

Endbenutzer-Programmierung mit dem Apple Automator

von Lars Blumberg im Sommer 2008

Hasso-Plattner-Institut für Softwaresystemtechnik GmbH

Zusammenfassung. Im Rahmen des Seminars *End-user programming* habe ich mich mit der Software *Apple Automator* beschäftigt, um festzustellen inwieweit sie für den Endbenutzer geeignet ist, Arbeitstätigkeiten am PC ohne allgemeine Programmierkenntnisse zu automatisieren. Da ihr Bedienkonzept auf dem Pipe-Operator beruht, gebe ich zu Beginn Informationen über den historischen Hintergrund bevor diese grundlegenden Gedanken auf den Automator übertragen werden. Im Hauptteil dieser Ausarbeitung erläutere ich detailliert die Bedienung des Automators, stets mit dem Leitgedanken die Eignung für den Endanwender bewerten zu können. Dabei habe ich ebenso die Umsetzung von bekannten und wichtigen Programmierkonzepten untersucht. Zuletzt beleuchte ich einmal die Benutzergemeinde des Automators etwas näher und strebe Vergleiche zu anderen Automatisierungstools an.

1 Einleitung

„Computer sind zu einem festen Bestandteil der Lebens- und Arbeitswelt vieler Menschen geworden. In immer größerem Maße benutzen diese Endanwender von Computerprogrammen nicht nur vorgefertigte Software, sondern passen diese an ihre Bedürfnisse an, erweitern die Software und teilen dann ihre Ergebnisse mit anderen. Jedoch können nicht alle Endanwender, die fachlich kompetent sind, gleichzeitig programmieren oder haben das Interesse, sich diese Fähigkeit anzueignen. Methoden und Techniken, die Endanwender ohne formale Programmierausbildung dabei unterstützen eigene Software zu entwickeln, sind Scripting- und Domäne-spezifische Sprachen, Programmieren am Beispiel, visuelle und natürlichsprachliche Programmierung.“ Diese Einführung stammt aus der Beschreibung zur Veranstaltung *End-user programming*¹ des Fachgebiets für Software-Architekturen am Hasso-Plattner-Institut und bietet mir einen guten Einstieg in das Thema. In dieser Ausarbeitung möchte ich nämlich der Frage nachgehen, inwieweit das Programm *Apple Automator* den Endbenutzer dabei unterstützt, ohne allgemeine Programmierkenntnisse eigene Software zu entwickeln, um „ihre Arbeit zu erleichtern, Zusammenhänge zu verstehen oder ihre Freizeitaktivitäten zu unterstützen“.

¹ Mit meiner Teilnahme an diesem Seminar im Sommersemester 2008 ist unter anderem diese Ausarbeitung entstanden.

Das Computerprogramm *Apple Automator* ist Teil der Standardsoftware zum Betriebssystem MacOS seit der Version 10.4. Auf jedem neuen Macintosh-Computer befindet sich also neben dem aktuellen MacOS 10.5 eine Installation des Automators, inzwischen in der zweiten Hauptversion. Bevor ich gleich auf die Bedienweise dieses Computerprogramms eingehe, möchte ich Ihnen zuvor jedoch noch einige Hintergrundinformationen geben. Denn obwohl es sich beim Automator um ein vergleichsweise junges Softwareprodukt handelt, geht sein zu Grunde liegendes Bedienkonzept bereits auf die frühen 70er Jahre zurück. Und darauf lohnt es sich zunächst einen näheren Blick zu werfen.

1.1 Historischer und konzeptioneller Hintergrund

Bereits Ende 1950 ([2], [3]) hat sich Herr Doug McIlroy Gedanken darüber gemacht, wie man mit Hilfe eines Computers kleinere Aufgaben zu einer großen zusammenzufassen könnte, um diese dann in einem Schritt automatisch ausführen zu lassen. Schon damals dachte er an die Verknüpfung von Ergebnis und Eingabe zweier verschiedener Aufgaben zum Zwecke der Realisierung einer größeren. Ihm ging diese Idee lange nicht aus dem Kopf und 1972 schließlich, Kenneth Lane Thompson war gerade mit der Programmierung einer frühen Version des UNIX-Betriebssystems beschäftigt, wurde dieses Thema wieder aktuell. Angeblich in nur einer Nacht schrieb Thompson den Quelltext von UNIX so um, dass er daraufhin die von McIlroy geforderte Aneinanderkettung von mehreren Prozessen erlaubte. Mittels der später noch einmal geänderten Notation kann seit dem die Ausgabe eines Programms als Eingabe in ein anderes Programm umgelenkt werden. Nach einer Änderung der Notation wird seit dem der *Pipe-Operator* „|“ benutzt, der wie folgt verwendet wird:

```
echo -e "orange, \npeach, \ncherry," | sort | say
```

Listing 1.1. Beispielhafte Verwendung des Pipe-Operators.

Dieses Beispiel², welches im Terminal eines Macintosh-Computers eingegeben werden kann, führt zu Beginn den ersten Befehl `echo -e "orange, \npeach, \ncherry,"` aus. Alleinstehend würde dieser Befehl zeilenweise die drei angegebenen Früchte `orange`, `peach` und `cherry` auf dem Bildschirm ausgeben. Die Option `-e` hilft dabei, dass `\n` als Zeilenumbruch verstanden wird. Der `echo`-Befehl ist nun aber mit dem Befehl `sort` mit Hilfe des Pipe-Symbols `|` verbunden. Dieser bewirkt, dass die Früchte eben nicht auf dem Bildschirm ausgegeben werden sondern als Eingabe für den Befehl `sort` verwendet werden. Auch der Befehl `sort`, der die übergebenen Zeilen alphabetisch sortiert, würde sein Ergebnis in das Terminal schreiben, stünde dort nicht wieder eine Pipe, gefolgt von dem Befehl `say`. `say` greift im letzten Schritt die sortierten Früchte auf und eine Computerstimme liest den sortierten Text *cherry, orange, peach* vor.

Allgemein können wir die Notation für die Verwendung von Pipes also wie folgt formulieren:

² Dieses und weitere Beispiele sind Bestandteil einer zur Ausarbeitung gehörenden Videoaufnahme, die unter [1] abgerufen werden kann.

```
command_1 | command_2 [ | command_3 ...]
```

Abb. 1. Allgemeines Schema für die Verwendung der Pipe (nach [3]).

Dieses Konzept der Pipe hat sich aufgrund der vielseitigen Verwendung bis heute bewährt und ist immer noch in allen gängigen Betriebssystemen zu finden.

1.2 Einführung in den Automator



Abb. 2. Programmlogo des Apple Automators.

