BipVote: rural Mali voting system

Hans-Dieter Hiep, Roy Overbeek & Paweł Ulita

May 28th, 2018

Abstract

We introduce an ICT application that allows partial automation of counting missed calls by rural Malinese radio listeners using existing cellular phone deployments to send bips: calling and immediately hanging up. Bips are an existing practice, we use it to implement a voting system. This application counts and stores votes and voting rounds, and allows for illiterate administration using VoiceXML technology, and is built on the existing KasaDaka platform allowing deployment in a developing country.

1 Introduction

This document describes the design and development of the *BipVote* voting system. *BipVote* allows vote organizers to set up votes in which participants can cast votes by sending bips to designated phone numbers. A *bip* is a phone call that is cancelled before it is picked up: hence, this method is free of cost for the participant.

BipVote has been developed specifically for use by Malinese radio stations. These radio stations already host votes in which participants can vote through bips, but this process is not automated, which makes it time-consuming and unscalable. The idea for *BipVote* is taken from [1].

The document is structured as follows. Section 2 describes our proposal in detail: we provide a use case description and then formulate our business model and budget requirements. Section 3 provides more background on the Malinese social context, which help motivate our proposal. In Sections 4 and 5 we describe, respectively, the design and implementation of our system. Section 6 then provides instructions for system demonstration. We discuss the system in Section 7 and summarize suggestions for improvements.

2 Proposition

In this section we propose our project, as if the reader was judging from the perspective of a funding agency. We first state our use case description in the W4RA format. We then give a business model proposal to support the project in the long run.

2.1 Use case description

Name Our project is called *BipVote*. See the title page for its associated logo. Eventually it makes sense to use a comparable name in Bambara, the most popular language in Mali.

Summary of the key idea Radio hosts in Mali sometimes want to ask listeners for feedback on a radio session. For instance, they might want to know whether the listeners understood the information that was shared. Radio is, among all other ICTs, the most deployed in Mali. Calling or texting is too expensive for the listeners, and (female) listeners may be illiterate. For this reason, listeners presently send a *bip* to a phone number representing their feedback. A bip is a call that is cancelled before the recipient picks up: it is free of charge. When the voting period is over, radio hosts manually count and deduplicate the missed calls to register the results of the voting. The purpose of the *BipVote* application is to do this automatically. Moreover, radio hosts should be able to manage votes through a voice-based application, since computer literacy might be low.

Actors and goals There are three groups of actors. Two are human actors and one system actor. The human actors are organizers of the voting, and the voting participants. The system actor is the voting system. The goal of the organizers is to let the participants take part in the voting in a cheap (even free), easy, and reliable way. The participants, on the other hand, want to be able to take part in such a voting without worrying about the costs or whether their vote is counted.

The responsibilities of the organizers are: preparing the system for voting (setting a start date), broadcasting the voting phone numbers, shutting down the voting (setting an end date), and retrieving the vote counts. Responsibilities of the participants are: finding out what the voting numbers are, and making at least one bip to at least one voting number. Responsibilities of the voting system are: waiting for incoming vote bips, and counting the bips per voting option by taking only the last vote of each user.

Context and scope The application depends on three pre-existing technologies: a radio network for broadcasting the radio session, a cellular network for transmitting bips, and the KasaDaka system for handling the incoming bips and running *BipVote*.

The main stakeholder is the radio host, since she no longer has to perform tedious and error-prone manual data processing. A benefit for the listeners is that the results will be known earlier. Additionally, voters might perceive the system to be more reliable and partially anonymous, but this would have to be assessed empirically.

The system is designed for votes in which there is little incentive for system abuse, and in which the vote options are typically binary. If these conditions do not hold, then reliability would most likely suffer.

Key Performance Indicators can be assessed in a number of ways:

- The use rate of the system could indicate user satisfaction. We have key indicators for the satisfaction of different stakeholders: the number of voters that participate, the number of voting rounds initiated by organizers, and the time since last failure of the system.
- Assuming that data on previous vote counts is available, it should be investigated whether the system improves the efficiency and the reliability of the counting process.
- The radio host and a representative group of listeners should be interviewed about the system. In particular, it is important to know to what extent the system is beneficial and what problems remain to be solved.

Use case scenario script A typical scenario is:

1. the vote organizers start a vote by either calling an administration phone number or using a web interface,
2. the vote organizers broadcast the voting topic and the voting phone numbers to the participants via radio,
3. the participants vote by making bips to the voting phone numbers received from the organizers,
4. the organizers retrieve the results from the system after the voting is finished, and broadcast the result via radio.

In Section 6 we give more detailed instructions for demonstrating our proof-of-concept for evaluation purposes.

