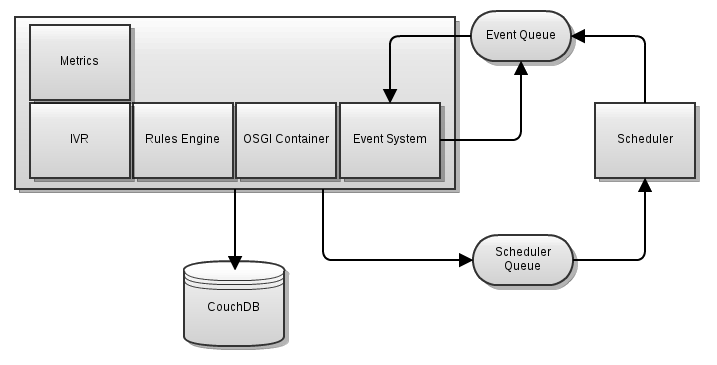
|  |
| --- |
| HCL Technologies Ltd. |
| MOTECH Framework – Solution Approach on Tamil language support in IVR |
|  |

****

# Overview

1. MOTECH – Mobile Technology for Community Health
2. Set of open source technologies which address core needs of **Mobility Health (mHealth)**
3. **mHealth** - a spectrum of tools and applications to extend health services to underserved populations, helping to educate and inform patients, and helping community-based workers diagnose, treat and monitor a broad range of issues
4. **MOTECH** Suite delivers an integrated set of complementary applications that are scalable, sustainable and readily deployed for mHealth implementation
5. **General features of MOTECH applications :**
   1. Communicate information to patients by sending reminders, via **voice** or **SMS**
      * For appointments, lab visits, to take medicine.
      * To take medication
      * To take children for scheduled immunization services
   2. Collect data from patients or caregivers:
   3. Alert caregivers of the status of their patients
   4. Facilitate communication between patients, caregivers, and/or health administrators
6. **Implementations**
   1. Mobile Midwife & Nurses Application – Ghana
   2. TAMA – Urban India
   3. Ananya – Bihar, India
      * 1. MOBILE HANDSET APPLICATION FOR FRONT LINE HEALTH WORKERS (FLWS)
        2. IVR TRAINING TOOLS FOR FLWS
        3. KILKARI
   4. World Health Partners TB Case Management

# Architecture



**Motech High level Architecture**

In a high level system architecture, MOTECH is logically divided into three tiers/layers :

* 1. Core layer
  2. Module layer
  3. Implementation layer

1. **Core Layer**

* Wraps several well-Known open source systems and exposes their features to the layers above it
* Wraps ActiveMQ and present an event interface to the module and implementation layers
* Provides an interface to the scheduler and access to CouchDB
* provides a module loading environment (OSGi- Open Service Gateway initiative)

1. **Module Layer**

* module layer is a collection of reusable components
* Modules interact with the core platform through its APIs
* Modules interact with other modules either through their service interfaces or by consuming their events
* Modules may also register servlet controllers which allow them to respond to HTTP requests.