Die Entwickler des Apple Automator haben dieses Pipe-Konzept ebenso aufgegriffen, allerdings bevorzugten sie eine andere Visualisierung. Anstatt die Befehle mit der Tastatur in einer Konsole aneinanderreihen zu müssen, verbindet der Benutzer nun grafische Elemente miteinander. Diese Elemente heißen hier nicht Befehle (oder eigentlich besser: Kommandozeilenprogramme), sondern *Aktionen* (Actions). Diese Aktionen werden aber konzeptionell genauso verknüpft wie die Befehle in einer Konsole mit der Pipe: Das Ergebnis einer Aktion wird von der nachfolgenden Aktion als deren Eingabe konsumiert. Diese verarbeitet daraufhin die Eingabe und produziert eine Ausgabe, die wiederum von der nächsten Aktion konsumiert werden kann. Die Aneinanderreihung mehrerer Aktionen wird übrigens als *Arbeitsablauf* (Workflow) bezeichnet.

In der Abbildung 3 wird die Ähnlichkeit zur Pipe-Notation deutlich³. Auch beim Automator handelt es sich um eine Aneinanderreihungen von beliebig vielen Aufgaben, die gemeinsam eine größere Aufgabe erledigen. Eine Aufgabe gibt auch hier ihr produziertes Ergebnis an die nachfolgende Aufgabe weiter.

Links und rechts des schematischen Arbeitsablaufs habe ich beispielhaft veranschaulicht, welche Aktionen der Automator bereitstellt und welche Daten zwischen den Aktionen ausgetauscht werden können. Die Symbole auf der linken

³ „Otto“ – wie der Spitzname des Roboters aufgrund der englischen Aussprache von „Automator“ lautet – hält deshalb auch eine Röhre in den Händen, welche an die Pipe erinnern soll (vgl. Abb. 2).

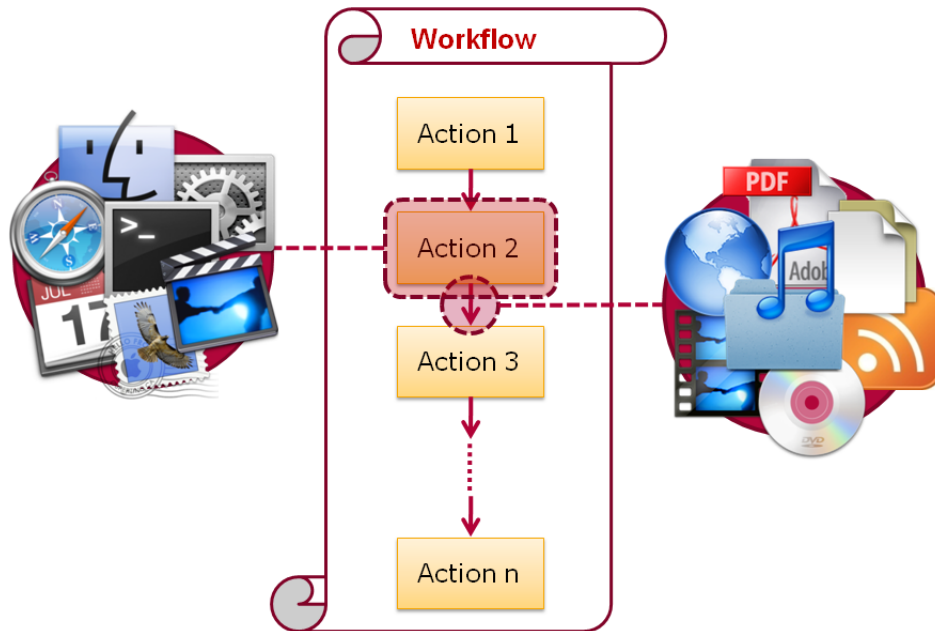


Abb. 3. Allgemeines Schema eines Automator-Workflows.

Seite stehen für Aktionen, die Kalendereinträge im Programm *iCal* bearbeiten und erzeugen können, Video-DVDs erzeugen, *iTunes*-Titel importieren, E-Mails erstellen und versenden und Webseiten durchsuchen. Weiterhin steht das Finder-Logo für die Anbindung an das Dateisystem, denn es gibt Aktionen, die Dateien erstellen, verschieben, duplizieren, löschen oder auch komprimieren können. Im Gegensatz zur Pipe in der Kommandozeile werden zwischen den Aktionen typisierte Daten und kein einfacher Datenstrom ausgetauscht. Dies bedeutet aber auch, dass zwei aufeinanderfolgende Aktionen in der Aus- und Eingabe zusammenpassen müssen. Vereinfacht gesagt kann eine Aktion, die Bilder verarbeitet, keine E-Mail entgegennehmen, die zuvor von einer anderen Aktion erstellt wurde. Neben Bildern und E-Mails können Aktionen weiterhin Musiktitel, Dateien aus dem Dateisystem im Allgemeinen, RSS-Feeds, PDF-Dokumente oder Webseiten verarbeiten. Wie gesagt handelt es sich auch hier nur um eine kleine, veranschaulichende Auswahl.

Mit dieser konzeptionellen Erfassung habe ich jetzt zwar den theoretischen Hintergrund des Automators vermittelt. Inwieweit sich dieses Programm aber auch sinnvoll einsetzen lässt und ob er auch von Endanwendern ohne fachliche Programmierkenntnisse bedient werden kann, möchte ich nun im Hauptteil der Ausarbeitung herausfinden.

2 Endbenutzer-Programmierung mit dem Automator

Gleich nach dem Start der Anwendung erscheint dem Benutzer zu Beginn ein Assistent, der ihm helfen soll, die erste Aktion zu wählen. Dieser Dialog ist die Konsequenz, die man aus einer wenig überraschenden Beobachtung gezogen hat als es diese Hilfestellung noch nicht gab. Es ist eine Situation wie sie vielfach beim Benutzen einer neuen Software auftritt: Den Benutzern schwebt zwar vor, welche Art von Problemstellung sie damit lösen wollen und sie wissen zumindest, dass dieses Programm ihnen dabei helfen kann. Sie wissen aber weder auf Anhieb wie das Programm zu bedienen ist, noch welchen Schritt sie als ersten durchführen *müssen* um mit der Problemlösung zu beginnen. Nach dem Motto „Aller Anfang ist schwer“ bietet der Automator dem Benutzer deshalb zu Beginn fünf Startmöglichkeiten an, die häufig enthaltene Arbeitsaufgaben beschreiben (vgl. Abb. 4).