Interaction and communication In Section 4.1 we give an overview of the call-flow graph. We also design a procedure for converting arbitrary numbers into sequences of voice samples in both English, French and Bambara language.

In Mali, history was mostly transferred orally by a griot, a historian poet of high esteem. Only later did parts of Mali start to read and write literature. This is our motivation that VoiceXML is a good technological fit for use in Mali.

An important aspect of user interaction is localization and internationalization. Localization is translating the application such that the local user can understand and use it. For example, we have translated the voice application in

French because in Mali there are more French speakers than English speakers. However, it would be best to translate the application also in Bambara: that is spoken by more than 80% of Malinese population. Internationalization is ensuring that the data formats are compatible with local cultural norms. For example, the currency used in Mali is CFA franc. Another example is date format: it is typically Day-Month-Year (contrary to some parts in Ghana, in which Year-Month-Day is more typical).

Technology infrastructure We make use of KasaDaka as base station for receiving calls; it implements the voice platform. We make use of the Django web framework in Python for handling the application logic and serving dynamic voice documents. We make use of an SQL database for storing records of the previous voting rounds. We may require file storage, for storing recorded audio fragments concerning the voting round titles.

In Section 4.3 we show the architecture of our application.

Feasibility and sustainability We think the system is feasible since it largely relies on existing infrastructures, namely the radio and cellular networks. Also, the system of voting by means of bips is already in place in Mali: the introduction of the *BipVote* changes very little for the voters. This gives reason to believe that adoption in Mali will be likely. It must be ensured, however, that all stakeholders trust the system. Relatedly, if the system is used for votes that are relatively consequential then fraud becomes a risk, with social unrest as a result.

Cost considerations We do not consider the larger infrastructure incurs a relevant cost. Radio networks are already operational in both Europe and Mali. In Mali, many (male) listeners already own a basic mobile phone. Moreover, bips are free of charge. An identified risk is that telecom providers might block the telephone numbers, in case too many non-calls are made and their cost becomes too high.

To support development and deployment of this system, we propose the business model in the next section.

2.2 Business model

Our core idea to support our application and its use cases on the long run is to operate and launch a similar service in Europe. This service is offered as a commercial service, generating revenue for supporting deployments in Mali. Our unique selling point is that European customers are directly supporting a good cause, and up to 20% of revenue is spent in Mali. We follow the standard Business Model Canvas to formally describe our business model.

Infrastructure Our *Key Activities* are: improving KasaDaka voice services. Our *Key Resources* are: developers (human), translators (human), supporters (human), equipment (physical) and existing open-source software (intellectual). Our *Partner Network* consists of: funding agencies (e.g. NGOs), coordination bodies (universities), Internet service providers (ISPs), hardware suppliers.

Offering Our *Value Propositions* are providing qualitative commercial support to businesses in Europe that make use of KasaDaka. We provide installation and deployment services, and allow businesses to request improvements. These services directly benefit both users in Europa and in Mali: we offer a product in Europe that lets the customer pay for its own installation and an additional installation in Mali. Any technological improvements are shared by both paying customers and Malinese users.

European customers are offered a mass product: a standard installation on possibly more expensive, more reliable and more capable hardware than in Mali. It is foreseeable to offer these customers standard applications they can install on top of the base system, thereby creating an ecosystem of KasaDaka applications. These paying customers are reached through partner channels, such as European ISPs, and are personally assisted. Malinese users are reached through coordination bodies.

Finances The cost structure of our project is cost-driven. The application needs to be further developed and maintained, but at minimum price. Our key resource is students: since labor cost of students is supposedly free (fixed cost). The application must be localized: this means that every aspect needs translations (fixed cost). The KasaDaka system components must be be purchased and assembled (economy of scale). Moreover, for on-location deployment, we need support staff trained to deploy and explain the KasaDaka system and its *BipVote* application (fixed labor cost, variable travel cost). Finally, an important part of the costs is the distribution and deployment in Mali (variable).

The revenue streams are mainly covered by Europese customers. This stream consists of subscription fees for technical support, and an asset sale for initially acquiring the device, and licensing customers to resell the system.

2.3 Budget

In our project proposal we include a budget estimate that covers the operational expenses for the continuation period of one year. This estimate (in Euros) is based on our current experience and excludes any travel costs.

Expenses		Incomes	
Development	4,704	Capital	2,500
Translations	2,100	NGOs	7,500
Operations	9,856	Revenues	3,000
Licensing	800	Loans	4,820
Rents	360		
Total	17,820	Total	17,820

Development includes 3 months part-time (30%) professional web developer (market rate €30/h), and is supplemented by discount free labor provided by students. Translations are provided by third party professional services. Operational costs include 8 months day-to-day part-time (20%) management (discounted rate €25/h), acquiring equipment (€800) and installation (€2000). Includes fixed licensing costs for third-party software, and rents to cover the loan (7%) for financing this budget.

