Note de lecture pour la revue TALN à propos du livre VoiceXML

José ROUILLARD, VoiceXML : Le langage d'accès à Internet par téléphone, Vuibert, 2004, 224 pages, ISBN 2-7117-4826-x.

par Luiggi SANSONETTI
EA2290 SYLED – EA170 CALIPSO – ILPGA – Paris III
luiggi.sansonetti@noos.fr

Cet ouvrage présente le langage VoiceXML, standard permettant l'accès à Internet par téléphone. L'approche se veut pragmatique en donnant les bases du langage, en passant par son histoire, et les outils nécessaires pour la mise en oeuvre d'un serveur vocal avec des exemples concrets d'applications vocales.
Le premier chapitre présente de façon générale le pourquoi d'un tel langage. L'auteur nous rappelle les fondements historiques avec les projets pionniers comme PhoneWeb, Phone Markup Language, VoxML, SpeechML pour aboutir à la recommandation du W3C nommée VoiceXML (1.1). Après avoir cité les quelques applications vocales existantes comme l'IHM ou le DHM (1.2), José Rouillard nous pose la question suivante : pourquoi créer des applications vocales ? (1.3). Il y a de plus en plus de téléphones portables et ils sont de plus en plus sophistiqués. Avec un téléphone, il n'y a plus de problème de compatibilité graphique pour la connexion Internet. Le handicap de lecture / écriture et balayé. Il s'agit d'une interaction mains libres et/ou yeux libres. L'auteur souligne le fait qu'en écoutant nous pouvons regarder autre chose en même temps. Les avantages pour les entreprises sont nombreux : informations disponibles 24/24, contact permanent avec l'employé et/ou le client.

Il y a deux grands types d'applications vocales (1.4) : les requêtes pour obtenir des informations d'ordre général et consulter des données, et les transactions pour exécuter une action spécifique comme une mise à jour des données. Le langage à balise qu'est VoiceXML va présenter à l'utilisateur les données de manière vocale. Les avantages d'un tel langage par rapport au HTML sont que les applications sont interactives, le langage VoiceXML gère déjà les mêmes infrastructures que pour les interfaces visuelles, peut aussi gérer dynamiquement en intégrant l'ASP et le PHP, l'environnement est ouvert (et donc non propriétaire) et multi plateformes, les entrées se font par voix et/ou par clavier du téléphone (DTMF), les aides peuvent être nombreuses et variées (1.5). Mais la technologie vocale reste source d'erreurs car l'utilisateur a un certain débit, une certaine prosodie, un accent avec les ambiguïtés, les noms propres, les mots inconnus. Les grammaires vocales sont limitées car il s'agit d'un langage interprété et non compilé. Il y a dans la langue une surcharge cognitive qui serait difficile à implémenter (trop de nuances, de sousmenus). Et il y a surtout un dernier point : la volonté de l'homme à parler à une machine.

En considérant tous ces paramètres et ces possibilités, il reste à concevoir une machine conversationnelle à états finis prenant en compte un état à la fois (1.6). Pour ce faire, il y a deux types de dialogues : les formulaires qui sont dirigés ou à initiative mixte et les menus qui proposent plusieurs choix. Le fonctionnement d'une telle application est alors assez simple : nous avons un utilisateur qui passe par le réseau téléphonique et arrive sur une passerelle VoiceXML. Cette passerelle va envoyer une requête http à un serveur pour consulter une base de données. Dans le deuxième chapitre, l'auteur nous décrit les moyens possibles pour aborder les bases du développement d'applications en VoiceXML. Tout d'abord, nous pouvons simuler une interaction avec un ordinateur (2.1) pour tester son application avec IBM Websphere Voice Server SDK. Nous obtenons le schéma de communication suivant :

[Téléphone – Réseau – Passerelle] – [Serveur]
[Simulation par ordinateur] – [Réel]