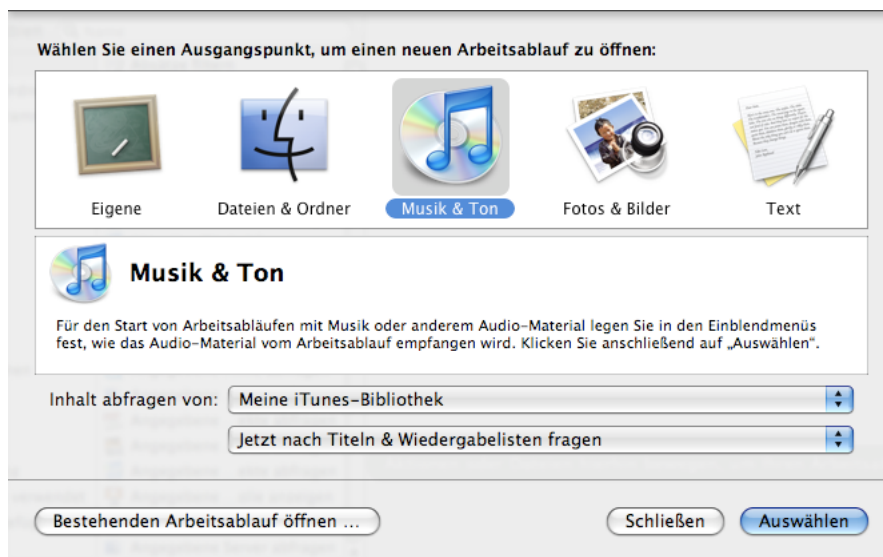


Abb. 4. Dieser Assistent hilft dem Benutzer eine erste, geeignete Aktion zu finden.

Das Festlegen des gewünschten Startpunktes – hier die Auswahl von vorhandenen *iTunes*-Titeln – fügt bereits erste Aktionen in den neuen Arbeitsablauf ein. Mit einem einfachen Beispiel werde ich auch das folgende Kapitel einleiten; allerdings möchte ich die gute Anbindung des Automators an das Betriebssystem aufzeigen. Dazu beginne ich mit der Erstellung des Arbeitsablaufs nicht erst im Automator sondern bereits in dem Dateibrowser des *Finder*.

2.1 Einfache Arbeitsabläufe

Um Ihnen die Bedienweise des Automators aufzuzeigen, möchte ich mit einem überschaubaren, aber eindrucksvollen Beispiel beginnen. Und zwar möchte ich eine Reihe von Urlaubsfotos, die in einer hohen Bildauflösung aufgenommen wurden, mit einer reduzierten Dateigröße im Internet veröffentlichen. Sicherlich gibt es eine Reihe von Tools, mit denen sich diese Aufgabenstellung auch einzeln realisieren ließe: zuerst verwenden wir ein Programm, das alle Dateien automatisch eine gute Nummerierung vergibt. Danach skalieren wir die Bilder mit einem anderen Programm auf eine geeignete, Web-gerechte Auflösung. Und zum Schluss bemühen wir einen FTP-Client für die Veröffentlichung der Bilder im Internet. Die Stärke des Automators liegt nun aber darin, dass er diese Schritte nicht nur alle vereint; sondern dass der dadurch entstehende Arbeitsablauf anschließend zur Wiedervorlage erneut verwendet werden kann – bspw. für die Bilder des nächsten Ausflugs.

Nach dem Markieren der Bilddateien im Finder steht im Kontextmenü der Stichpunkt *Automator / Arbeitsablauf erstellen* zur Verfügung⁴, über den ich den Automator starte. Dort ist auf der rechten Programmseite bereits die zugehörige Aktion *Angegebene Finder-Objekte abfragen* zu finden (Abb. 5).

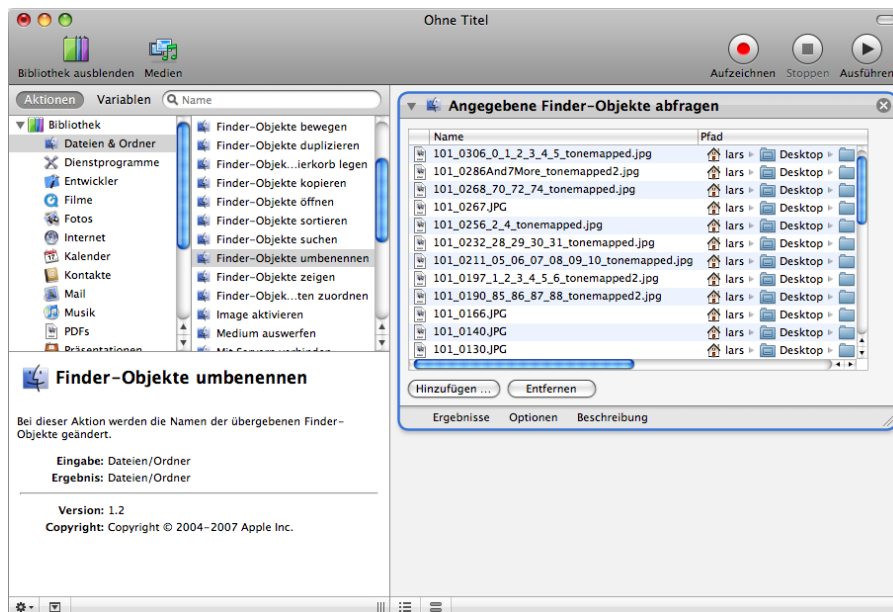


Abb. 5. Der Automator nach der Auswahl einiger Bilddateien im Finder.

⁴ Auch dieses Beispiel kann in der Videoaufnahme [1] angeschaut werden.

Auf der linken Programmseite des Automators findet der Benutzer die zur Verfügung stehenden Aktionen in der *Bibliothek* wieder. Da sie übersichtlich in verschiedene Kategorien unterteilt sind und dort auch eine Suchfunktion zur Verfügung steht, ist der Endbenutzer in der Lage die gewünschten Aktionen schnell ausfindig zu machen. Das Beschreibungsfeld unterhalb der Bibliothek gibt detaillierte Auskunft über eine ausgewählte Aktion: Hier erfährt der Nutzer nicht nur, welche Arbeit eine Aktion automatisiert, sondern auch welche Eingabe zugelassen und welches Ergebnis produziert wird. Die gewünschte Aktion kann der Benutzer nun mittels Drag & Drop oder einem Doppelklick zum Arbeitsablauf hinzufügen.

Da die oben erwähnten Urlaubsfotos also im nächsten Schritt erst einmal eine einheitliche Benennung mit fortlaufender Nummer bekommen sollen, füge ich als nächstes die Aktion *Finder-Objekte umbenennen* hinzu. Dass die bisherige und die neue Aktion zusammenpassen, erkennt der Nutzer an das Ineinandergreifen der beiden Aktionen. Hier wird noch einmal ganz deutlich, dass eine Aktion ihr produziertes Ergebnis als Eingabe für die nachfolgende Aktion weitergibt.



