Xpert.press

Die Reihe **Xpert.press** vermittelt Professionals in den Bereichen Softwareentwicklung, Internettechnologie und IT-Management aktuell und kompetent relevantes Fachwissen über Technologien und Produkte zur Entwicklung und Anwendung moderner Informationstechnologien.

Achim Zeeck

Speech Application SDK mit ASP.NET

Design und Implementierung sprachgestützter Web-Applikationen

Mit 99 Abbildungen und 19 Tabellen

Springer

Achim Zeeck
sasdk@achimzeeck.de
http://achimzeeck.de

Bibliografische Information der Deutschen Bibliothek
Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

ISSN 1439-5428
ISBN-10 3-540-20872-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-20872-3 Springer Berlin Heidelberg New York

Springer ist ein Unternehmen von Springer Science+Business Media
springer.de

Printed in The Netherlands

Satz und Herstellung: LE-TeX Jelonek, Schmidt & Vöckler GbR, Leipzig
Umschlaggestaltung: KünkelLopka Werbeagentur, Heidelberg
Gedruckt auf säurefreiem Papier 33/3142/YL - 5 4 3 2 1 0

Widmung

Ich widme das Buch meinen Eltern, die mich in all meinen bisherigen Vorhaben in jeder Hinsicht unterstützt haben – auch wenn die meisten Versuche, Ihnen meine Arbeit verständlich darzustellen, meistens scheiterten, da in ihren Augen die Informationstechnologie nach wie vor ein Buch mit sieben Siegeln ist und etwas Magisches und manchmal sogar auch Unheimliches besitzt.

Dem kann ich nur zustimmen.
Aber ich arbeite daran…
Ich danke Euch.

Mein Dank gilt sowohl meinen beiden Lektoren Dr. Frank Schmidt sowie Frau Jutta Maria Fleschutz aus dem Springer-Verlag, die mich in den zurückliegenden Monaten mit der notwendigen Inspiration und Motivation bei Laune gehalten haben, als auch der Produktionsleitung, vertreten durch Herrn Michael Reinfarth. Außerdem danke ich allen Freunden und Bekannten für ihre Anregungen und Ideen.

Vorwort

Als ich mich vor über zwei Jahren dem Thema .NET widmete, war der Grund sehr profan: Die Verheißung, mit diesem neuen und auf absehbare Zukunft sehr viel versprechenden „Tool“ den Spaß am Entwickeln, Ausprobieren und Entdecken wieder zu finden. Viele Unzulänglichkeiten in der Programmierwelt führten dazu, dass mir der frühere Enthusiasmus abhanden kam und ich meinen Schwerpunkt mehr auf die technische Beratung verlagerte.

Insbesondere die Webentwicklung war vor allem mit ISAPI und COM kein Vergnügen – viel besser waren da JSP und Co. Mit ASP.NET jedoch wurde die ganze Sache wieder spannend, und mit der meiner Meinung nach wirklich sehr gelungenen Entwicklungsumgebung Visual Studio war ich bereits vorher vertraut, als ich noch C++ für *die* Sprache schlechthin hielt. Bis die .NET-Sprachen debütierten.

Mit Sprachtechnologien kam ich erst ein Jahr später in Berührung. Ein guter Freund, der eine Zeitlang für eine Firma in dieser Branche arbeitete, erzählte mir von den Möglichkeiten, die sich durch Sprachdialogsysteme eröffneten, zumal die zugrunde liegende Technologie bereits sehr weit fortgeschritten war. Auch er konnte anfangs nicht glauben, dass es funktioniert, und war umso mehr überrascht zu erfahren, welche großen Chancen sich bei einer produktiven Umsetzung boten.

Nicht nur die Kostenersparnis seitens der Kunden, sondern auch die gesteigerte Anwenderzufriedenheit durch rund um die Uhr erreichbare, immer freundliche und sachkompetente Sprachdialogsysteme war auch für mich ein Argument, mich mehr mit dieser Materie zu beschäftigen.

In dem vorliegenden Buch – übrigens dem Ersten seiner Art, und dann auch noch auf Deutsch! – habe ich versucht, so praxisnah wie möglich Ihnen von der Entstehung über die Einbindung bis zur Umsetzung alle Schritte des Speech Application Software Development Kits (SASDK) näher zu bringen. Allerdings setze ich dabei einige Vorkenntnisse voraus, und zwar vor allem deshalb, um nicht den Rahmen des Buches zu sprengen. C# oder eine andere CLR-konforme Sprache wird dabei als bekannt vorausgesetzt, ebenso der Umgang mit ASP.NET und dem Visual Studio .NET. Außerdem gehe ich davon aus, dass auch Kenntnisse im Umgang mit Webtechnologien vorhanden sind. So sollten Sie wissen, was http,

HTML, XML und DOM bedeuten oder wie sich clientseitige Skripte (JScript oder Javascript) erstellen lassen.

Das vorliegende Werk kann aufgrund der hohen Informationsdichte, die durch die umfangreiche Klassenbibliothek und den darin enthaltenen Komponenten, den in ihnen verankerten Typen, Eigenschaften, Aufzählungen usw. resultiert, keinesfalls vollständig all diese Elemente en détail beschreiben. Sie sind also zusätzlich auf die umfangreiche Online-Hilfe angewiesen, was während der Entwicklungsphase zur Beschaffung kontextaffiner Informationen ohnehin notwendig werden wird.

Da – wie alle schnelllebigen (Software-)Technologien – auch das SASDK dem permanenten Entwicklungszyklus unterworfen ist, haben Sie die Möglichkeit, sich auf meiner Webseite http://achimzeeck.de zukünftig über den aktuellen Entwicklungsstand sowie Neuigkeiten aus der Welt des SASDK zu informieren. Auch Errata des Buches werden Sie dort finden.