Nous pouvons ensuite avoir une interaction réelle mais avec un hébergement distant (2.2), par exemple si on n'a pas le matériel disponible. Pour cette solution intermédiaire, l'auteur nous présente quelques hébergeurs et services comme VoiceXML Gateway (2.2.1), Sophone (2.2.2) et VoxBuilder (2.2.3). Le schéma est alors légèrement différent :

[Téléphone – Réseau] – [Passerelle] – [Serveur]
[Réel] – [Hébergement distant par un service] – [Réel]

Enfin, nous pouvons avoir une interaction réelle avec un hébergement local (2.3). Mais il faut disposer alors de tous les logiciels et matériels nécessaires. Le schéma peut être représenté ainsi :

[Téléphone – Réseau] – [Passerelle] – [Serveur]
[Réel] – [Hébergement local] – [Réel]

Dans le troisième chapitre, José Rouillard nous donne tous les éléments de base du langage (3.1) pour faire parler et faire écouter la machine à travers des petits exemples très faciles à réaliser. Pour faire parler une machine, il y a la diffusion de synthèses vocales en mode TTS pour Text-To-Speech (3.2.1), de fichiers sonores au format wave par exemple (3.2.2). Il est possible d'améliorer la prononciation à l'aide de pauses, d'emphases, en proposant des choix de voix (3.2.3). Pour faire écouter une machine, il y a la reconnaissance vocale qui permet à l'utilisateur de parler plutôt que de taper sur une touche (3.3.1). L'enregistrement vocal permet de stocker un message pour un traitement ultérieur (3.3.2). Enfin, dans certains cas, l'utilisateur utilise sa voix comme le clavier (3.3.3).

Le quatrième chapitre (le plus gros) explique comment gérer un dialogue homme-machine en VoiceXML. Le tout est, comme précédemment, illustré de petits exemples simples et concrets. Nous voyons d'abord le rôle des grammaires vocales (internes et externes) pour comprendre ce que dit l'utilisateur (4.1). Il peut y avoir des importations de grammaires (4.1.3) ainsi que de variables et expressions ECMAScript (4.2). Pour gérer un dialogue homme-machine, le langage VoiceXML permet d'effectuer des branchements conditionnels de type « si … alors … sinon » (4.3.1). Il est également possible de proposer des aides aux utilisateurs pour les guider dans leurs choix ou expliciter une demande (4.3.2). Pour guider l'utilisateur, les principales applications vocales procèdent par menus et formulaires (4.4). Cette structure permet de diriger le dialogue (4.5), comme de laisser l'utilisateur prendre, à un moment autorisé, l'initiative de l'interaction (4.6). Ces transitions dans les dialogues à initiative mixte peuvent se faire par sous-dialogues internes ou externes (4.7). Elles peuvent aussi se faire par des envois simples ou par liens associé à une grammaire vocale (4.8). Beaucoup d'autres éléments sont possibles en VoiceXML comme les transferts d'appels et les déconnexions (4.9), la mise en mémoire d'objets ou composants (comme une horloge par exemple), l'intégration de scripts de calculs (4.10), la gestion de plusieurs documents (4.11) ou plusieurs sessions (4.12). Ce chapitre se termine par une dizaine de conseils pour améliorer les dialogues homme machine (4.14). Le dernier chapitre consiste à résumer toutes les notions vues jusque là en trois études de cas. Le premier exemple est un jeu qui consiste à retrouver le nombre choisi par l'ordinateur (5.1). Le deuxième exemple est une calculatrice vocale (5.2). Et le dernier utilise un service Web des codes postaux pour retrouver le nom de la ville française associée à son code postal (5.3). Les annexes présentent la procédure d'installation d'IBM Voice Serer SDK (A), les balises de VoiceXML avec tous les attributs autorisés (B), des listes de ressources tant opérateurs ou passerelles (C.1) que documents relatives au langage (C.2).

Pour conclure, l'ouvrage de José Rouillard, par son approche simple et détaillée, nous donne envie de programmer en VoiceXML, de comprendre ce langage et permet également d'aller plus loin avec les nombreux exemples d'applications et la qualité des annexes.