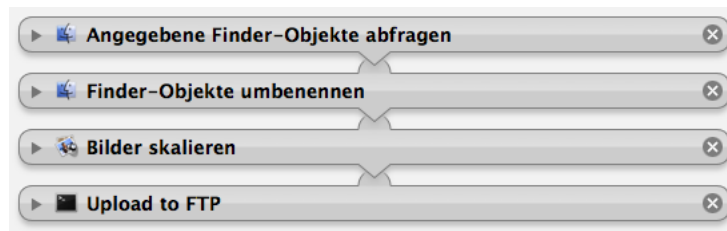
Abb. 6. Zwei aufeinanderfolgende Aktionen greifen ineinander, wenn Ergebnis- und Eingabetyp zusammenpassen.

Jede Aktion bietet eine kleine Oberfläche an, in der Aktions-spezifische Konfigurationen vorgenommen werden können. Im Falle der ersten Aktion ist es die Liste der auszuwählenden Eingabedateien. In der zweiten Aktion kann der Endbenutzer die Art der Umbenennung festlegen. So kann bspw. das Erstelldatum der Datei als neuer Name dienen oder es können einzelne Fragmente des Dateinamens durch andere Begriffe ersetzt werden. Ich entscheide mich hier für eine fortlaufende Nummerierung der Bilddateien. Nach dem gleichen Prinzip werden auch die übrigen Aktionen zum Arbeitsablauf hinzugefügt und konfiguriert, um den gewünschten Endzustand zu erreichen. Dies stellt natürlich eine Herausforderung für den Endbenutzer dar; er muss sein zu lösendes Problem in kleinere Teilschritte zerlegen, die von den Aktionen automatisiert werden sollen. Ein umfangreiches Wissen über alle verfügbaren Aktionen sehe ich dabei als Vorteil an. So kann der Endbenutzer gut abschätzen, ob es überhaupt eine passende Aktion für einen Teilschritt gibt und welche Aktion für einen Teilschritt geeignet scheint. Um unsere eingangs erwähnte Aufgabenstellung zu bewältigen, habe ich also die in Tabelle 2.1 aufgelisteten Aktionen identifiziert.

Tabelle 1. Vollständige Aktionskette des beispielhaften Arbeitsablaufs

<i>Teilschritt</i>	<i>Konsumiert</i>	<i>Produziert</i>
1. Bilder auswählen	–	Liste der ausgewählten Dateien
2. Dateien umbenennen	Liste von Dateien	Liste der umbenannten Dateien
3. Bilder skalieren	Liste von <i>Bilddateien</i>	Liste der skalierten Bilddateien
4. Bilder veröffentlichen	Liste von Dateien	Liste der Internet-Adressen (URLs), unter denen die Dateien dann erreichbar sind

Nach dem Einfügen von zwei weiteren Aktionen – dem Skalieren der Bilder und dem Veröffentlichen der Bilddateien⁵ – ist der Arbeitsablauf bereits vollständig. Um Platz auf dem Bildschirm zu sparen, kann der Endbenutzer jede Aktion einklappen, so dass nur noch deren Name zu sehen ist. Die folgende Abbildung 7 zeigt den so erstellten, fertigen Arbeitsablauf; gestartet wird er mit einem Klick auf den *Ausführen*-Knopf.

**Abb. 7.** Der fertige Arbeitsablauf aus dem Einführungsbeispiel.

Die Lösung ließe sich jetzt noch beliebig erweitern; beispielsweise könnte der Nutzer die entstandenen Internet-Adressen mit einer E-Mail-Aktion an seine Bekannten versenden lassen sobald alle Bilder veröffentlicht sind. Mit diesem einfachen Einführungsbeispiel wollte ich jedoch vor allem verdeutlichen, wie leicht es sein kann, mit dem Automator wiederkehrende Tätigkeiten am Computer zu automatisieren. Aufgrund der Vielfalt der zur Verfügung stehenden Aktionen kann der Endbenutzer ein breites Spektrum an Arbeitsaufgaben automatisieren. So stehen u. a. Aktionen aus den Aufgabenbereichen wie Dateiverwaltung, E-Mail, Software-Entwicklung, Internet oder Medien-Verarbeitung zur Verfügung. Von dieser Vielfalt profitieren nicht nur private Endanwender; viele Anbieter von professionellen Software-Produkten stellen inzwischen Automator-Aktionen zur Verfügung, mit denen sich Arbeitsschritte in deren Software automatisieren

⁵ Die Aktion „Upload to FTP“ habe ich von der Quelle [4] zusätzlich installiert, da sie nicht zum Standardumfang des Automators gehört.

lassen. Im gewerblichen Umfeld setzen Endanwender zum Beispiel *Photoshop*-spezifische Aktionen ein, um große Bild-verarbeitende Arbeitsketten zu automatisieren. Diese umfassen die Stapel-weise Nachbearbeitung von Bildern aus Rohdaten mit automatischer Anwendung von Filtern über die Sortierung und Einordnung anhand von Metadaten bis hin zur automatischen Produktion der aufbereiteten Bilder auf CDs inklusive dem Ausdruck eines gleichzeitig erstellten PDF-Katalogs dieser Bilder.

Erfreulich ist, dass trotz dieser umfangreichen Einsatzmöglichkeiten das Bedienkonzept des Apple Automator auf einem einfachen und meiner Beobachtung nach sehr leicht zu erlernenden Bedienkonzept beruht: Der Endbenutzer muss lediglich die geeigneten Aktionen aus der gut sortierten Bibliothek zum Arbeitslauf hinzufügen. Schritt um Schritt entsteht somit eine Prozessvorlage, die stets wiederkehrende, monotone Arbeitsabfolgen automatisiert und dem Benutzer mehr Zeit für kreative und wesentlichere Tätigkeiten lässt.

Der Apple Automator ist also, wie ich es soeben aufgezeigt habe, nicht nur für den fachlich unerfahrenen Computer-Benutzer interessant. Auch der fachlich versiertere Endanwender nutzt den Automator, stellt aber aufgrund seiner Erfahrungen mit anderen Computeranwendungen weitaus höhere Forderungen an eine Programmierumgebung, wie sie auch der Automator darstellt. Wer komplexere Arbeitsabläufe entwirft, kommt nämlich schnell in eine Situation, in der er gern den Arbeitsablauf bei nicht erfüllten Vorbedingungen abbrechen möchte; er versucht ist, einige Teilschritte wiederholt auszuführen oder einen bereits bestehenden Arbeitsablauf einbinden möchte. Inwieweit der Apple Automator diesen Erwartungen an die Fähigkeit zur Gestaltung eines flexibleren und besser kontrollierbaren Automatisierungsprozesses gerecht werden kann, möchte ich Ihnen im nun folgenden Abschnitt aufzeigen.