Our expected revenue includes the sale and servicing of four KasaDaka installations in Europe over the course of one year. The loan is taken at the start of the project, with an expected payback term of 2 years. Capital and NGOs funds are starting capital. Since neither capital nor NGO funds are currently available, our budget deficit is estimated at €10,000.

With 20% of revenue reserved for Mali deployments, this results in covering equipment ownership or replacement costs of six installations (€600).

3 Social context

To help understand where and how *BipVote* could be used, we summarize some relevant statistics of the Malinese social context.

- Mali is among the poorest 25 countries in the world [2].
- Mali has an estimated population of $17,885,245$ (2017 est.) [2].
- Only 26% of the population has access to electricity. This figure is 53% in urban areas and only 9% in rural areas (2016 est.) [2].
- Internet usage is restricted to 11.1% of the population (favouring the elite) while $20,182,160$ cellular phones were estimated to be in use (2016 est.) [2, 3].[1]
- Of all media, radio is by far the most popular. Radio stations are often serving small local communities, but in a large city like Bamako 80,000 listeners may be reached [3].
- Mali is home to a wide variety of languages. An extensive list is available at [5]. French is the official language, but it is not the most widespread: only 2,995,000 are estimated to speak French (2016 est.), of which only 15,000 speak French as their first language (2013 est.). Bambara (listed under the synonym Bamanankan) is by far the most popular language with 14,000,000 speakers. Of these speakers, 4,000,000 have Bambara as their first language (2012 est.).
- 33.1% of the population is literate (2015 est.) [2]. This figure is biased towards men: 45.1% of men are literate, against 22.2% of women.

In summary, resources usually assumed in ICT projects (such as power and internet connectivity) are largely unavailable, mobile phones and radio stations are abundant, people are poor, literacy rates are low, and the variety of spoken languages is large. These factors help explain why *BipVote* is designed as follows:

- a voice-based application that is heavily reliant on mobile telephony: mobile telephony is the highest penetrated personal communication technology in Mali, thus finds many users.
- our application intends to empower radio broadcasters, since radio is the most penetrated broadcasting technology in Mali,

[1]We note that both the usage of cellular phones and internet are increasing rapidly: in 2012 there were estimated to be only 56 cellular phones and 4 internet users per 100 inhabitants [4].

- by employing recognition of bips, our application is cost-effective for end-users that wish to vote. A bip is placing a call and directly canceling it. The use of bips is already a common practice, also among Malinese inhabitants, so adoption is likely.
- Our application is suitable and configurable in the field for multiple languages: English, French and Bambara. This allows one to adapt the voice samples to be changed in the field, to adapt to the local inhabitants' accent.

3.1 Logo

The BipVote logo (see cover page of this document) contains an abstract representation of the head of a Bambara fertility statue. The motivation behind this design was to integrate a Malinese cultural element into the BipVote logo. We have two reservations regarding this design choice:

1. This decontextualisation of the fertility statue might be confusing to some Malinese, or even deemed morally inappropriate.
2. Mali is home to a wide variety of peoples. Many of these peoples might feel excluded by this design choice.

We note that there is also positive evidence supporting our design choice:

> As the twentieth century drew to a close, scholars increasingly questioned the validity of assuming a one-to-one correspondence between a cultural or ethnic group and an art style. Indeed, historical documentation indicates that antelope crest masks and other arts labeled as Bamana have transcended linguistic, religious, and cultural boundaries since at least the end of the nineteenth century [6].

However, we submit that field work must ultimately decide the matter. Should the logo require a redesign, then the statue could for instance be replaced with a representation of a vote being cast.

4 Design

This section presents the design of our system. This includes 2 types of interfaces and an architecture. We will start with the voice interface which plays the central role in communities with high illiteracy rates. Later, we will move to the web interface: a more advanced, but also more flexible option for literate people with internet access. Finally, we will reach the architecture according to which the system is built.

4.1 Voice interface

The voice interface is the central point of our system. Its purpose is to provide an easy access to *BipVote* for illiterate users through a mobile phone. We designed 2 ways of interacting with the system to address the needs of 2 types of users: an interactive audio interface for radio hosts and bip recognition for voters.

Radio hosts This interface provides a basic set of actions that a radio host would like to perform. The diagram of all possible paths through the voice interface is presented below. To ensure integrity and ease of use, it is never possible to make an action which doesn't make sense, and users are promted for confirmation before making final decisions. To summarise, a radio host can:

- create a new and end an existing poll,
- get current and past results.