1. **Implementation Layer**

* logical layer that contains implementation specific code

1. **Stateless**

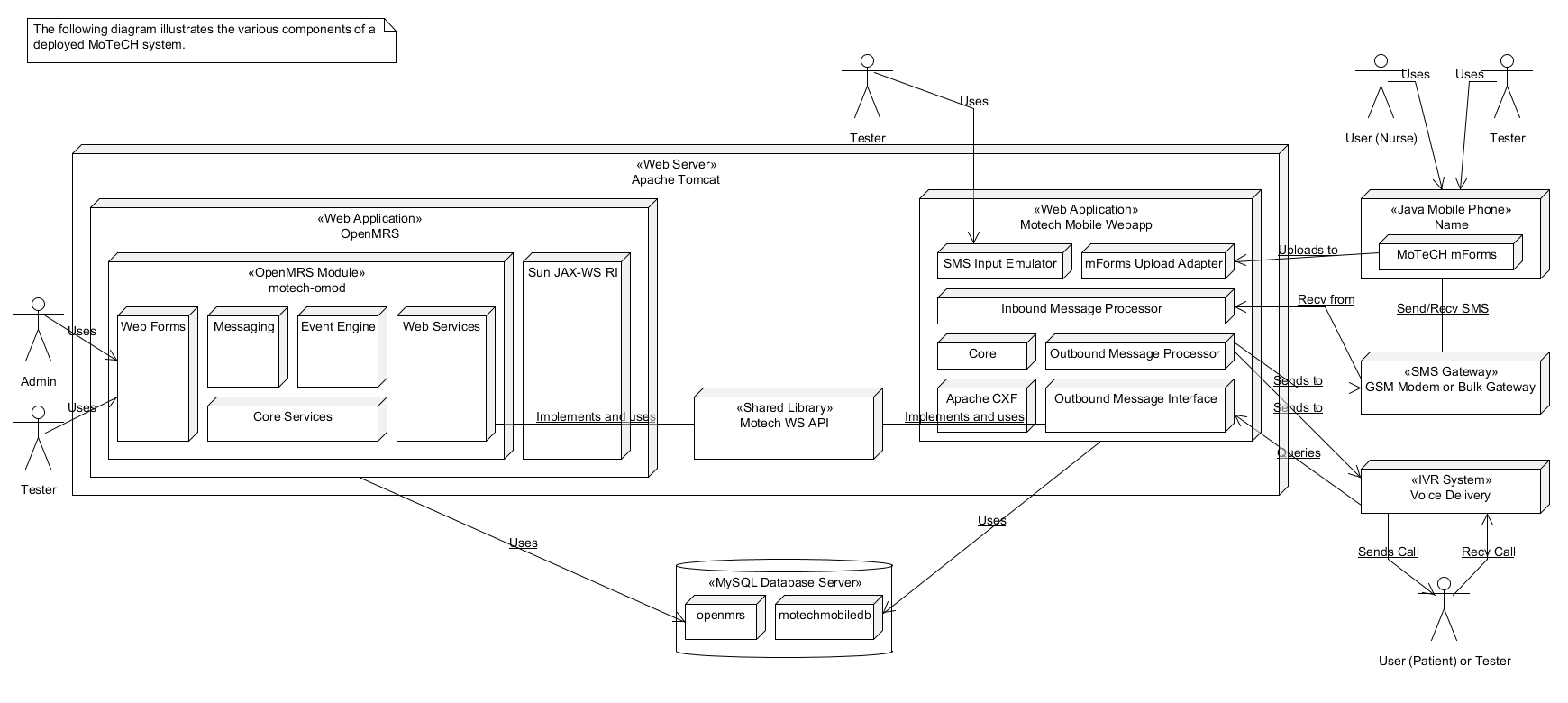
* MOTECH is stateless – MOTECH server perform a single action per request and then return
* The module should never persist any state in memory and expect that state to be available to later requests

1. **Events**

* MOTECH engine provide an event system
* Any module can emit an event by calling the *eventRelay* and passing it a *MotechEvent* and a subject.
* To register for an event a module just needs to annotate a method with the list of event subjects or interest.

1. **Scheduled Events & Timers**

* MOTECH provides access to a flexible scheduling system
* Uses Quartz (open-source) application to schedule events

**The component architectural figure of MOTECH system:**

# Module List

Following are the modules available in the MOTECH suite:

1. [**Alerts**](http://www.motechproject.org/modules/alerts.html) :- Collects alerts for users in an inbox-like container
2. [**Appointments**](http://www.motechproject.org/modules/appointments.html) :- Provides appointment scheduling and reminding
3. [**CMS Lite**](http://www.motechproject.org/modules/cms-lite.html):- Provides basic content storage and retrieval
4. [**Decision Tree**](http://www.motechproject.org/modules/decision-tree.html) :- Provides APIs for constructing an IVR decision tree
5. [**IVR**](http://www.motechproject.org/modules/ivr.html):-Provides basic specification for integrating platform with an IVR service provider; Also, connects the platform IVR with an asterisk server using VoiceGlue VXML browser
6. [**IVR Kookoo**](http://www.motechproject.org/modules/ivr-kookoo.html):-Integrates the platform with Kookoo’s hosted IVR
7. [**IVR Voxeo**](http://www.motechproject.org/modules/ivr-voxeo.html):-Integrates the platform with Voxeo’s hosted IVR
8. [**Message** Campaign](http://www.motechproject.org/modules/message-campaign.html):-Enrolls users in message campaigns with flexible content scheduling rules
9. [**Mobile forms**](http://www.motechproject.org/modules/mobile-forms.html):-Supports configurable forms and data collection though mobile devices.
10. [**MRS (Medical Record System)**](http://www.motechproject.org/modules/mrs.html):-Provides basic specification for integrating platform with a medical record system
11. [**OpenMRS**](http://www.motechproject.org/modules/openmrs.html):-Integrates platform with OpenMRS system. OpenMRS is the database where health records are stored.
12. [**Outbox**](http://www.motechproject.org/modules/outbox.html):-A voicemail like messaging system for end users
13. [**Pill Reminder**](http://www.motechproject.org/modules/pill-reminder.html):-A flexible reminder system focused on medication
14. [**Schedule Tracking**](http://www.motechproject.org/modules/schedule-tracking.html):-Enrolls users for alerts based on complex scheduling rules
15. [**SMS**](http://www.motechproject.org/modules/sms.html):-Provides basic specification for integrating platform with an SMS provider to send/receive SMS messages
16. [**SMS HTTP**](http://www.motechproject.org/modules/sms-http.html):-Allows platform to integrate with an HTTP-based SMS gateway to send SMS messages
17. [**SMS SMPP**](http://www.motechproject.org/modules/sms-smpp.html):-Allows platform to integrate with an SMPP-based SMS gateway to send/receive SMS messages

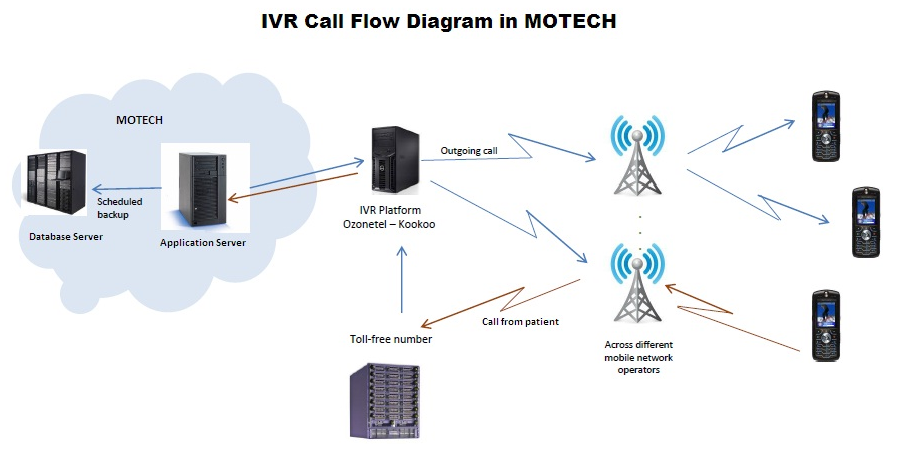
# Features

1. **Demand Generation** - for health services and health-seeking behaviors through direct messages or reminders to clients. These messages/reminders can be delivered via IVR or SMS
2. **Registration** – patient and health workers both are registered for better coordination and delivering more effective healthcare. Registration can be done in any of the following modes :
   1. via IVR
   2. via Simple forms
   3. via Mobile applications
   4. via Web UI
3. **Front Line Health Workers Empowerment** - training Front Line Workers (FLWs), help them manage their workload and facilitate more productive interactions with their patients
   1. Training Courses
   2. Medical Protocols facilitated by mobile devices
   3. Mobile Data entry
   4. Follow-up and defaulter alerts
   5. Works with available technologies like IVR, SMS, handset application, Web based UI
4. **Supply Chain Logistics** 
   1. Reporting stock outs
   2. Notification of new supply shipment
   3. Inventory management
5. **Integration -** with existing health systems , governmental or NGO databases, national ID systems, and other help systems
6. **Adherence Monitoring –** checking if advised actions are followed and intervening if not.
7. **Hosting**

* Self-hosted
* Hosted in the cloud

# IVR Integration in MOTECH

IVR means **Interactive Voice Response**. IVR is used in MOTECH mainly for patients to interact with system for different issues. Patients, in rural areas, will not have the access to mobile application of MOTECH and may not have the literacy skills to navigate written information in the application. For them, IVR can be more user friendly mode to gather and provide information. Patients will receive interactive voice messages encouraging them to pursue health-seeking behaviors. Pregnant women and mothers can receive IVR calls that remind them of healthy behaviors, and encourage them during their pregnancy and baby’s first year to follow the schedule of care.