2.2 Anspruchsvollere Arbeitsabläufe

Nun möchte ich Ihnen zeigen, wie Sie als Endbenutzer wesentlich anspruchsvollere Arbeitsabläufe realisieren können. Unter „anspruchsvoll“ verstehe ich die Programmierung jenseits des linearen Ablaufs, wie Sie ihn bisher kennengelernt haben. Arbeitsabläufe sind nämlich nicht darauf beschränkt, dass ihre Abwicklung in notwendiger Weise von der ersten bis zur letzten Aktion eines Arbeitsablaufs erfolgen muss. Außerdem gibt es Möglichkeiten, auf frühere Ergebnisse einer Aktion im Ablauf zurückgreifen. Der Automator stellt dazu Werkzeuge bereit, die an bekannte Konzepte aus der prozeduralen Programmierung erinnern. Dabei handelt es sich erst einmal nur um die Verwendung von Variablen, Schleifen und Methodenaufrufen. Als Sonderform betrachte ich anschließend die Möglichkeit, Skripte in einen Arbeitsablauf zu integrieren. Sie ermöglichen einen weitaus höheren Freiheitsgrad beim Entwurf von Arbeitsabläufen und stellen auch diejenigen Endbenutzer zufrieden, welche mit den vordefinierten Aktionen schnell die Einsatzgrenzen des Automators erfahren.

2.2.1 Speichern von Aktions-Ergebnissen in Variablen

Manchmal bietet es sich an das von einer Aktion produzierte Ergebnis für die spätere Ver-

wendung aufzuheben. Auch wenn die nachfolgende Aktion dieses Ergebnis zwar sofort konsumiert, entsteht manchmal beim Entwurf eines Arbeitsablaufs die Situation, dass dieses Ergebnis zu einem späteren Zeitpunkt erneut oder überhaupt erst zu einem späteren Zeitpunkt benötigt wird. Ich habe zum Beispiel öfter Arbeitsabläufe erstellt in denen eine Liste von Dateien erzeugt, komprimiert und anschließend per E-Mail versendet wird. Da es sich dabei nur um temporäre Dateien handelte, wollte ich dieselbe Liste von Dateien am Ende des Arbeitsablaufs in den Papierkorb verschieben; nun stand diese Liste dann aber nicht mehr zur Verfügung. In anderen Fällen kann es nötig sein den Nutzer während des Arbeitsablaufs um einige Eingaben zu beten; diese werden zwar erst später benötigt, sollten aber aufgrund der folgenden, Zeit-intensiven Aktionen bereits zu Beginn erfragt werden, um seine Aufmerksamkeit nicht erneut zu verlangen. In beiden Fällen wird also ein bereits produziertes Ergebnis zu einem späteren Zeitpunkt (erneut) benötigt. Für eben solche Fälle stellt der Automator zwei Aktionen zur Verfügung⁶ – eine speichert ein Ergebnis in einer Variablen ab (und leitet diese Eingabe als Ausgabe durch) und die andere produziert den in einer Variablen gespeicherten Wert als Ergebnis. Es bleibt also nun lediglich Aufgabe des Endbenutzer, diese beiden Aktionen Zweck-gemäß zu verwenden. In der Benutzeroberfläche der beiden Aktionen hat der Nutzer die Möglichkeit einen Variablennamen einzugeben oder eine bestehende Variable auszuwählen. Diese Eingabe ist meines Erachtens so einfach wie möglich gehalten, so dass auch unerfahrenere Benutzer – haben sie einmal das Variablenkonzept verstanden – die beiden Aktionen „spielend“ leicht in ihren automatisierten Arbeitsabläufen verwenden können. In der Abbildung 8 veranschauliche ich den Einsatz dieser beiden Aktionen in einem beispielhaften Arbeitsablauf.

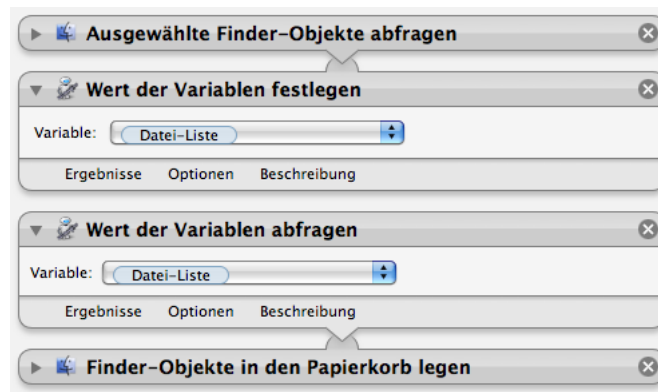


Abb. 8. Mit Variablen können Ergebnisse einer Aktion gespeichert und später wieder abgefragt werden.

⁶ Das Verwenden von Variablen ist erst seit der zweiten Hauptversion des Automators ab Mac OS 10.5 möglich.

2.2.2 Wiederholen von Arbeitsabläufen in einer Schleife Ein anderes und wichtiges Konzept in der imperativen Programmierung ist das wiederholte Ausführen eines Quelltextabschnitts. Übertragen auf den Automator könnte also der Wunsch entstehen, dieselbe Folge von Aktionen mehrere Male nacheinander, gegebenenfalls in bestimmten Zeitabständen, auszuführen. Dabei kann es sich um eine Wartungsaufgabe handeln, bei der regelmäßig der Inhalt eines Ordners überprüft wird. Eine andere Verwendungsmöglichkeit ist die regelmäßige Aufnahme eines Fotos mit der (in vielen Macintosh-Computern eingebauten) Videokamera, welches automatisch im Internet veröffentlicht wird. Das folgende Beispiel zeigt die Verwendung der Aktion *Endlosschleife* in einem Arbeitsablauf, um eine solche *Webcam*-Funktionalität zu programmieren.

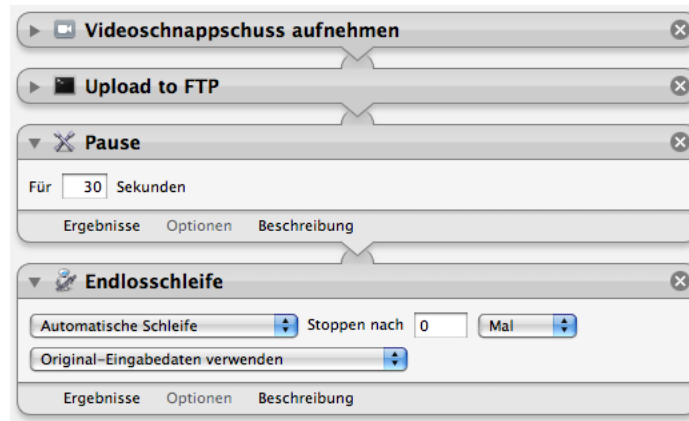


Abb. 9. Mit einer einer Endlosschleife werden alle vorherigen Aktionen wiederholt.