Sollten Sie Lob oder Kritik üben oder sonstige Fragen sowie Anmerkungen zu diesem Thema loswerden wollen, dürfen Sie diese gerne an die spezielle E-Mail-Adresse sasdk@achimzeeck.de verschicken. Bitte haben Sie aber Verständnis, wenn ich nicht sofort antworte, da auch ich nicht ständig vor dem Rechner sitze. Dennoch werde ich versuchen, möglichst alle eingehenden Anfragen so kompetent und zeitnah wie möglich zu bearbeiten. Bei Fragen zum Buch allgemein oder ähnlichen SASDK-fremden Informationen dürfen Sie sich gerne auch an den Verlag wenden. Die Postanschriften stehen im Impressum, im Internet finden Sie den Verlag unter www.springeronline.de. Dort befindet sich auch der Link zu einem Kontaktformular.

Ich danke Ihnen für den Kauf durch Ihr Interesse an diesem Thema und würde mich freuen, wenn beim Lesen interessante und innovative Ideen herauskommen. Auch darüber dürfen Sie mir gerne berichten!

Hamburg, im März 2005 Achim Zeeck

Inhaltsverzeichnis

Vorbemerkungen

„Wir kommen in friedlicher Absicht!“
In Tim Burtons Film „Mars Attacks!“ von 1996 wird ein vermeintlich freundlich gesinnter Ausspruch des Anführers der auf der Erde gelandeten Marsbewohner von einer Übersetzungsmaschine zwar korrekt wiedergegeben, die xenophobe Intention zur Abwendung der bevorstehenden Katastrophe blieb dem Gerät allerdings verborgen.

Diese Szene zeigt exemplarisch, dass es noch sehr lange dauern (und selbst mit exponentiell steigender Rechenleistung in naher Zukunft Realität) wird, dass Maschinen rabulistische oder ambiguose Aussagen, die ein bestimmtes, meist eigennütziges Ziel verfolgen, korrekt interpretieren oder simulieren können. Dazu bedarf es keiner künstlichen Intelligenz: Selbst Menschen haben damit ja ihre Probleme…

1 Spracherkennung und Sprachsteuerung

Der Wunsch, sich mit einer Maschine wie mit einem Menschen unterhalten zu können, wird bei zunehmender Komplexität der Software und der damit entstehenden Anwendungen immer konkreter. Das hat vorrangig pragmatische Gründe, da ein Anwender keine speziellen Kenntnisse über die Bedienung benötigt, sondern zur Durchführung bestimmter Aufgaben lediglich Befehle in seiner natürlichen (Landes-) Sprache absetzt – verbunden mit der Hoffnung, dass seine Anweisungen auch richtig (akustisch und grammatikalisch) verstanden und entsprechend umgesetzt werden!

Selbst zu Beginn des Computerzeitalters war an eine solche Möglichkeit aus technischer Sicht nicht zu denken, da die geringen Rechnerkapazitäten eine Sprachverarbeitung unmöglich machten. Daher blieb es viele Jahre lang bei dem Wunsch, der insbesondere im Film immer wieder variantenreich artikuliert wurde.[1]

Durch die fortschreitende Miniaturisierung von elektronischen Bauteilen und der zunehmenden Komplexität von Schaltkreisen sowie der parallel damit einhergehenden Zunahme von Speicherkapazität und Prozessorleistung[2] wurde es schließlich möglich, per Spracheingabe mittels fest programmierter Befehle eine Software zu definierten Aktionen zu veranlassen. Diese Software war zunächst nur auf ganz speziellen, leistungsfähigen Rechnern einsetzbar und dementsprechend exklusiv und teuer. Während die Sprachausgabe relativ einfach umsetzbar war, musste für die Spracheingabe und die damit verbundene interne Verwaltung der Signalverarbeitung und -zuordnung ein ungleich größerer Aufwand betrieben werden.

In den folgenden Abschnitten wird skizziert, wie sich die Technologie der Spracherkennung und Sprachsteuerung in den letzten Jahren entwickelt

---

[1] Es gibt zahlreiche Beispiele, in denen Computer „menschliche“ Reaktionen zeigen. Im visionären Film *2001: Odyssee im Weltraum* von Stanley Kubrick aus dem Jahr 1968 beispielsweise beginnt die computergesteuerte Bombe *HAL* mit den Astronauten eine philosophische Diskussion über den Sinn des Daseins und das Für und Wider, zerstört zu werden.

[2] Erwähnenswert ist in diesem Zusammenhang das „Moore'sche Gesetz“ von 1965 (begründet durch den Amerikaner Gordon Moore), welches besagt, dass sich die Anzahl der Transistoren auf einer gegebenen Fläche Silizium etwa alle zwei Jahre verdoppelt. Konservative Annahmen gehen heute sogar von Leistungssteigerungen um mindestens den Faktor 100 oder mehr in den nächsten Jahren aus.

hat, welche Mechanismen es derzeit gibt und wie es um den Reifegrad der aktuellen Spracherweiterungen bestellt ist. Wenn Sie an zusätzlichen Details über diese Thematik interessiert sind, finden Sie im Anhang entsprechende Verweise zu Sekundärliteratur.

1.1 Entwicklung der Spracherkennungssysteme

Die Erforschung und Entwicklung der Spracherkennung (*Automatic Speech Recognition und Transcription, ASR*) geht auf das Jahr 1936 zurück, als in den Bell AT&T Laboratorien für und mit Universitäten an dieser Technologie gearbeitet wurde.

Natürlich war – wie bei vielen Innovationen – das Militär involviert, maßgeblich beteiligt durch die Forschungseinrichtung des Verteidigungsministeriums.[3] Auf der Weltausstellung 1939 stellte das Labor einen ersten Sprachsynthesizer namens *Voder* vor, der mittels Tastatur und Fußpedalen bedient wurde, um die Ausgabe und Geschwindigkeit zu kontrollieren.

Bereits in den frühen 1970er Jahren wurden die Forschungsanstrengungen ausgeweitet, zumal auch namhafte Firmen aus der Computerindustrie wie zum Beispiel IBM und Philips sich daran beteiligten. Im Laufe der Zeit wurden neue Algorithmen entwickelt, die eine Verbesserung in der Erkennung mit sich brachten.

