application. It also has an acoustic component reflected in the voice models the speech engine uses during recognition. These voice models can be either unique per speaker or speaker-independent.

Figure 1 shows the resources the ViaVoice speech engine uses during the recognition process (http://www-3.ibm.com/software/speech/dev/sdk_windows.html). The domain-specific resources, such as the vocabulary, can vary dynamically during a given recognition session. A dictation application can transcribe spoken input directly into the document's text content, a transaction application can facilitate a dialog leading to a transaction, and a multimedia indexing application can generate words as index terms.

In terms of application development, speech engines typically offer a combination of programmable APIs and tools to create and define vocabularies and pronunciations for the words they contain. A dictation or multimedia indexing application may use a predefined large vocabulary of 100,000 words or so, while a transactional application may use a smaller, task-specific vocabulary of a few hundred words.

Although adequate for some applications, smaller vocabularies pose usability limitations by requiring strict enumeration of the phrases the system can recognize at any given state in the application. To overcome this limitation, transactional applications define speech grammars for specific tasks. These grammars provide an extension of the single words or simple phrases a vocabulary supports. They form a structured collection of words and phrases bound together by rules that define the set of speech streams the speech engine can recognize at a given time. For example, developers can define a grammar that permits flexible ways of speaking a date, a dollar amount, or a number. Prompts that cue users on what they can say next are an important aspect of defining and using grammars. It turns out that speech grammars are a critical component of enabling the Voice Web.