Wird die Aktion *Endlosschleife* in einem Arbeitsablauf platziert, springt der Automator bei die Abwicklung wieder zur ersten Aktion zurück und führt den gesamten Arbeitsablauf bis zum Erreichen der Schleifen-Aktion erneut aus. Dabei handelt es sich allerdings nicht, obwohl der Name es suggeriert, um eine endlose Wiederholung des Arbeitsablaufs. Stattdessen kann der Endnutzer die Anzahl der Wiederholungen auf einen festen Wert beschränken oder eine Zeitspanne vorgeben, in der die Wiederholungen durchgeführt werden. Ebenso besitzt er die Freiheit über die Eingabe zu entscheiden, welche die erste Aktion nach einem Schleifenschritt erhält. So kann er die Originaleingabe verwenden, die beim Start des Arbeitsablaufs zur Verfügung stand oder das Ergebnis der letzten Aktion wiederverwenden. Wie bei der Verwendung von Variablen, ermöglicht der Automator auch hier ein bekanntes Programmierkonzept unter Zuhilfenahme einer Elementaraktion leicht in Arbeitsabläufen zu integrieren.

In Verbindung mit der nun folgend beschriebenen Möglichkeit, bestehende Arbeitsabläufe einzubetten, kann der Endbenutzer auch nur einen Teil und nicht den gesamten Arbeitsablauf wiederholen lassen. Sobald das gesamte Schlei-

fenkonstrukt inklusive eines Abbruchkriteriums in einen zweiten Arbeitsablauf ausgelagert wird, kann der Endbenutzer mehrere solcher Schleifen als *Unterprogramme* in seinen Hauptarbeitsablauf einbetten.

2.2.3 Unterprogramme durch Wiederverwendung von Arbeitsabläufen

Eine Wiederverwendung von Arbeitsabläufen bietet sich nicht nur an, wenn ganze Abschnitte eines Arbeitsablaufs in anderen Arbeitsabläufen wiederverwendet werden sollen, anstatt sie jedes Mal zu kopieren. Auch im Falle von sehr umfangreichen Arbeitsabläufen erscheint es geeignet, einen Teil des Arbeitsablaufs auszulagern. Sie können sich diesen Arbeitsablauf dann als Unterprogramm oder als eine Prozedur im Sinne der prozeduralen Programmierung vorstellen, die von einem anderen Arbeitsablauf aufgerufen wird. Dazu stellt der Automator wieder eine Aktion bereit, um dieses Konzept umzusetzen: Zuerst wird das Unterprogramm als eigenständiger Arbeitsablauf entwickelt und abgespeichert. Dieser kann dann mit Hilfe der Aktion *Arbeitsablauf ausführen* in anderen Arbeitsabläufen aufgerufen werden. Wahlweise kann das Ergebnis des eingebetteten Arbeitsablaufs weiterbenutzt werden (es gleicht damit einem Methodenaufruf) oder das ursprüngliche Ergebnis vor dieser Aktion wird durchgereicht (damit gleicht das Verhalten dem Aufruf eines Unterprogramms). Ein Sonderfall stellt die Möglichkeit dar, den eingebetteten Arbeitsablauf parallel zum eigentlichen Arbeitsablauf auszuführen; damit entfällt verständlicher Weise aber die Verwertung des Ergebnisses. Diese Einstellung wird ebenso wie alle anderen in der Aktion selbst vorgenommen (vgl. Abb. 10).

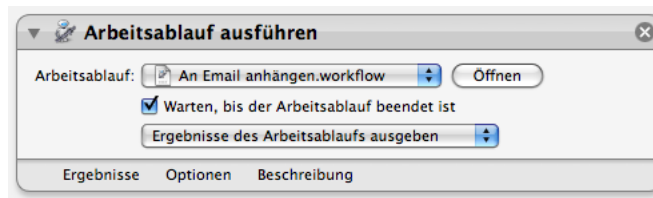


Abb. 10. Diese Aktion ermöglicht das Ausführen eines anderen Arbeitsablaufs.

Beim Ausführen einer solchen Aktion wird das Ergebnis der vorherigen Aktion als Eingabe für den eingebetteten Arbeitsablauf verwendet. Anschließend wird gewartet bis dieser Arbeitsablauf komplett ausgeführt wurde oder sofort mit der nachfolgenden Aktion fortgefahren.

2.2.4 Verwendung von Skript-Sprachen (insbesondere AppleScript)

Die bisherigen Programmierkonzepte, die ich Ihnen vorgestellt habe, werden vom Automator mittels Einsatz spezieller Aktionen ermöglicht. Sequenzen, Variablen, Schleifen und der Aufruf von Unterprogrammen werden durch die Verwendung von entsprechenden Aktionen im Arbeitsablauf realisiert; der Endanwender muss

dazu keine einzige Zeile Programmtext schreiben. Und das ist ja auch gut so, denn schließlich untersuche ich die Tauglichkeit des Automators für Endbenutzer ohne Programmiererfahrung. Wie sieht es aber nun dennoch mit der Unterstützung von anderen, wichtigen Konzepten der Programmierung aus? Was ist, wenn der erfahrenere Endanwender komplexere Kontrollstrukturen verwenden möchte, um beispielsweise eine Wenn-Dann-Aussage zu formulieren? Und scheitert der Endbenutzer vielleicht daran, individuellen Umrechnungen oder Manipulationen von Ergebnissen vorzunehmen um sie für die nachfolgende Aktion aufzubereiten? Eine für diese Bedürfnisse mehr als zufriedenstellende Lösung stellt der Automator wieder einmal mit Hilfe einer Aktion bereit. Dabei handelt es sich um die Aktion *AppleScript ausführen*, die ebenfalls zum Standardreperoire der Automator-Bibliothek gehört. Hier kann der Benutzer das Potential einer vollwertigen Programmiersprache ausschöpfen um auch anspruchsvollere und ausgefallene Aufgaben zu automatisieren.

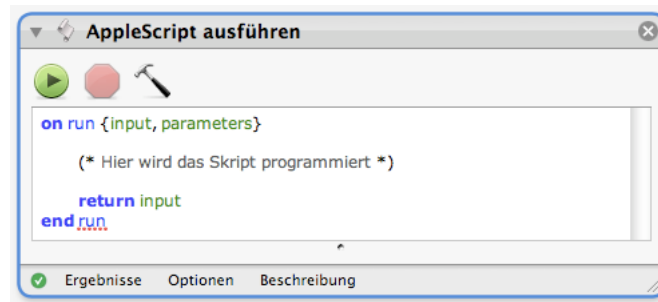


Abb. 11. Mit einer AppleScript-Aktion kann Programmtext ausgeführt werden.