Außerdem begann Anfang der 1980er Jahre das Computerzeitalter, wo die Rechner kompakter, leistungsfähiger und auch für den Heimanwender erschwinglich wurden.

In dieser Periode wurden zahlreiche – auf Sprachtechnologie spezialisierte – neue Unternehmen gegründet, von denen einige auch heute noch am Markt sind statt, andere Unternehmen fusionierten oder drangen ebenfalls in diesen viel versprechenden Markt mit Eigenentwicklungen ein. Microsoft investierte beispielsweise mehrere Millionen Dollar in einen Kooperationsvertrag mit *Lernout & Hauspie*, um deren Spracherkennungstechnologie in eigenen Produkten einsetzen zu können.

Außerdem wurden von Microsoft, das zunehmend stärker expandierte, Unternehmen übernommen, die in diesem Forschungsbereich bereits große Fortschritte vorweisen konnten, wie etwa Entropic, die zu dieser Zeit das weltweit genaueste Spracherkennungssystem entwickelt hatten. In ▶Kapitel 2, *Entwicklungshistorie*, erfahren Sie Näheres über die Entwicklung der Sprachtechnologie bei Microsoft.

Lernout & Hauspie übernahm derweil Dragon Systems für 640 Millionen Dollar. Im Jahre 2001 wurden sie selbst von ScanSoft geschluckt, zwei

[3] http://www.darpa.mil

Jahre später übernahm das Unternehmen auch noch SpeechWorks. Außerdem wurde ein Partnervertrag mit IBM für den Vertrieb der *ViaVoice*-Produkte geschlossen, ScanSoft selbst brachte die Software *Dragon NaturallySpeaking* auf den Markt. Diese beiden Produkte bzw. die darauf basierenden Technologien stellen nach Angaben der Hersteller derzeit die weltweit führenden Spracherkennungssysteme dar.[4]

Mit der Ausdehnung des Internet ab 1990 und der damit verbundenen Möglichkeit, von überall aus auf der Welt auf Daten zugreifen oder Daten verschicken zu können, entwickelten sich neben den bekannten Textauszeichnungssprachen HTML und XML weitere Derivate, um auch die Spracherkennung über den Browser oder das Telefon zu ermöglichen.

Die beiden bislang am weitesten verbreiteten Erweiterungen sind VoiceXML und SALT, über die Sie in den ▶Kapiteln 1.4 und 1.5 mehr erfahren. Daneben existieren nach wie vor die „klassische" Variante des Mensch-Maschine-Dialogs innerhalb eigenständiger, auf dem Einzelplatzsystem ablaufender Programme, sowie die bereits genannte Möglichkeit der Sprachsteuerung über das Telefon. Beide Techniken werden im Folgenden näher betrachtet.

1.2 Desktopbasierte Sprachsteuerung

In diesem Kontext geht es um die Sprachsteuerung in dialogorientierten Anwendungen wie beispielsweise Microsoft *Word* oder dem Tool *RoboScreenCapture*[5], mit denen die Texte und Bildschirmabbildungen (Screenshots) für dieses Buch erstellt wurden.

Die lokale, nicht notwendigerweise vernetzte Anwendung wird dazu auf dem Einzelplatzsystem gestartet, und mittels eines Mikrofons lassen sich Befehle an das Programm verschicken. Zusätzlich ist auch die „herkömmliche" Arbeitsweise mit der Tastatur bzw. der Maus möglich. Anwendungen, die sich anhand verschiedener Techniken steuern lassen, werden auch *multimodale Applikationen* genannt. Nähere Informationen dazu finden Sie in ▶Kapitel 1.4, *Multimodale Sprachsteuerung*.

Der Anwender bedient die Applikation über eine grafische Schnittstelle (*Graphical User Interface, GUI*). Desktopbasierte sprachgesteuerte Anwendungen für das Windows-Betriebssystem nutzen zum Beispiel COM-Schnittstellenaufrufe und setzen auf dem *Microsoft Speech SDK* auf, das diese Schnittstellen zur Verfügung stellt. Diese Applikationen werden mit einem entsprechenden nativen Compiler für die Windows-Plattform er-

[4] http://www.scansoft.de/company/ sowie http://www.scansoft.com/viavoice/
[5] http://www.ehelp.com/products/roboscreencapture/

stellt – die gesamte Sprachsteuerung ist also in der binären Einzelplatzanwendung gekapselt. Weitere Informationen zu COM finden Sie auch in ▶Kapitel 2.1, *Von COM zu .NET*.

Ähnlich wie ein Wörterbuch können sprachgesteuerte Applikationen durch neue Einträge – in diesem Fall Aktionen – erweitert werden. Welche Einsatzmöglichkeiten sich dabei ergeben, wird in ▶Kapitel 3, *Überblick über die Einsatzmöglichkeiten*, detaillierter erläutert.

1.3 Webbasierte Sprachsteuerung

Während bei einer desktopbasierten sprachgestützten Anwendung der Client eine modale (Standalone-)Applikation ist, stellt bei der webbasierten Sprachsteuerung der Browser den Client dar. Modale Dialoge sind zwar auch mit einem Browser – etwa durch Einsatz von Scriptsprachen bei der Datenvalidierung – möglich, normalerweise findet ein Feedback aber immer über zustandslose http-GET- oder POST-Aufrufe statt.

Dieser Aufwand, der betrieben werden muss, um bestimmte Ergebnisse zu erhalten, hat zwar negative Auswirkungen auf die Gesamtperformance, er wird jedoch durch wesentlich mehr Flexibilität wieder wettgemacht. Durch weiter steigende, flächendeckend hohe Bandbreiten wird kurzfristig auch der Geschwindigkeitsnachteil obsolet werden. Ein großer Vorteil ist auch, dass sich bereits fertig entwickelte Webapplikationen mit vertretbarem Aufwand durch Sprachunterstützung aufwerten lassen.

Auch das Problemthema „Sicherheit“ ist bei webbasierten Anwendungen mittlerweile durch Protokolle wie HTTPS oder SSL im Griff, darüber hinaus lässt sich über PIN- und TAN-Kennziffern und Firewalls der Datenschutz weiter ausbauen. Dafür ist eine nahtlose Zusammenarbeit sowohl auf der Entwicklungs- als auch auf der Administratorenseite unumgänglich.