## Seamless support of IVR Integration in MOTECH

## MOTECH framework has two major portion

* Base IVR framework
* Framework to integrate actual IVR

## Base IVR Module in MOTECH

1. MOTECH uses IVR module as base platform to implement functionality of different IVR (KooKoo,Voxeo etc.).
2. It has an interface **IVRService.java**
3. This interface is implemented by actual IVR module
4. IVRService.java has a method **initiateCall()** 🡪overridden in IVR implementation
5. **CallRequest** object 🡪 contains phone no. , caller id,call time out, etc.
6. initiateCall() method use CallRequest Object to pass caller detail to IVR

**Implementation Flow :**

CallRequest Object

Implemented by

Other IVR Module (KooKoo, Voxeo, etc.)

IVRService.java

initiateCall()

**Interface**

|  |  |  |
| --- | --- | --- |
| Interface  IVRService.java   |  | | --- | | Member Variable   * EXTERNAL\_ID : String = “external\_id” * CALL\_TYPE : String = “call\_type” | | Method   * initiateCall(CallRequest) : void | |

**Structure of Base IVR Framework**

**CallRequest Object**

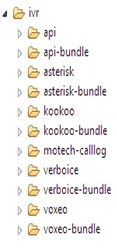
1. CallRequest object is used by initiateCall Method
2. initiateCall method is used to initiate/start (outbound) call
3. CallRequest object have the data require to make an outbound call
4. CallRequest has
   * **phone** (String) - The phone number to call
   * **timeOut** (int) - The time out value for the call
   * **callBackUrl** (String) - The URL for VXML content
   * **payload** (Map) - A map for adding additional data to the call, this is where your unique identifier would be placed.
   * **onSuccessEvent** (MotechEvent) - The event fired when the call is successful.
   * **onBusyEvent** (MotechEvent) - The event fired when the call is busy.
   * **onNoAnswerEvent** (MotechEvent) - The event fired when the call is not answered.
   * **onFailureEvent** (MotechEvent) - The event fired when the call failed for some reason

### Actual IVR Module in MOTECH

1. Framework to integrate **IVR server** with **MOTECH server**.
2. Have to implement **IVRService** interface of **Base** IVR framework
3. Using base IVR interface, many different IVR systems can be integrated with MOTECH system
4. Some of IVR used in different projects of MOTECH are
   * [**KooKoo**](http://www.kookoo.in/)
   * [**Voxeo**](http://voxeo.com/)
   * [**Verboice**](http://verboice.instedd.org/)
   * [**intellIVR**](http://www.yo.co.ug/index.php?option=com_content&task=blogcategory&id=25&Itemid=85)

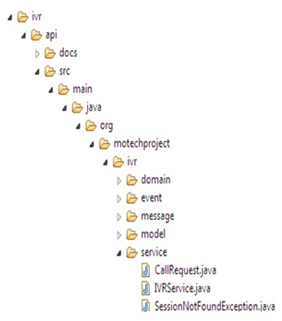
For India based project, KooKoo is used as KooKoo is **India based Telephony Service** and supports different Indian language.

The below directory structure represent the **IVR** **module** inside the **MOTECH** framework. It contains the base IVR module of MOTECH (folders : api and api-bundle)and the modules implementing other IVRS like KooKoo, Voxeo, Verboice etc.



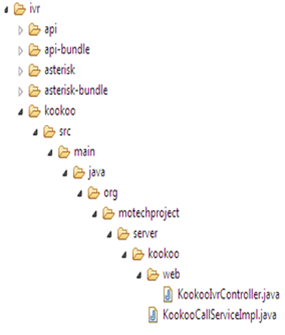
**IVR folder in MoTech**

Below directory structure shows the folder and file structure of base IVR module in MOTECH.