Interessanter Weise ist *AppleScript* für Benutzer mit wenig Programmiererfahrung oder solche Benutzer, die sie sich aneignen wollen, ausgelegt. In der Sprache *AppleScript* formulierte Programmtexte lesen sich wie natürliche, englischsprachige Sätze, weshalb der Einstieg und das Erlernen dieser Programmiersprache vielen Endbenutzern erfahrungsgemäß sehr leicht fällt. Ich möchte Ihnen jetzt aber keine Einführung in diese Sprache geben, sondern lediglich einen Eindruck vermitteln, welches Potential ein Skript in einem Arbeitsablauf entfalten kann.

In der Abbildung 11 sehen Sie die Schablone für ein Skript. Beim Ausführen des Arbeitsablaufs wird die enthaltene Methode `run` mit zwei Parametern aufgerufen, wobei hier nur der erste von Bedeutung ist; er enthält nämlich das Ergebnis von der vorherigen Aktion. Im obigen Beispiel wird dieser mit `input` benannte Wert standardmäßig als Ergebnis der *AppleScript*-Aktion zurückgeliefert. Dieser Programmtext veranlasst also lediglich die Durchreichung der Eingabe als Ergebnis der Aktion; erst der Endbenutzer entscheidet was mit dieser Eingabe geschehen soll.

Ein Beispiel, das hier ohne Verwendung der Eingabe auskommt, ist das Löschen des Papierkorbs mit Hilfe von AppleScript, denn der Automator stellt standardmäßig keine entsprechende Aktion bereit. Um den Papierkorb dennoch zu entleeren, eignet sich folgendes Skript:

```
on run
    tell application "Finder" to empty the trash
end run
```

Listing 1.2. Ein Skript zum Leeren des Papierkorbs.

Dieses Beispiel zeigt bereits, dass mit Hilfe von AppleScript andere Anwendungen ferngesteuert werden können. In diesem Fall wird die Anwendung *Finder* beauftragt den Papierkorb zu leeren. Tatsächlich kann nahezu jede Macintosh-Software dazu gebracht werden, bestimmte Aktionen auszuführen und damit automatisiert zu werden, sei es nur über das Senden von Tastaturbefehlen. Viele Software-Produkte gehen jedoch soweit, dass sie explizite AppleScript-Schnittstellen anbieten. Über einfach verständliche Programmierbefehle kann der Endbenutzer damit wiederkehrende Produkt-bezogene Arbeitsschritte automatisieren, die sonst langwierig mit der Tastatur und der Maus realisiert werden müssten.

Zum Abschluss dieses Abschnitts möchte ich Ihnen demonstrieren, wie mittels eines Skripts von dem Ergebnis der vorherigen Aktion Gebrauch gemacht werden kann. Und zwar soll zufällig ein Element aus einer Liste bestimmt und als Ergebnis des Skripts zurückgegeben werden. Dieser Zufallsgenerator kann beispielsweise in Verbindung mit einer iTunes-Titelsuche verbunden werden, bei der zufällig ein Lied eines bestimmten Interpreten abgespielt werden soll. Ein solcher Zufallsgenerator lässt sich mit dem folgendem Skript realisieren:

```
on run {input, parameters}
    if class of the input is list then
        return some item of the input
    else
        return input
    end if
end run
```

Listing 1.3. Dieses Skript wählt zufällig ein Element aus einer Liste aus.

Neben AppleScript erlaubt der Automator aber auch die Ausführung von Shell-Skripten und mittels einer nachträglich installierbaren Aktion die Ausführung von Python-Code [5]. Meines Erachtens nach eignen sich diese Sprachen jedoch nur wenig für einen Endbenutzer mit keiner oder geringer Programmiererfahrung.

Im nun letzten Kapitel der Ausarbeitung werfe ich noch einen Blick auf die Benutzergemeinde rund um den Apple Automator. Darin teile ich meine Eindrücke mit, wo und inwieweit ein Austausch von Erfahrungen und Arbeitsabläufen zwischen den Endbenutzern stattfindet. Anschließend gelange ich noch zu einer kritischen Einschätzung in der ich Grenzen des Automators aufzeige

und ihn mit anderen Produkten, vor allem mit denjenigen für die Microsoft Windows-Umgebung, vergleiche.

2.3 Die Benutzergemeinde

Wie jedes andere weit verbreitete Software-Produkt, das sowohl von Heimanwendern als auch im beruflichen Umfeld professionell genutzt wird, besitzt auch der Automator eine Benutzergemeinde, die im Internet Erfahrungen austauscht. Wer also einschlägige Suchmaschinen um den Suchbegriff *Apple Automator* bemüht, wird auch hier schnell und zahlreich fündig: Es gibt großzügig angelegte Dokumentationen vom Hersteller ([8], [9]), Blog-Einträge von Endanwendern, die sich mit verschiedenen Aspekten des Automators auseinandergesetzt haben, zahlreiche Foreneinträge und sogar einige größere Portale ([11], [12], [13], [14], [15]), die sich nur dem Apple Automator widmen. Letztere bieten nicht nur vorgefertigte Arbeitsabläufe sondern auch zusätzliche Aktionen aus verschiedenen Bereichen an, die heruntergeladen und installiert werden können. Überraschend wenig wird von den Endanwendern hingegen die Möglichkeit genutzt, sich der englisch-sprachigen Mailing-Liste [16] anzuschließen um dort Fragen zu formulieren. Von dieser Mailing-Liste erhalte ich so selten Antworten und neue Nachrichten, dass ich von deren Benutzung abrate.

2.4 Kritische Betrachtung

Nun habe ich den Apple Automator in dieser Ausarbeitung oft wegen seines gut durchdachten und leicht verständlichen Bedienkonzepts gelobt – um Arbeitsabläufe am PC zu automatisieren werden vordefinierte Teilschritte einfach aneinander gereiht. Die grafisch anschaulich visualisierten Aktionen werden mit der Maus per Drag & Drop angeordnet und formen damit den Arbeitslauf. Bei der Wahl der Teilschritte stehen dem Endbenutzer zahlreiche Aktionen aus verschiedenen Aufgabenbereichen in einer gut sortierten Bibliothek zur Verfügung. Für fast jede häufig verwendete Tätigkeit gibt es eine passende Aktion – vom Kopieren von Dateien, über das Erstellen und Versenden von E-Mails bis hin zum automatischen Import von Bildern in ein Fotoalbum. Für sehr gelungen halte ich es auch, dass die Aktionen ihre Konfiguration direkt in der Oberfläche der Aktion selbst beinhalten. Der Endbenutzer kann die Zuordnung von individuellen Einstellungen zu Aktionen somit sehr leicht nachvollziehen und schnell ändern. Um Platz zu sparen und die Übersicht zu wahren können die Aktionen sogar einfach zusammengeklappt werden. Weiterhin unterstützt der Automator beim Entwurf von Arbeitsabläufen bekannte Programmier-Konzepte: Neben der sequentiellen Aneinanderreihung von Aktionen ist die Einbindung von bereits erstellten Arbeitsabläufen möglich (entspricht dem Aufruf von Prozeduren) und das Wiederholen von Arbeitsschritten bietet die Schleifen-Aktion an. Außerdem ermöglicht der Einsatz von Variablen die mehrfache und spätere Verwendung von produzierten Ergebnissen.