Webapplikationen müssen dem Anwender nicht notwendigerweise eine grafische Schnittstelle zur Verfügung stellen, sondern können ebenso über einen Webserver autarke Telefonsysteme repräsentieren, die mittels Sprache (*Voice-only*) oder Tastencodes (*Dual-Tone Multi-Frequency, DTMF*) bedienbar sind. Mehr zu diesen Themen finden Sie in ▶Kapitel 3, *Überblick über die Einsatzmöglichkeiten*.

1.4 Erweiterungen von Webseiten durch SALT

Neben der VoiceXML-Sprache existiert seit 1992 ein weiteres, von Microsoft als eines der Gründungsmitglieder zur Umsetzung der Sprachtechno-

logie in eigenen Systemen präferiertes Konzept: *Speech Application Language Tags* (SALT).

Wie der Name vermuten lässt, handelt es sich hierbei um eine Spracherweiterung für (X)HTML in Form von zusätzlichen Strukturelementen und Attributen (Tags), um Webseiten mit Sprachfunktionalität auszustatten. Neben diesen XML-Elementen nutzt SALT Eigenschaften, Ereignisse und Methoden des *Document Object Model* (DOM).

SALT erlaubt den Zugriff über multimodale oder Telefonieapplikationen auf Anwendungen, Informationen oder auch Web Services mit nahezu allen vorhandenen Devices: PCs, Tablet PCs, PDAs, Smartphones oder Telefone. Mit Ausnahme des Telefons wird bei allen anderen Geräten selbstverständlich der Einsatz eines Mikrofons für die verbale Kommunikation vorausgesetzt.

Die SALT-Spezifikation wurde am 15. Juli 2002 veröffentlicht und am 13. August 2002 in der Version 1.0 zur Verabschiedung dem *World Wide Web Consortium* (W3C) vorgelegt. Über 70 Unternehmen – darunter namhafte wie Intel, Cisco, Philips, Samsung, ScanSoft oder SpeechWorks – haben sich bereits diesem Forum[6] angeschlossen bzw. fungieren als Initiatoren. Einige von ihnen bieten eigene SALT-Browser an, andere wiederum unterstützen diese Technologie durch Frameworks oder Entwicklungsumgebungen, allen voran Microsoft.

In diesem Kontext ist die .NET-Plattform eine ideale Basis, die auch von vielen SALT-Partnern unterstützt wird. Microsoft ist hier die treibende Kraft und bietet zudem durch seine große Marktpräsenz ein hohes Potential.

Ende März 2003 entschied sich das aus sechs Gründungsmitgliedern bestehende Board of Directors für eine offenere Struktur, in der Sponsoren oder Unternehmen, die das Industrieforum aktiv unterstützen, weitergehende Mitspracherechte eingeräumt werden, um die Attraktivität an einer Mitarbeit zu steigern und neue Interessenten zu gewinnen.

Damit Sprache als Eingabemedium genutzt werden kann, reicht SALT im HTML-Kontext aber nicht aus. Der eingesetzte Browser benötigt Unterstützung durch entsprechende Erweiterungen: Damit zum Beispiel der (Pocket) Internet Explorer „SALT-compliant“ wird, muss ein entsprechendes *Speech Add-in* installiert werden, und für multimodale Applikationen wird im http-Header ein Substring namens HTTP_USER_AGENT mit Modul- und Versionsangaben benötigt.

Der Webserver (bei Microsoft ist dies der *Internet Information Server, IIS*) muss in der Lage sein, eingebettete (proprietäre) Speech Controls in SALT-Code zu rendern. Sie finden detaillierte Erläuterungen dazu in Kapitel 6, *Sprachverarbeitung zur Laufzeit*.

---

[6] www.saltforum.org

Der Einsatz des .NET-Frameworks bietet auch unter dem Gesichtspunkt *Mobility* interessante Einsatzmöglichkeiten, da es für die sich verstärkt im Markt etablierende Generation der Pocket-PCs und Smartphones in Verbindung mit webbasierter Spracheingabe ideale Voraussetzungen bietet. Microsoft fasst diese Geräte unter dem Begriff „Smart Devices" zusammen. In ▶Kapitel 3, *Überblick über die Einsatzmöglichkeiten*, wird darauf näher eingegangen.

Das installierte Windows Mobile 2003 Betriebssystem ermöglicht zusammen mit dem .NET Compact Framework eine Portierung bestehender multimodaler Webapplikationen (mit entsprechender Anpassung), zumal das SASDK beide Frameworks unterstützt.

1.5 Vergleichende Betrachtung von SALT und VoiceXML

SALT und VoiceXML[7] verfolgen – sehr vereinfacht gesagt – im Kern dasselbe Ziel: Mit Hilfe der menschlichen Stimme Anwendungen zu steuern. Während es sich bei VoiceXML um eine vollwertige Applikationssprache handelt, besteht SALT „nur" aus ein paar XML-Tags, die in (X)HTML-Seiten eingebettet werden (das ist natürlich nicht alles und wurde im vorherigen Kapitel bereits angedeutet. Spätestens nach der Lektüre dieses Buches werden Sie dies bestätigen!)

Abgesehen davon, dass VoiceXML seit 1994 existiert und daher etwas später entwickelt wurde als SALT, liegen die Unterschiede eher in der Ausrichtung – daher stehen beide Konzepte auch nicht in direkter Konkurrenz zueinander, sondern ergänzen sich, da jedes seine spezifischen Schwerpunkte setzt.

Obwohl es in diesem Buch um das SASDK und daher auch um SALT geht, möchte ich Ihnen dennoch zum Abschluss dieses Kapitels kurz die auffälligsten Unterschiede und Gemeinsamkeiten beider Konzepte – ohne Anspruch auf Vollständigkeit – vorstellen.[8]

Gemeinsamkeiten

- Unterstützung von Telefonieapplikationen (Voice-only und DTMF)
- Wiedergabe von Text durch synthetisches Audio und Audiodateien
- Erweiterung von HTML-, XHTML- und XML-Seiten durch Tags

---

[7] http://www.voicexml.org

[8] Im Anhang finden Sie auch Literaturangaben über VoiceXML-Bücher.

