden kann, wofür ein Entwickler Stunden oder gar Tage brauchen würde. Um näher an unserem Thema zu bleiben, ist es günstiger, statt allgemein von Code-Generatoren, von Skript-Generatoren zu sprechen. Zum einen sind Skripte nichts anderes als Programmcode und zum anderen wird dadurch die Richtung des Fortschritts mehr offensichtlich. Obig beschriebenes Dilemma in der Entscheidung zwischen der Verwendung von Makros oder Skripten wird bei Verwendung eines Skript-Generators dadurch aufgelöst, dass das Skript vollautomatisch generiert wird. Das heißt, statt des Menschen, legt eine Maschine die Anweisungen in einer Datei ab, die eine andere Maschine abarbeiten wird, - womit das Analogon zu Makros gegeben ist. Damit findet die Lösung des Problems auf einer höheren Abstraktionsstufe statt: Der Mensch gibt der Maschine vor, wie sie die Anweisungen für eine (andere) Maschine zu geben hat. Wir nähern uns dem Software-Roboter - aber vom Skript-Generator zum Software-Roboter ist noch ein weiter Schritt

Software-Roboter

Der Leser mag bereits jetzt erahnen, dass Makro-Rekorder die Urahnen der Software-Roboter sind, denn beide dienen dem Zweck die Datenbearbeitung durch den Menschen zumindest in Teilen zu automatisieren. Aber in einen Makro-Recorder ist fest einprogrammiert, wie er die Daten aus einer Makro-Datei zu lesen hat und was er damit tun soll – nämlich in die Anwenderschnittstelle einspeisen. Ein Software-Roboter dagegen ist diesbezüglich viel flexibler. Ein oder mehrere Skripten ‚sagen' ihm, wie er die Input-Datei(en) zu lesen hat und was er mit den gelesen Daten tun soll. Ein Einspeisen der Daten in eine Anwenderschnittstelle ist eine Möglichkeit, das Erzeugen von Output-Dateien in beliebigen Format eine andere. Zum Beispiel können dies Skript- oder allgemeine Code-Dateien sein, wie sie ein Code-Generator erzeugen würde oder ein zusammenfassender Bericht für die Geschäftsleitung, wie ihn ein Dokumentengenerator erzeugen würde oder eine Excel Tabelle als Ergebnis einer Internetrecherche oder … oder …

Vergleichbar zu einem physikalischen Roboter ist ein Software-Roboter eine hochflexible Automatisierungssoftware, die mittels Instruktions- bzw. Skriptdateien für eine spezielle Aufgabe eingerichtet wird und diese dann auf Knopfdruck vollautomatisch durchführt. Entscheidend dabei ist, dass das Einrichten eines Software-Roboters ungewohnt viel einfacher ist, als das Programmieren einer entsprechenden, herkömmlich entwickelten Automatisierungssoftware. Eine Zeitersparnis um den Faktor größer 10 ist dabei normal. Genau dieser Faktor ermöglicht es Software-Roboter überaus kurzfristig für Aufgaben zu instruieren und sofort einzusetzen, ohne dass das Wagnis ‚Aufsetzen eines Softwareprojektes zum Erstellen einer herkömmlichen Automatisierungssoftware' eingegangen werden muss.

Der Software-Roboter als Assistent

Normaler Weise wird der Software-Roboter vom Anwender auf Knopfdruck gestartet und soll seine Aufgabe sofort und möglichst schnell erledigen. Hierbei erfüllt er seinen Zweck als extrem nützliches Werkzeug. Zum umsorgenden Assistenten wird der Software-Roboter aber erst, wenn er seine Umgebung beobachtet und auf bestimmte Ereignisse mit bestimmten Aktionen reagiert. Mit der Umgebung des Software-Roboter ist natürlich seine digitale Umgebung gemeint; also alle Daten die ihm (in digitaler Form) zur Verfügung stehen. Ein typisches Ereignis auf das der Software-Roboter reagieren kann ist zum Beispiel, wenn sich ein bestimmtes Datum in seinem Wert ändert oder, wenn es einen bestimmten Wert annimmt. Ersteres könnte z.B. das Speicherdatum einer Datei sein, letzteres der Eintritt einer bestimmtem Tagesuhrzeit. Eine dazu passende Aktion könnte z.B. das Aktualisieren eines Berichts und die Benachrichtigung einer Person sein.

Welche Daten der Software-Roboter überwachen kann, hängt grundsätzlich davon ab, auf welche Datenquellen er Zugriff hat. Bei einen PC kann das Betriebssystem oder das Dateisystem als Datenquelle dienen. Hat der PC eine Verbindung zum Internet, so steht auch dieses als Datenquelle zu Verfügung. In einen Mobilgerät (Smartphone) könnte auch GPS („Ziel erreicht!“) oder die Kamera eine Datenquelle sein. Bei Embedded Systemen (z.B. Haustechnik) stehen meist noch viel ausgefallenere Datenquellen (wie z.B. ein Rauchmelder) zur Verfügung. Der Phantasie sind hier keine Grenzen gesetzt.

Der Traum ist aber schnell ausgeträumt, wenn auf herkömmlich Weise eine entsprechende Automatisierungssoftware programmiert werden soll, da aufgrund des zu erwartenden hohen Aufwands, oft mit deren Erstellung erst gar nicht begonnen wird.

Ganz anders, wenn ein passender Software-Roboter zur Verfügung steht. Dann werden Ideen mehr oder weniger sofort Realität, welches wiederum die Phantasie beflügelt.

Weiterführende Dokumente:

http://www.sss.de/files/triple-s/downloads/downloads_X2X_2015/Software-Roboter%20Definition.pdf

http://www.sss.de/files/triple-s/pdf/X2X-Software-Roboter-Broschuere.pdf

http://www.sss.de/x2x-downloads?file=files/triple-s/downloads/downloads_X2X_2015/Sourcecode_generieren.pdf

http://x2x.sss.de