Ohne den Einsatz eines Skripts ist es jedoch nicht möglich weitere Programmier-Konzepte anzuwenden: Arbeitsabläufe können nämlich nicht verzweigen,

denn es werden keine Kontrollkonstrukte wie Fallunterscheidungen unterstützt. Dies könnte Ihnen vielleicht als erste Schwachstelle und Einschränkung für den Endbenutzer vorkommen. So habe ich zuerst auch gedacht, dass die fehlende Unterstützung von wesentlichen Konzepten wie Kontrollstrukturen den Freiheitsgrad des Endbenutzers bei der Gestaltung von Arbeitsabläufen erheblich einschränkt. Dann habe ich mich aber daran erinnert, wofür der Automator eigentlich konzipiert wurde: nämlich für die Automatisierung von alltäglichen Arbeiten am PC. Solche Tätigkeiten führt der Benutzer in der Regel *immer* linear aus; d. h. er beginnt mit einem Teilschritt, führt anschließend den nächsten aus und tut dies so lange, bis er sein Ziel erreicht hat. Es wird selten vorkommen, dass er sich für die selbe Tätigkeit, die er ja zu automatisieren versucht ist, heute einmal so und morgen ganz anders entscheidet. Kommt es dennoch einmal vor, dass der Benutzer alternative Aktionen in seinem Arbeitsablauf einbauen möchte, was ich bisher jedoch noch nie machen wollte, kann er auf die mächtige und leicht zu erlernende Skriptsprache AppleScript zurückgreifen. Sie erlaubt neben der Verwendung von Kontrollstrukturen auch die direkte Ansteuerung von Anwendungssoftware.

Als weitere und letzte Erleichterung, die der Automator mit sich bringt, möchte ich die Installationsmöglichkeiten eines Arbeitsablaufs erwähnen. Abgesehen von der direkten Ausführung können Arbeitsabläufe nämlich als Systemweite Plugins installiert werden, so dass sie aus jeder Lage heraus in der Menüleiste verfügbar sind. Ebenso können Arbeitsabläufe nicht nur in das Kontextmenü des Finders integriert werden, um Datei-basierte Operationen zu automatisieren, sondern auch mit einem Kalender-Ereignis verknüpft werden. Damit können Arbeitsabläufe ohne Zutun des Endbenutzers automatisch zu bestimmten Zeitpunkten, wie z. B. einmal im Monat oder zu bestimmten Anlässen wie Geburtstagen ausgeführt werden. Letztendlich ist es auch möglich Arbeitsabläufe als selbstablaufende Programme zu erstellen um sie an Bekannte weiterzugeben, damit sie dort selbstständig schwierige Aufgaben verrichten, zu der die Bekannten vielleicht nicht in der Lage sind.

Verglichen habe ich den Automator auch mit ähnlichen Software-Produkten, allerdings waren solche nur für die Windows-Plattform zu finden (vermutlich weil der Automator bereits kostenlos mit dem Betriebssystem Mac OS mitgeliefert wird). Beim Vergleich mit Produkten wie *AutoIt!*, *AutoMate*, *Automise*, *RoboTask* und *WinAutomation* bin ich stets zu dem selben Ergebnis erkommen: Entweder sind diese Produkte überschaubar aufgebaut, da sie nur einen sehr geringen Funktionsumfang anbieten oder sie verfolgen ein ähnliches Konzept wie der Automator und bieten einen großen Umfang an elementare Teilaufgaben an, die zum großen Ganzen zusammengesetzt werden können. Dann verlieren sich diese Tools aber sehr schnell in vielen Details, die eine sehr hohe Einarbeitungszeit erfordern; ihre komplexe Benutzerschnittstelle ist für viele Endnutzer zu unübersichtlich und damit ungeeignet.

2.5 Abschlusswort

Während dieser Ausarbeitung habe ich herausgefunden, dass dem Automator als einziges Produkt in seiner Kategorie dieser eben erwähnte Spagat zwischen einfacher Bedienung und universeller und umfassender Verwendbarkeit gelingt. Die Palette der zu automatisieren Tätigkeiten ist sehr groß; in Zukunft wird die Benutzergemeinde aber hoffentlich noch weitere Aktionen programmieren. Die gut durchdachten Bedienkonzepte sorgen dafür, dass in Kürze Automatisierungen entstehen, die dem Benutzer Zeit für kreativere Tätigkeiten ermöglichen. Alles in allem halte ich den Automator deshalb für ein sehr attraktives aber bisher noch nicht viel beachtetes Produkt, das sich nicht nur ungeübte Endanwender mit geringen Programmierkenntnissen anschauen sollten.

Literatur

1. Lars Blumberg: Screenshot „So funktioniert der Apple Automator“
<http://www.lars-blumberg.de/projekte/apple-automator/>.
2. Unix Pipes – powerful and elegant programming paradigm
<http://www.softpanorama.org/Scripting/pipes.shtml>
3. Pipes: A Brief Introduction
<http://www.bellevuelinux.org/pipes.html>
4. Peter Dekkers: Automator Action „Upload to FTP“
http://editkid.com/upload_to_ftp/
5. toxicsoftware: Automator Action „Run Python Script“
<http://toxicsoftware.com/run-python-script/>
6. Kai Surendorf: Das Praxisbuch Mac OS X Tiger – Kapitel 21: AppleScript einsetzen
http://www.galileodesign.de/download/mac/openbook/mac_osx_tiger_21/mac_osx_tiger_21_001.htm
7. Matthew Russel: What Is Automator (and Can It Make Your Life Easier)
<http://www.oreillynet.com/pub/a/mac/2005/09/06/what-is-automator.html>
8. Apple Developer Connection: Guides: Apple Applications / Automator
<http://developer.apple.com/documentation/AppleApplications/Automator-date.html>
9. Apple Developer Connection: Introduction to Automator Programming Guide
<http://developer.apple.com/documentation/AppleApplications/Conceptual/AutomatorConcepts/Automator.html>
10. Apple Developer Connection: Working with Automator
<http://developer.apple.com/macosx/automator.html>
11. Webseite vom Automator-Chefentwickler Sal Soghoian
<http://automator.us/>
12. Automator World
<http://automatorworld.com/>
13. Automator-Actions.com
<http://www.automator-actions.com>
14. MacScripter: Actions/Workflows
<http://www.automatoractions.com>
15. Web Test Automator Actions
<http://www.ottomate.org/>
16. Automator Mailing List
<http://lists.apple.com/mailman/listinfo/automator-users>