Unterschiede

SALT

- Über 70 Mitgliedsunternehmen (maßgeblich Microsoft)
- Vorlage zur W3C-Spezifikation, Version 1.0
- Fokus liegt auf multimodalen Webapplikationen

VoiceXML

- Über 600 Mitgliedsunternehmen
- Seit 03.02.2004 W3C Proposed Recommendation Status (Version 2.0)
- Fokus liegt auf telefongesteuerten Anwendungen
- Eigene Sprache (XML-Derivat) mit eigener DTD
- Unterstützt alle aktuellen Browser, plattformunabhängig

Teil 1
Das Speech Application SDK (SASDK)

2 Entwicklungshistorie

2.1 Von COM zu .NET

Um den zunehmend gestiegenen Ansprüchen der Anwender nach zusätzlicher oder erweiterter Funktionalität in Softwaresystemen nachzukommen, dabei aber die bisher erlernten Bedienungsschritte in den Anwendungen weitestgehend beizubehalten, kamen bei der Programmierung bereits frühzeitig Klassenbibliotheken zum Einsatz.

Diese Bibliotheken – häufig werden auch alternativ die englischen Begriffe *Framework*, *Application Programming Interface (API)*[1] oder *Software Development Kit (SDK)*[2] verwendet – sind insbesondere hinsichtlich der Wiederverwendung von bereits erstellten (und getesteten!) Klassen und Modulen hilfreich, da sie eine relativ einfach zu handhabende abstrakte Schicht über die hardwarenahen Implementierungsdetails legen.

Als eine der schnellsten und flexibelsten Programmiersprachen auf dem Markt gilt auch derzeit noch die fast durchgängig objektorientierte Programmiersprache C++. Microsoft brachte früh eine eigene Klassenbibliothek namens *Microsoft Foundation Classes (MFC)* speziell für diese Sprache heraus, die sich bis heute in diesem Bereich als Quasi-Standard etabliert hat, denn auch andere Anbieter wie Borland oder Watcom setzten darauf. Die darin enthaltenen Klassen und Strukturen konnten sowohl für lokale (allein stehende) als auch vernetzte (webbasierte) Softwareprojekte eingesetzt werden, zum Beispiel durch Nutzung der *Internet Server API (ISAPI)*.

Um nun aber auch Programmierern aus anderen „Lagern“ wie zum Beispiel Visual Basic oder Delphi den Zugriff auf definierte Schnittstellen zu ermöglichen, bedurfte es eines sprach- und systemübergreifenden Ansatzes. Microsoft schuf daher mit dem *Component Object Model (COM)*[3] einen Standard für die binäre Repräsentation von Objekten, der die objekt-

---

[1] Manchmal auch bezeichnet als *Application Programming Infrastructure*.

[2] In einem SDK sind neben der eigentlichen Klassenbibliothek normalerweise noch weitere Tools enthalten.

[3] Zu Beginn wurde COM noch unter dem Begriff *Common Object Model* geführt.

orientierte Programmierung mit gemischtsprachlichen Architekturen ermöglicht.

Letztendlich war und ist aber die Schnittstellenprogrammierung mit COM und der damit verbundenen fehleranfälligen Zeigerarithmetik (vor allem bei C++) relativ schwierig zu handhaben. Es braucht nicht nur große Erfahrung, sondern führt insbesondere bei der teamorientierten Softwareentwicklung immer dann zu Problemen, wenn Teammitglieder ausscheiden und ihr Wissen mitnehmen anstatt anstatt, ganz besonders, wenn die Software nicht ausreichend dokumentiert ist.

Um diesen technischen und defätistischen Problemen zu begegnen, entwickelte Microsoft eine vollständig neue Klassenbibliothek namens *.NET*, und dazu passend eine Reihe neuer Programmiersprachen wie zum Beispiel *C#*[4], *Visual Basic.NET, JScript.NET* oder *Managed C++*. Der große Vorteil lag nun unter anderem darin, dass es nur noch *ein* Framework für *alle* Sprachen gab und damit die Möglichkeit geschaffen wurde, Projekte mit gemischten Entwicklungsteams führen und gleichzeitig auf die Vorlieben der Programmierer Rücksicht nehmen zu können.

Mittlerweile gibt es eine Vielzahl von Programmiersprachen, die .NET unterstützen, darunter sogar „altehrwürdige“ Vertreter wie COBOL und Fortran, aber auch moderne wie Java oder Delphi sowie eine Vielzahl von Scriptsprachen.[5]

Das .NET-Framework verfolgt ein völlig neues Paradigma: Es stellt nicht nur eine Klassenbibliothek dar, sondern besteht aus einer Vielzahl voneinander abhängiger und aufeinander aufbauender effizienter Dienste, die erst im Zusammenspiel miteinander die Leistungsfähigkeit ausmachen. Bereits im Vorwort des Buches wurde darauf hingewiesen, dass das Framework beim Leser als weitestgehend bekannt vorausgesetzt wird, daher werde ich nicht näher auf einzelne Details wie CLR, CLS oder CTS eingehen.

Im Anhang finden Sie eine Liste empfehlenswerter deutsch- und englischsprachiger Literatur, die über alle Sprachelemente und Techniken erschöpfend Auskunft geben.

Es sei noch darauf hingewiesen, dass Microsoft intensiv an einer neuen Version 2.0 arbeitet, die derzeit noch den Codenamen „Whidbey“ trägt und voraussichtlich Mitte 2005 auf den Markt kommen soll. Darin wird es

---

[4] Sprich: „C sharp“. Das Zeichen # stammt aus der Musik und bedeutet die Erhöhung einer Note um einen halben Ton. Dadurch sollte – ähnlich wie bei C++ das Doppelplus – die Verbesserung der Sprache C verdeutlicht werden.