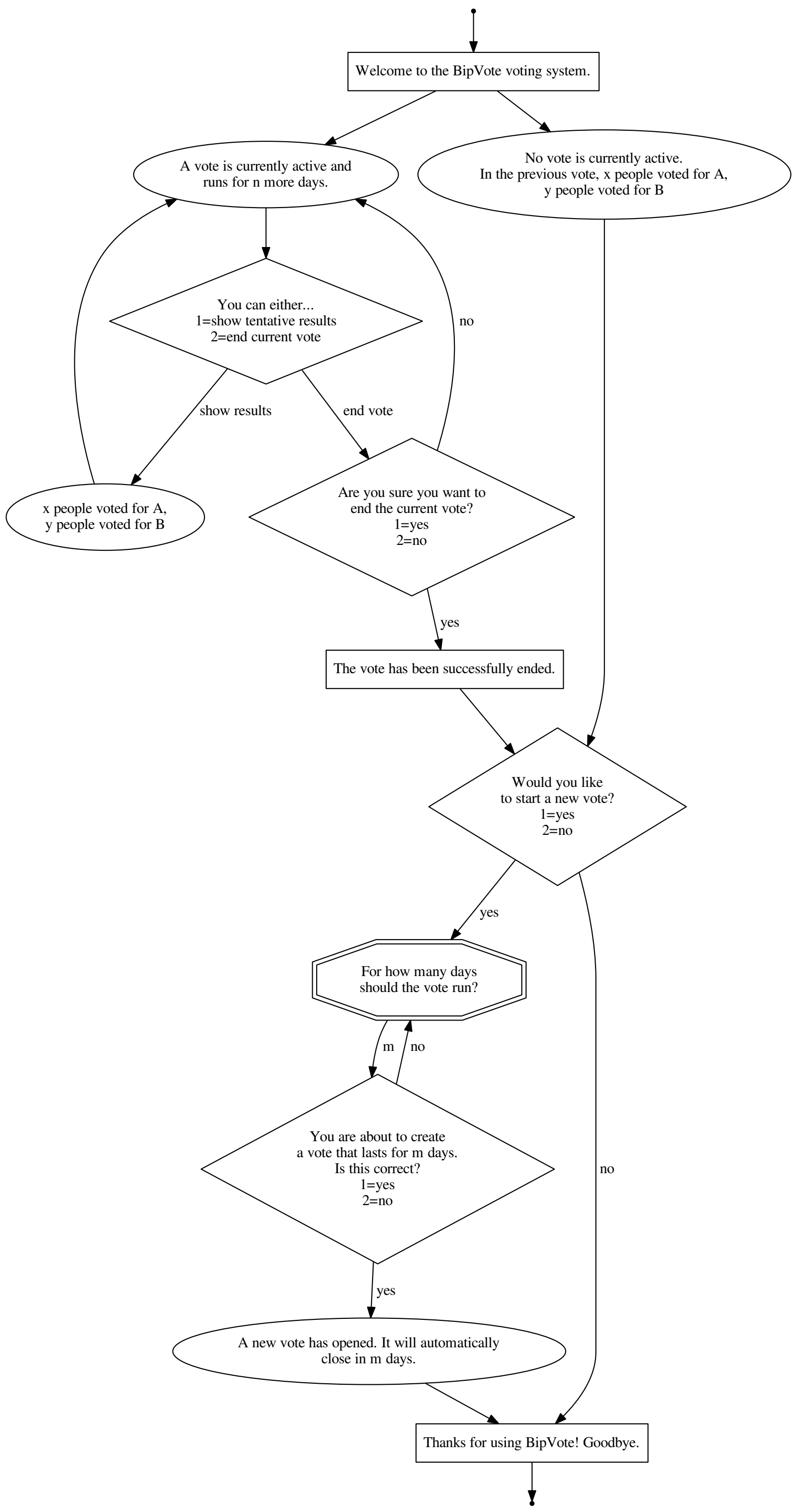
Even though such a way of managing the system is desirable for our target group of radio hosts, it comes with a certain set of restrictions. The most imporant are:

- all created polls have exactly 2 options,
- duration can be specified only in days,
- fetching the results of previous polls can be done only for the last poll, and when there's no active poll.

Those restrictions are lifted in the web interface.

Bip recognition This interface does not involve any actual voice interaction. However, voters are able to cast votes by making bips, i.e. using their mobile phones to start making a call, and hand up before the system picks it up. The system recognizes that a bip was made, and saves it in the database as a vote.

