顧客チャネルとしてのスマートスピーカー
―応答の仕組みとアプリ開発のポイント―

近年、日本でもスマートスピーカーの認知率が高まり、PCやスマートフォンに次ぐ新しい顧客チャネルとなり得るデバイスとしての期待も寄せられている。本稿では、スマートスピーカー向けのサービスを開発する上で押さえておくべき重要なポイントについて、実際の開発を通じて得られた知見を基に解説する。

NRIネットコム デジタルインテグレーション事業本部
クラウド事業推進部 モバイル事業推進課 主任システムエンジニア
佐藤（さとう） 雄也（ゆうや）
専門はWebアプリケーションおよびスマートフォンアプリの設計・開発

認知率が高まるスマートスピーカー

ここ数年、米国Amazon.com社の「Amazon Echo」や米国Google社の「Google Home」をはじめとする、音声で操作可能なアシスタントデバイスであるスマートスピーカーが話題となっている。音声による操作といえば、米国Apple社の「Siri」やGoogle社の「Google Assistant」のような、利用者の音声に応答するスマートフォンのアシスタント機能としてすでに利用されているが、スマートスピーカーは主に自宅やオフィス、車の中などで据え置き型のデバイスとして利用する。

スマートスピーカーには共通して、利用者へ情報（天気、交通情報など）を伝えたり音楽を再生したりするスピーカーと、利用者の音声を聞くマイクの他、クラウドに接続するための無線LAN通信モジュールが搭載されている。また、事前の設定が必要ではあるが、音声の指示に従って家電製品を制御するといった高度な操作にも対応する。

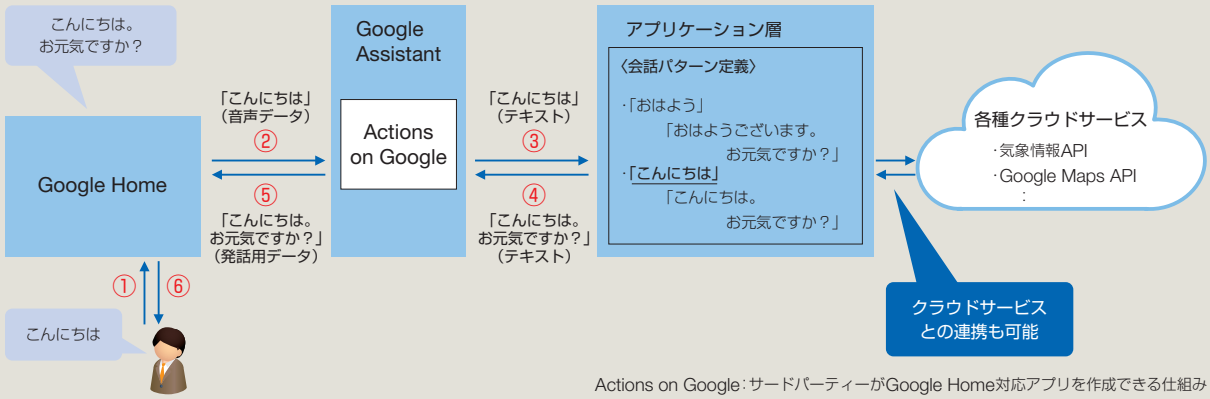
日本では、まだスマートスピーカーの所有率は低いと思われるが、2018年2月のアイブリッジ社の調査によると、認知率はすでに50%を超えている（http://www.research-plus.net/html/investigation/report/index128.html参照）。今や生活必需品となっているPCやスマートフォンと同様に、スマートスピーカーも遠からず家庭とクラウドをつなぐ重要なインターフェースとなることは間違いないであろう。

スマートスピーカーのもう1つの大きな特徴は、サードパーティーによる機能の拡張が可能だということである。すでに多くの企業や個人が拡張機能のアプリを公開している。まだそれほど普及しているわけではない日本でも、今後の普及を見越して、あるいは普及に弾みをつけるために、2018年5月現在「Amazon Echo」用には700種類以上、「Google Home」用には300種類以上のアプリが配信されている。