**Directory Structure for IVRService interface in Base IVR Module of MoTech**

The figure below shows the directory structure of KooKoo IVR implementation module in MOTECH. It contains the implementation file, which actually implements the IVRService interface from base IVR module of MOTECH.



**Directory structure for kookoo implementation Files**

### Integration of IVR server with MOTECH Server

1. Two files require to integrate base IVR
   1. **IVRService implementation file** – implement IVRService.java interface and override initiateCall()
   2. **IVRController file** – controls the request and response made to actual IVR Server
2. Every IVR module implemented into MOTECH framework will have its own copy of IVRService implementation file and IVR controller file as a shown in above figure.
3. Implementation file of a particular IVR Module will communicate with IVR Server
4. Communication mode can be of any type (HTTP Request/Response, webservices, etc.) based on IVR Server

Base IVR

KooKoo IVR Module

* KookooCallServiceImpl.java
* KookooIvrController.java

IVRService interface

Voxeo IVR Module

* VoxeoIVRService.java
* IvrController.java

Verboice IVR Module

Other IVRs



**MOTECH Server**

**MOTECH implementation to integrate IVR System**

## Example : KooKoo Server Integration

**MOTECH Server**

HTTP Request

Base IVR

**KooKoo**

**IVR Server**

Call details like caller no. etc.

IVRService interface

KooKoo

IVR Module

HTTP Response

**KooKoo Tunes** (XML) Commands (like <playtext>,<playaudio> etc.)

**KooKoo IVR integration with MOTECH**

**Steps:**

1. KooKoo server gets a call from caller
2. Make a HTTP request to MOTECH Server’s KooKoo Module, passing along the caller detail
3. Caller detail, caller no. etc. are stored in CallRequest Object
4. InitiateCall(CallRequest) method in KooKooCallServiceImpl.java is executed
5. KooKooIVRController.java calls the appropriate method based on the request made
6. This method generate the XML (KooKooML Commands) as per the request
7. These commands are passed to KooKoo Server as response.
8. KooKoo perfoms the action as the command and thus interact with the user.

**Steps of KooKoo Integration with MOTECH**

# Call Flow for IVR Execution

1. **HW/PL Dials the Number**

Enter the pin

C:\Users\kumari.r\Desktop\Capture.PNG

**5. Perform Action**

**4.** **Kookoo performs the action specified in the XML on your behalf and interacts with the caller**

**MOTECH Server**

**KooKoo**

HTTP Request

Ex. http://www.mysite.com/myapp.jsp

**2. Kookoo Answers the call, and makes a HTTP request to the URL of Motech Application.**

**KooKoo Response**

**KooKoo Server**

Response

KooKooML (XML) Commands (like <playtext>,<playaudio> etc.)

3**. Motech Application receives the request performs some action and sends an XML response back to kookoo to perform some actions.The response XML is called Kookoo tunes.**

**Call flow between KooKoo IVR and MOTECH**

**Steps of the Call Flow**

Step 1: A HW/PL dials the number.

Step 2: Kookoo Answers the call, and makes a HTTP request to the URL of MOTECH Application. Let us say our URL is <http://www.mysite.com/myapp.jsp>.

Step 3: MOTECH Application receives the request performs some action and sends an XML response back to kookoo to perform some actions. The response XML is called Kookoo tunes.

Step 4:Kookoo performs the action specified in the XML on your behalf and interacts with the caller.

Step 5**:** Perform Action.

# Pattern for Multi-language support in IVR

For integrating multi language in MOTECH framework, recorded audio files must be used. That is, if we want IVR messages to be played in different language, we need to record those messages in that particular language and play through IVR systems. These recorded files must be in “**WAV**” file format.

**Text to speech engine is not implemented in MOTECH for multi-language support**.

## Directory Structure of multi-language WAV files

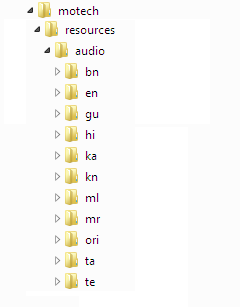
The wav files must be stored in a folder structure to be accessible to IVR Systems. The wav files for different language must be stored in separate folders named with language code.