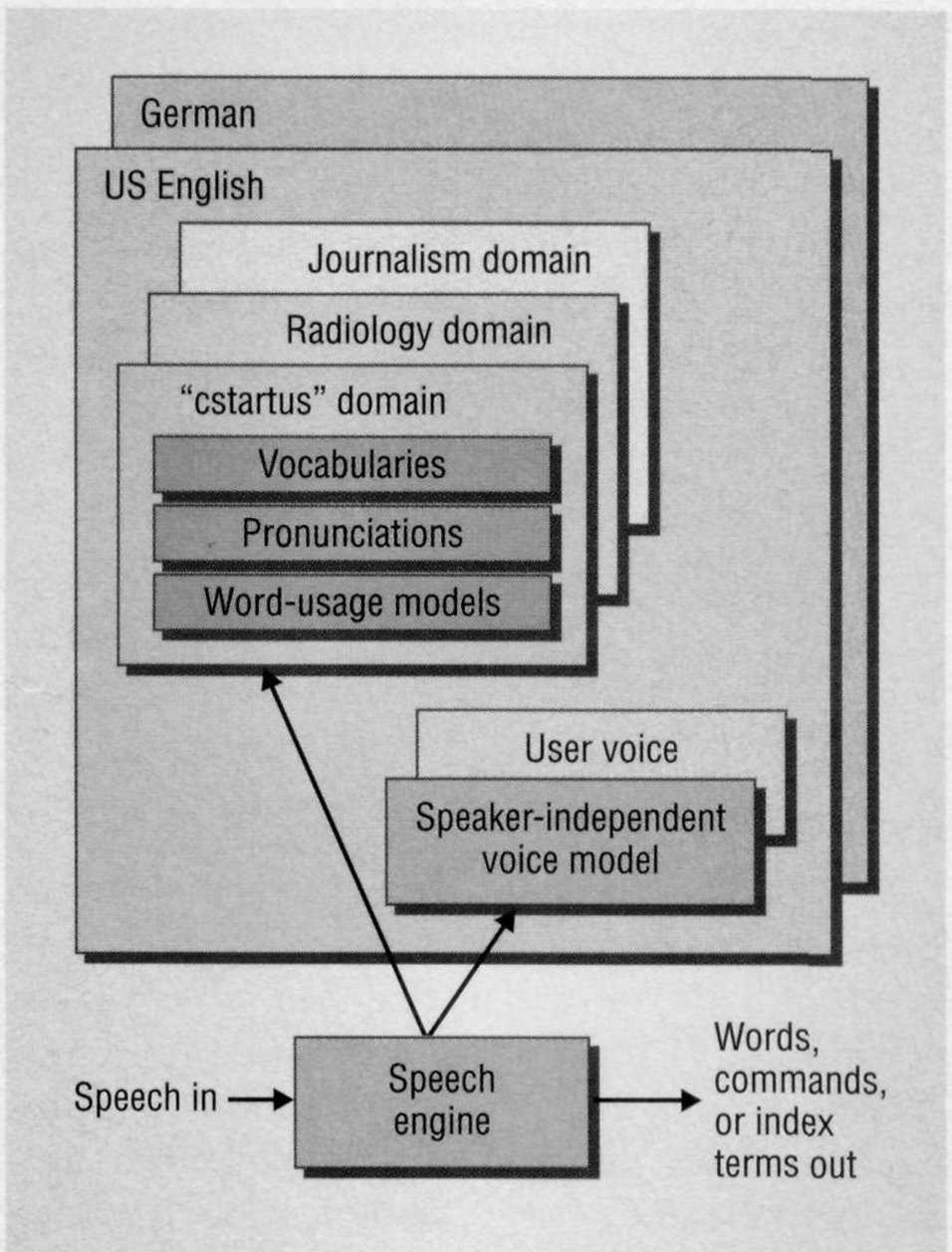

Figure 1. Speech recognition resources. Domain-specific resources—such as the vocabulary—can vary dynamically during a given session to accommodate, for example, dictation, transaction, or multimedia indexing applications.

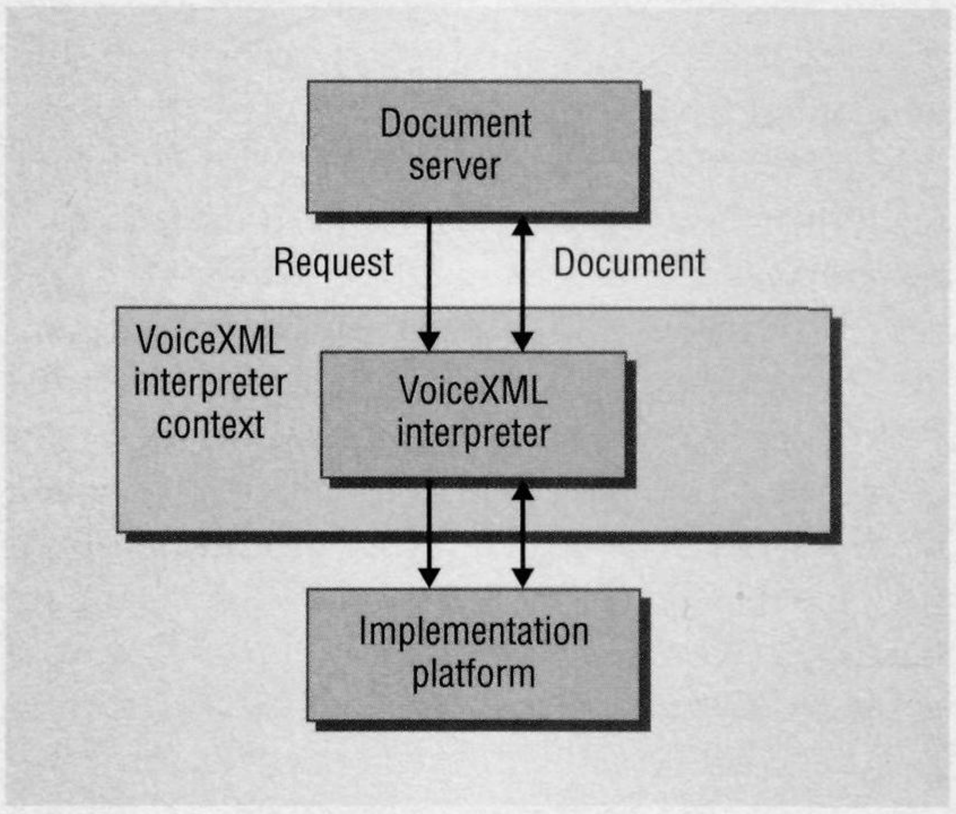

Figure 2. VoiceXML architectural model. The model uses the client-server paradigm and views a voice service as a sequence of interaction dialogs between a user and an implementation platform.

THE VOICE WEB

In March 1999, AT&T, IBM, Lucent, and Motorola established an industry organization, the VoiceXML Forum, to develop and promote a new computer language, the Voice extensible Markup Language (http://www.w3.org/Voice/). Developers can use VoiceXML to create audio dialogs that feature synthesized speech, digitized audio, recognition of spoken and touchtone key input, recording of spoken input, telephony, and mixed-initiative conversations.

The four partners designed VoiceXML to make Internet content and information accessible using speech recognition and mobile devices. The VoiceXML forum was founded to bring the advantages of Web-based development and content delivery to interactive voice response applications.

Figure 2 shows VoiceXML's architectural model, which uses the familiar client-server paradigm to integrate voice services with data services. A voice service is a sequence of interaction dialogs between a user and an implementation platform. Document servers, which can be external to the implementation platform, provide the dialogs. Document or