顧客チャネルとしての活用

以上のような特徴や可能性から、スマート

図1　スマートスピーカー「Google Home」のアーキテクチャー



Actions on Google：サードパーティーがGoogle Home対応アプリを作成できる仕組み

スピーカーの用途として有望視されるのが顧客チャネルとしての活用である。実際、米国Domino's Pizza社の英国法人は、ピザの宅配サービスの注文をスマートスピーカーでできるようにしている。これは、もともとWebサイトやFacebookなどで提供していた「イージーオーダー」と呼ばれる機能（事前に登録したメニューを少ない操作で注文できる機能）をスマートスピーカー用アプリに移植し、自宅などから音声のみで注文できるようにしたものである。同社によると、サービス開始から2カ月後には、「イージーオーダー」で注文する顧客のうち5人に1人がスマートスピーカーを利用するようになったという（https://digiday.jp/brands/dominos-getting-people-order-pizza-amazon-alexa/ 参照）。一見すると単なる趣味の対象のようなデバイスが、アイデアや企画次第でビジネスにつながることを示すものといえる。

スマートスピーカーの仕組み

サービスの開発に当たっては、スマートスピーカーとそのアプリがどのような仕組みで動作しているのかを理解する必要があることは言うまでもない。ここでは、スマートスピーカーの仕組みとともに、利用者に受け入れられ、長く使われるアプリを生み出すための企画の進め方や開発における注意点について、NRIネットコムが顧客のサービスの利便性を高めることを目的に実施したPoC（Proof of Concept：概念実証）の経験に基づいて解説することにする。

ここでは代表例として、「Google Home」向けの「Google Assistant」用アプリが動作する仕組みを解説する。「Google Home」以外の一般的なスマートスピーカーでも基本的な仕組みは変わらない。

「Google Home」本体を含むアーキテクチャーの全体像を図1に示す。特徴的なのは、アプリを含むリソースがすべてクラウド上に置かれているという点である。利用者側からは「Google Home」の中でアプリが動いているように見えるが、実際にはネットワークを介してクラウド側のサーバー上で動作する。そのため、開発者側がアプリに機能を追加した場合も、利用者はバージョンアップの操作をすることなく新しい機能を利用することが

できる。

「Google Assistant」は、「Google Home」（あるいはスマートフォンなどの「Google Assistant」対応端末）とアプリの間をつなぐ仲介役で、利用者の呼び掛けに応じてアプリを起動し、発話内容をアプリに伝えたり、アプリからの応答を「Google Home」へ伝えたりする役割を持つ。「Google Home」から送られる音声化された発話内容は、「Google Assistant」がテキストデータに変換してアプリに伝えている。

アプリは基本的に「Google Assistant」の背後で受動的に動作する仕掛けとなっており、利用者の発話内容の解析、解析結果に基づく返答内容の選択を役割とする。アプリは主に「Dialogflow」と呼ばれるクラウドサービスを用いて開発される。応答の仕組みは非常にシンプルだが、アプリによって利用者と「Google Home」との会話のバリエーションを増やし、よりスマートな音声アシスタントとすることができる。

クラウドサービスや既存システムとの連係

図1では、「Google Home」との間であいさつを交わすだけの単純なやり取りを示したが、気象情報や交通情報、経路情報などを提供するクラウドサービスと連係させて、もっと高度な応答をさせるアプリを開発することも容易である。

例えば、「明日の天気は？」と話し掛けると、気象情報サービスから天気の情報を取得し、「明日の天気は晴れです」と応答するアプリである。こうしたクラウドサービスとの連係は、リアルタイムな情報を応答に加えることを可能にする。

ただし、外部のサービスと連係させる場合には、スマートスピーカーに特有の注意点もある。例えば、交通情報サービスと連係させる場合に、その情報が全体の交通情報を1つにまとめた文章（長文のテキスト）であったとする。PCやスマートフォンのように文字を表示できるデバイスであれば、それをそのまま表示してもほとんど問題はない。しかし、スマートスピーカーのように音声だけで情報を伝えるデバイスでは、必ずしも人が理解しやすい応答になるとは限らない。スマートスピーカーはテキストの文脈を理解する機能を持つわけではないので、聞き取りやすい速度にしたり、文節ごとに「間」を置いたりするようなことは現状では難しいからだ。