**Example of Language Code**

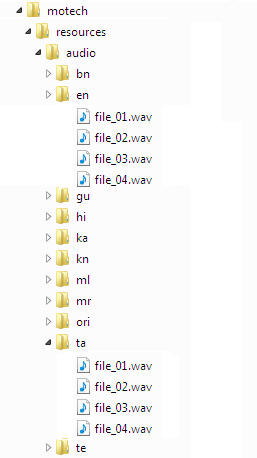
* English : ‘en’
* Hindi : ‘hi’
* Tamil : ‘ta’
* Kannada : ‘ka’
* Malayalam : ‘ml’
* Telugu : ‘te’
* Gujarati : ‘gu’
* Bengali : ‘bn’
* Marathi : ‘mr’
* Oriya : ‘ori’

The directory structure to store multi language WAV files in MOTECH:

A.

To integrate different languages in MOTECH framework, we can use wav files to play the content in a specific language. These wav files are stored in directory under “audio” folder. We can create different folders for different language with language code as folder name. For example, for English we have “en” folder, for Tamil we have “ta” folder.

B.

Inside each language folder, the sub folders will be same. That is, directory structure inside Tamil folder will be same as that of English folder. Inside these sub folders, the wav files are residing. The name of these wav files will also be same in each language folders. Only the content of the files will be in different language. That is, “en” folder will contain the wav files recorded in English and “ta” folder will contain the wav files recorded in Tamil.

## Getting WAV files for specific language

The Call Flow in MOTECH is basically through audio files of “**WAV**” format. All the required IVR messages, to Health workers and also to patients, are recorded in wav file format. These wav files are played via IVR systems.

**The logic to get the appropriate wav files according to call flow can be implemented in two ways**:

* If IVR system provides APIs to implement its service, then the call flow logic can be written on MOTECH framework, i.e. MOTECH framework can decide the logic to pick the wav files
* And if not, then these call flow logic must be implemented in IVR servers, i.e. the logic to pick the correct depends on IVR framework which is integrated.

For integrating Indian language, we consider to implement KooKoo IVR into MOTECH framework. KooKoo is very flexible and provide APIs for integration. So, we can have call flow business logic in our framework.

There is a configuration file “**ivr.properties**”, which contain IVR specific parameters as given below:

|  |  |
| --- | --- |
| **Configuration Parameters** | **Description** |
| kookoo.outbound.url | http://kookoo.in/outbound.php |
| kookoo.api.key | get api key from kookoo.in |
| max.number.of.attempts | Number of attempts for login |
| application.url | Default url for IVR Controller http://{host}:{port}/{appcontext}/ivr/reply |
| kookoo.outbound.url | http://kookoo.in/outbound.php |

The location of “wav” files is stored in “content.location.url” parameter of configuration files.In this path, the wav files for different language are stored in folders for different language. In each language folder the wav file for particular task is stored with the same name. Language folders are named with language as specified earlier. So a file for a specific language need to be searched in the folder named with the code of that specific language.

Therefore, to locate a wav file for a particular language, say tamil, we need to pass following things:

* content.location.url – configuration parameter where root location of wav files is stored
* language folder name – language code, if tamil then “ta”
* file name – since file name is same in each folder.

The end path will be :

**http://{ content.location.url }/{language code : i.e. ta}/{file name}.wav**

**Example**

Suppose there is a wav file “birth\_prep\_01.wav” and folder specified in “content.location.url” is “resource”. So the hierarchy of storing file in different language folders is as below:

**...**

The *content.location.url* parameter in *ivr.properties* file will contain

**http://www.example.com/resource/**

Thus, as in example the path for the wav file “birth\_prep\_01.wav” will be:

**http://www.example.com/resource/ta/bith\_prep\_01.wav**

# References

<http://www.motechproject.org/devdocs/installation.html>

<http://motechsuite.org/index.php/installing-motech>

<http://motechsuite.org/index.php/installing-motech>

* <http://www.motechsuite.org/>
* <http://www.motechproject.org/>
* <https://code.google.com/p/motech/>
* <http://www.kookoo.in/>