[5] Informationen über die .NET-Sprachen von Microsoft finden Sie unter http://msdn.microsoft.com/vstudio/productinfo/whitepapers. Eine umfangreiche Liste mit Sprachen, die das .NET-Framework unterstützen, gibt es unter http://www.jasonbock.net/programming.html (auf Englisch).

eine Vielzahl von Verbesserungen und Erweiterungen insbesondere für den Webentwickler geben, aber auch die Entwicklungsumgebung *Visual Studio .NET* sowie die Sprache C# wird davon profitieren. Details dazu finden Sie ebenfalls auf diversen Microsoft-Seiten im Internet.

2.2 Die Entwicklung des SASDK

Während Microsoft intensive Erfahrungen mit der Sprachtechnologie sammelte, entstand parallel dazu eine COM-basierte Klassenbibliothek, mit der sich beispielsweise unter Visual C++ und Einsatz der MFC dialogorientierte sprachgesteuerte Anwendungen erstellen ließen. Diese Bibliothek hieß schlicht *Speech SDK* und wird seitdem – unregelmäßig aktualisiert – als kostenloser Download ins Internet gestellt.

Die älteste derzeit noch erhältliche Bibliothek in der Version 4.0a steht nur noch aus Gründen der Abwärtskompatibilität zur Verfügung, und basiert auf frühen Spracherkennern wie *Whistler* und *Whisper*. Eine noch ältere Version steht nur noch Kunden zur Verfügung, die Zugang zum Microsoft Developer Network (MSDN) besitzen.

Die derzeitige Fassung trägt die Versionsnummer 5.1 und ist in der Lage, mit der *Win32 Speech API (SAPI)* umzugehen, die *Windows Automation* unterstützt. SAPI ist eine Middleware, gegen die sowohl Anwender des SDK als auch Anbieter von Speech Engines programmieren. Von Microsoft werden dem SDK frei distributierbare *Text-to-Speech-Engines (TTS)* sowie *Speech Recognition Engines* in den Sprachen „U.S. English“, „Simplified Chinese“ und „Japanese“[6] zur Verfügung gestellt.

Kurz nach Einführung des .NET-Frameworks wurde auch eine .NET-Variante des Speech SDK entwickelt. Am 7. Mai 2002 kündigte Microsoft auf der AVIOS Speech Expo in San Jose (Kalifornien) die Version 1.0 Beta 1 an, das als bislang erstes und einziges Webentwicklungstool auf den *Speech Application Language Tags (SALT)* aufsetzt. Nähere Informationen zu dieser Spezifikation finden Sie in Kapitel 6.1.

Seit der Beta 3 heißt diese Bibliothek nun offiziell Microsoft Speech Application SDK (SASDK). In Tabelle 1 sind – soweit mir entsprechende Informationen vorlagen – alle veröffentlichten Versionen sowie deren Erscheinungstermine aufgeführt. Die Angaben basieren auf Daten von den Microsoft-Seiten sowie aus den Informationen des Microsoft Developer Network (MSDN). Da die ersten beiden Ausgaben des Speech SDK unter .NET durch neuere ersetzt wurden, sind sie (leider) nicht mehr verfügbar.

---

[6] Nähere Details und Angaben zum Download der aktuellsten Bibliotheken finden Sie auch unter http://www.microsoft.com/speech/download/.

Tabelle 1. Versionsverlauf des Speech Application SDK

Bezeichnung	Version	Datum	Anmerkung
Speech SDK (SAPI)	3.0	15.05.1998	Nur MSDN
	4.0a	1999	
	5.0	12.01.2001	Nur MSDN
	5.1	März 2003	
.NET Speech SDK	1.0 Beta 1	Mai 2002	n. mehr erhältl.
	1.0 Beta 2	Okt. 2002	n. mehr erhältl.
.NET Speech Application SDK	1.0 Beta 3	Aug. 2003	n. mehr erhältl.
	1.0 Beta 4	Nov. 2003	n. mehr erhältl.
	1.0 Final	7. 5.2004	
	1.1 Beta	31.12.2004	

Bevor ich Sie kurz mit dem Speech Server und den Telephony Services vertraut mache, sollten der Vollständigkeit halber noch ein paar Begriffe erklärt werden, die in den vorherigen Abschnitten bereits erwähnt wurden.

Windows Automation

Mit Automation, Windows Automation oder *OLE Automation* wird die Fähigkeit eines Objekts (oder einer anderen Entität) bezeichnet, mit einem anderen Objekt zu kommunizieren, genauer gesagt, einen Datenaustausch durchzuführen oder den Aufruf von Schnittstellen zu gestatten. Die Einheiten selbst sind entweder Anwendungen oder so genannte ActiveX-Controls – wesentlich ist, das alle Kommunikationspartner COM unterstützen. So ist es zum Beispiel in einer Windows-Umgebung (und nur da) möglich, in Word die Funktionalität des Internet Explorers zu nutzen, oder in Excel die Textfunktionen von Word.

Durch Automation lassen sich *Dynamic Link Libraries (DLL)* aufrufen, wie etwa SAPI. Diese Bibliothek stellt per se kein Interface zur Verfügung, bietet also keine visuellen Darstellungsmöglichkeiten an. Der Einsatz von DLLs erfordert kein Neukompilieren des sie benutzenden Programmcodes, da diese Bibliotheken beim Programmstart automatisch geladen werden und sofort zur Verfügung stehen.

Automation als komplexe Technologie beherbergt Windows-spezifische Modelle, die unter den Begriffen ActiveX, COM und COM+ bekannt wurden, und geht auf *Object Linking and Embedding (OLE)* zurück. Ähnlich wie .NET ist die Automation-Programmierung sprachunabhängig, da sowohl Visual Basic, C++, JScript oder irgendeine andere COM-unter-

stützende Programmiersprache von Fremdanbietern eingesetzt werden kann, wobei hier C++ als „unhandlichste“ Variante auftritt.[7]

Bevor die Beschreibung der Speech Engines folgt, empfehle ich einen Blick auf die Struktur der SAPI-Layer, um zu verdeutlichen, wie die Kommunikationsstränge verlaufen. Für jede Richtung, also für die Sprachеingabe sowie die Sprachausgabe, stehen COM-Schnittstellen zur Verfügung. Darüber hinaus lassen sich unabhängig von einer Sprach-Engine mittels der XML-Schemadefinition „sapi.xsd“ Parameter wie Geschwindigkeit oder Lautstärke beeinflussen. Das *Device Driver Interface (DDI)* wird von Speech-Engine-Anbietern genutzt, um die hardwarenahen Ausprägungen zu programmieren.