従って、連係させるサービスを選定する際は、文章を返すサービス（主にWebを想定したサービス）ではなく、カテゴリー単位でデータを区切って提供してくれるサービスを選択するのが望ましい。そうすれば、応答の仕方をある程度まではコントロールできるようになる。

外部のサービスだけでなく、既存システムとの連係も可能である。「Google Home」の場合、OAuth（オーオース。あるシステムで保持している利用者の情報にシステムの外からアクセスするための認可の仕組み）と呼ばれる技術をサポートしている。このOAuthを活用して、利用者の氏名や年齢、住所などの情報を取り込んだ個人的な会話（誕生日を祝ったり近所の情報を伝えたりするなど）を

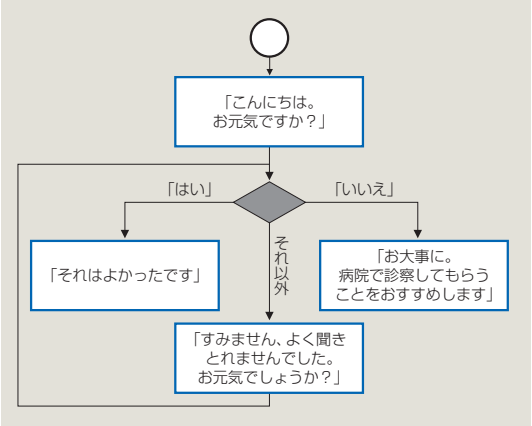
行うアプリの開発も可能である。既存のステムがOAuthに対応していない場合はシステムの改修が必要になるが、ユーザーフレンドリーなアプリを開発するためには必要不可欠といえる。

「Google Home」の「Voice Match」という仕組みも、ユーザーフレンドリーなアプリを開発する上で重要である。これは、あらかじめ利用者の声紋とGoogleアカウントを結び付けておき、話し掛けた人が誰かを自動で判別するもので、これを利用すれば、同じアプリでも利用者ごとに異なる会話をさせることができるようになる。ただし、まれに誤検知する場合があるため、少なくとも厳格な生体認証としての使用は避けるべきである。NRIネットコムのPoCでも「Voice Match」を利用した認証の実現可能性について検証したが、登録した声紋に似た声質を持つ人が話し掛けると誤検知する場合があった。あくまでもユーザビリティ向上のための仕組みと考えるべきであろう。

最も重要となる企画段階

ここまでは、主に技術面について触れてきたが、スマートスピーカー向けのアプリを開発する上で最も重要なのは、アプリの要件やコンセプトを定義する企画段階である。スマートスピーカーでは、画面に表示されるテキストや画像のような視覚的な要素が得られない分、音声を聞いた利用者の行動にばらつきが生じやすい。NRIネットコムのPoCでも、開発者側が想定していない応答を利用者がしたために、アプリが認識できずにその後の会話が成り立たなくなるケースがいくつか生じた。アプリの公開後にこのような事態を招かないようにするためには、企画の段階で大まかな会話の流れを決定し、例外的なパターンを含めてユーザーテストを入念に実施しておくことが望ましい。

図2　応答のフローチャートの例



NRIネットコムのPoCでは、図2に示すようなフローチャート形式で会話全体の流れを定義し、それに基づいて試作品を作り、ユーザーテストを実施した。フローチャートには「スマートスピーカーの発話内容」、会話の分岐条件となる「ユーザーの想定発話内容」が記載され、会話全体の流れを見渡せるようになっている。この流れに基づく試作品をつくって動かしてみて、会話の流れがスムーズか、発話内容が簡潔かという確認や、利用者の発話内容の想定が適切か、不足していないかなどの確認をする。なお、試作品をつくるためのリソースが用意できない場合には、フローチャートに基づいて人がスマートスピーカー役を演じることによってもある程度の検証が可能である。このように、利用者の立場に立って実際に体験し、改善点を洗い出していくことが重要である。■