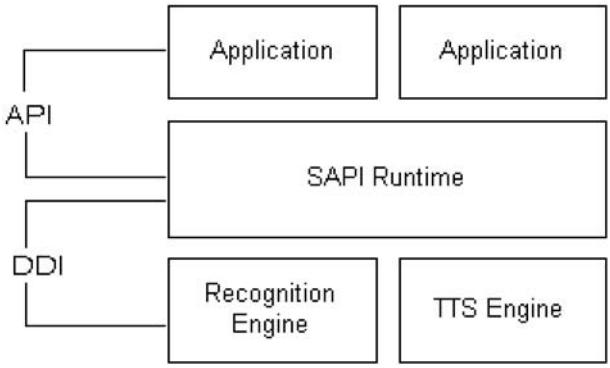

Abb. 2.1. Schematischer Aufbau der SAPI-Layer

Text-to-Speech Engines (TTS)

Der erste Typ der SAPI-Engine unterstützt die computergesteuerte Sprachausgabe, die auch unter dem Begriff Text-to-Speech-Engine oder kurz TTS geführt wird. Dieser zusammen mit dem Betriebssystem installierte Systemtreiber führt im einfachsten Fall die Wiedergabe von Texten in Form von Strings mit einer synthetischen Stimme aus. Das Stimmprofil lässt sich dabei zwischen drei voreingestellten Avataren wählen (siehe auch Kapitel 9.1, *Künstliche Wiedergabe*), die auch separat von den Microsoft-Seiten heruntergeladen werden können.

Es gibt weitere TTS-Engines von Drittanbietern, die unter anderem für spezielle Einsatzbereiche in der Ausgabe optimiert sind, wie etwa Medizin oder Technik, außerdem existieren diverse Sprachanpassungen zum Beispiel für Deutsch oder British English. TTS-Engines sollten normalerweise

[7] Das gilt natürlich nicht für diejenigen Entwickler unter Ihnen, die C++ mit der Muttermilch aufgesogen haben und daher virtuos mit Referenzzählern und Zeigerarithmetik umzugehen verstehen!

in der Lage sein, die zwei wesentlichen Methoden zur Sprachsynthese umzusetzen: *Formantsynthese* und *Konkatenative Synthese*[8].

Bei der Formantsynthese generiert die TTS eine künstliche Sprachausgabe, die auf linguistischen Rollen und Modellen basiert. Dabei wird durch den Einsatz diverser Filter die Natürlichkeit der menschlichen Stimme, die durch Mund- und Lippenform, Gaumen, Rachen, Kehlkopf und andere orale Merkmale geprägt ist, zu simulieren versucht.

Die digitale Ausgabe ist bereits sehr authentisch, dennoch unverkennbar als Computerstimme zu identifizieren. Die Berechnungen erfolgen anhand von aufeinander aufbauenden Wahrscheinlichkeitswerten (genauer: Erkennungsraten), die auch während der Ablaufphase mittels SML angezeigt werden und sich zudem manuell konfigurieren lassen. Nähere Auskünfte erhalten Sie sowohl in ▶Kapitel 5, *Speech Controls im Visual Studio .NET*, als auch in ▶Kapitel 9.4, *Die Semantic Markup Language*.

Bei der konkatenativen Synthese wird einen anderer, pragmatischer Weg verfolgt: Da nichts so echt ist wie die menschliche Stimme selbst, kommen bei diesem Verfahren Audiodateien zum Einsatz, die Worte, Sätze oder Satzfragmente beinhalten. Durch Aneinanderreihung (engl. *to concatenate*: verknüpfen) dieser Audioelemente lassen sich dann mit einiger Akribie neue Wörter und Sätze zusammenstellen. Dieses Verfahren ist entsprechend aufwändig und – sofern professionelle Sprecher daran beteiligt sind – auch verhältnismäßig teuer. Die Ausgabequalität ist jedoch hervorragend, vor allem wenn sie hochwertig produziert wird.

Der Einsatz konkatenativer Synthese ist aber nicht in allen Fällen sinnvoll. Zum Beispiel ist er nicht praktikabel, wenn es um die Ausgabe unvorhersehbarer Texte geht, wie etwa beim Vorlesen von E-Mails oder bei der dynamischen Generierung von Texten (Strings) zur Laufzeit.

Mehr Details zur Vorgehensweise bei der Programmierung mit dem SASDK finden Sie auch in ▶Kapitel 9.2, *Natürlichsprachliche Wiedergabe*. In Telefonie-Anwendungen sind wegen der relativ geringen Bandbreite komprimierte und auf 44,1 KHz gesampelte Monoaufnahmen in 16 Bit jedoch völlig ausreichend. Weitere Informationen dazu gibt es in Kapitel 3.2, *Voice-only Applikationen*.

Speech Recognition Engine (SR)

Spracherkenner oder *Speech Recognizer* wandeln die menschliche Stimme, die über ein Mikrofon oder Telefon an die Applikation geschickt wird, wieder in Strings um. Damit ist die Speech Recognition Engine das Gegenstück zur Text-to-Speech-Engine, mit dem Unterschied, dass die Ein-

[8] Es existiert noch die Artikulatorische Synthese, die aber wegen ihres hohen Rechenbedarfs und der bislang rückständigen Forschung hier keine Rolle spielt.

gabe immer natürlich (also durch eine Person) erfolgt, die Ausgabe wahlweise natürlich (als Audio) oder synthetisch.

Bei der Spracherkennung ist darüber hinaus noch eine Unterscheidung zwischen *aktiver* und *passiver Reaktion* zu treffen. Bei der aktiven Umsetzung der gesprochenen Worte auf dem Zielsystem kann eine Steuerung des Empfängers ermöglicht werden, da die umgewandelten Strings als Befehle interpretiert werden. Dadurch ist ein Dialog möglich, um zum Beispiel Überweisungstransaktionen durchzuführen.

Die passive Umsetzung führt lediglich zur Ausgabe der gesprochenen Worte, zum Beispiel in Form von Texten auf dem Bildschirm. Hier ist als Einsatz ein Schreib- oder Diktierprogramm vorstellbar. Weitere Beispiele für konkrete Einsatzzwecke – die keinesfalls den Anspruch auf Vollständigkeit erheben – finden Sie in Kapitel 3.

Bei der Spracherkennung ist eine Audioausgabe als Bestätigung – egal ob natürlich oder künstlich – nicht zwingend notwendig („stille Umsetzung“). So verarbeiten viele Programme lediglich die Spracheingabe, und die Ausgabe von Informationen erfolgt entweder über Bildschirmmitteilungen, oder es werden gesprochene Befehle in Aktionen transformiert, wie beispielsweise in Microsoft Office (Word und Powerpoint)[9] oder bei der Software „RoboScreenCapture“. Gerade hier ist es sehr sinnvoll, um unnötige Tastatur- anstatt oder Mauseingaben zu vermeiden.

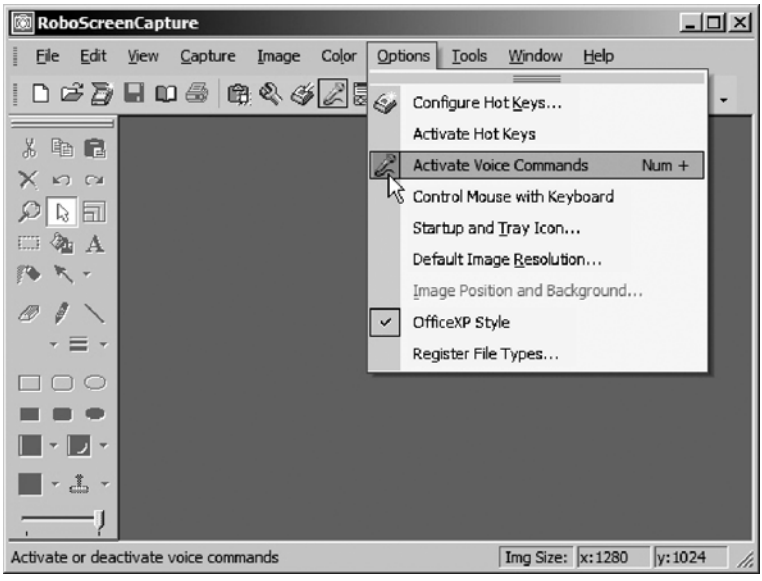

Abb. 2.2. Die Sprachsteuerung bei RoboScreenCapture basiert auf SAPI

[9] Dies gilt allerdings bislang nur für die Versionen in USA, Japan und China.

2.3 Microsoft Speech Server (SES, TAS und TIM)

Im Frühjahr 2003 wurde von Microsoft eine umfassende Sprach- und Telefonieinfrastruktur angekündigt, die auf ursprünglich zwei aufeinander abgestimmten Serversystemen basieren sollte: Speech Server und Telephony Server. Die Funktionalität des Telephony Server ist mittlerweile im Speech Server integriert, so dass es in der derzeitigen Fassung nur noch den *Microsoft Speech Server* (MSS) als Entwicklungs-, Test- und Managementplattform gibt.[10]

Der MSS stellt sich als vollständiges *Interactive Voice Response-System* (IVR) dar, welches unter Einsatz des SASDK als Unternehmenslösung für Telephony (DTMF)-, Voice-only- und Multimodal-Applikationen designed wurde. Der Fokus liegt dabei auf Hochlastverfügbarkeit, zum Beispiel für Callcenter, Buchungssysteme oder als flexible (Voice-) Kommunikationsplattform für Mitarbeiter. Betriebssystemvoraussetzung für den Einsatz des Speech Server (2004) ist die englische Fassung des Windows Server 2003 sowie das .NET Framework 1.1.

Im Kern besteht der MSS aus zwei Komponenten: den *Speech Engines Services* (SES) sowie den *Telephony Application Services* (TAS). Während die SES für die Spracherkennung und -wiedergabe zuständig sind, kontrollieren die TAS in Verbindung mit dem *Telephony Information Manager* (TIM) die Verbindungen vom und zum Anwender, wenn dieser über ein Telefon mit dem Speech Server in Verbindung tritt. Der TIM ist bewusst als flexible Lösung mit Schnittstellen zu Drittanbietersystemen entwickelt worden.

Die **SES** ermöglichen die Verbindung von Remote Clients über TCP/IP und SOAP (zum Beispiel via Pocket PC). Dabei wird durch *Engine pooling* die simultane Abarbeitung mehrfacher, gleichzeitiger Sprachanfragen geregelt, und zwar sowohl intern als auch extern, also für Spracherkennung und -ausgabe. Grammatiken, Audiodateien oder Promptdaten(banken) werden im Cache vorgehalten und erhöhen damit die Gesamperformance, da auf diese Elemente permanent zugegriffen werden muss. Außerdem unterstützen die SES Standardmechanismen eines Servers wie Verwaltung, Monitoring und Logging. Innerhalb der SES werden *SR-, TTS- und DTMF-Engine Instances* verwaltet.

Die **TAS** stellen die Schnittstelle vom Telefon[11] oder Handy zum SAS her. Auch die TAS ermöglichen das Management simultaner Anfragen

---

[10] Derzeit ist der Microsoft Speech Server leider nur in den USA erhältlich. Aktuellste Informationen finden Sie unter http://www.microsoft.com/speech.

[11] Mit „Telefon“ sind schnurgebundene Telefone (auch im Auto) gemeint.