Unfortunately, due to the current restrictions of our KasaDaka instance, making a bip means casting a vote to a random option. This is due to the fact that only one phone number is available to us, and we have no means of distinguishing between vote options.


Welcome to the BipVote voting system.
A vote is currently active and runs for n more days.
No vote is currently active. In the previous vote, x people voted for A, y people voted for B
You can either... 1=show tentative results 2=end current vote
no
show results
end vote
x people voted for A, y people voted for B
Are you sure you want to end the current vote? 1=yes 2=no
yes
The vote has been successfully ended.
Would you like to start a new vote? 1=yes 2=no
yes
For how many days should the vote run?
m
no
You are about to create a vote that lasts for m days. Is this correct? 1=yes 2=no
no
yes
A new vote has opened. It will automatically close in m days.
Thanks for using BipVote! Goodbye.

4.2 Web interface

BipVote also is accessible through a web interface. We explain the design of this interface in relation to its intended users, of which there are two designated user groups. The first user group is the admininistrtor, whose primary purpose to manage other users and to configure the voice interface. The second user group is the radio host, whose primary purpose is to manage polls and to read the results of polls.

Other user groups are possible, for example users who can only read information about polls. The admin can create new user groups on the fly, possibly in discussion with the radio station.

The admininistrator user group The admin has access to all functionalities of the web interface. Related to user management, the admin can perform generic user management tasks, such as creating and deleting users, configuring user permissions, assigning users to groups, and creating and deleting groups (a group defines a set of permissions for users belonging to it). In addition, the admin can define *voice labels*, *elements* and *voice services*:

1. A voice label is a phrase (such as a number or a question) which the admin can associate with spoken word samples in multiple languages.
2. An element is like a node in a call flow graph. The element can be associated with a voice label, which determines which sample is played in a call. The element can also have certain behaviour, such as reading or writing to a database, or branching to other elements based on a caller's input. Most elements have some successor element: if they don't, the call is terminated afterwards.
3. A *voice service* consists of a starting *element* (e.g. an element representing a welcome message), and a small number of configurations, such as which languages are supported by the voice service, and whether the language preferences of callers should be remembered.

The BipVote voice interface (as designated in Section 4.1) is an example of a voice service, so it can be modified by an admin.

Radio host user group While the radio host could in principle only use the voice interface, we also regard them as potential users of the web interface. The reason for this is that a web interface allows for much richer functionality than a voice interface: the latter would become unusable as more complex user interactions are added. Moreover, we expect that a non-negligible proportion of radio hosts is literate and has access to a computer, meaning that this use case is not redundant.

Two aspects are of interest to the radio host: (improved) poll management and data analysis. Regarding poll management, the radio host can create polls, modify their starting date, modify their duration, set the number of vote options, and document (in writing) what the question and options are. (Contrast this with the poll management supported by the voice interface, in which only binary polls can be created that become immediately active, and which retain no information about what the question and options are.)

Regarding data analysis, the radio host can at any time view the outcomes of any poll. Moreover, we have considered ways to present additional information about the polls. At the very least the information needs to be understandable for a Malinese radio host, so we have refrained from showing statistic confidence intervals (which, for an other audience, might be an insightful statistic to compute).

At the moment we have designed:

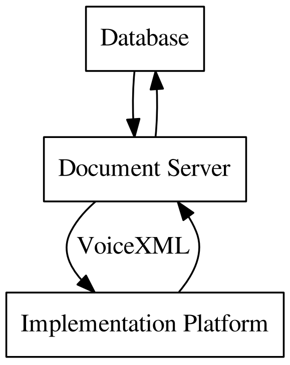
- A pie chart that plots the outcomes for a poll. Compared to raw numbers, the pie charts makes it easier to understand the proportions between the outcomes, especially if distinct polls (with different vote totals) are compared.
- A time chart that plots time against the cumulative number of votes per option. This makes it easier to see when people voted, so that the chart can be correlated with the broadcast. In addition, the time chart can be used while the broadcast is ongoing, since it supports live updates. This adds a layer of interactivity to BipVote, since now radio hosts can already get a sense of what people are thinking *while* they are broadcasting.

The visualizations could in principle also be made publicly accessible, so that those rare Malinese listeners with an internet connection (a number which is growing) can also view the results in an intuitive way.

4.3 Architecture

In our technology architecture we have the following assumptions, given as system components:

- The KasaDaka system, which implements handling incoming calls and registering bips (Implementation Platform).
- The Voice Service Development Kit, a fork of Django that implements the basics for generating VoiceXML documents (Document Server).
- A SQL database, a relational database storage backend (Database).

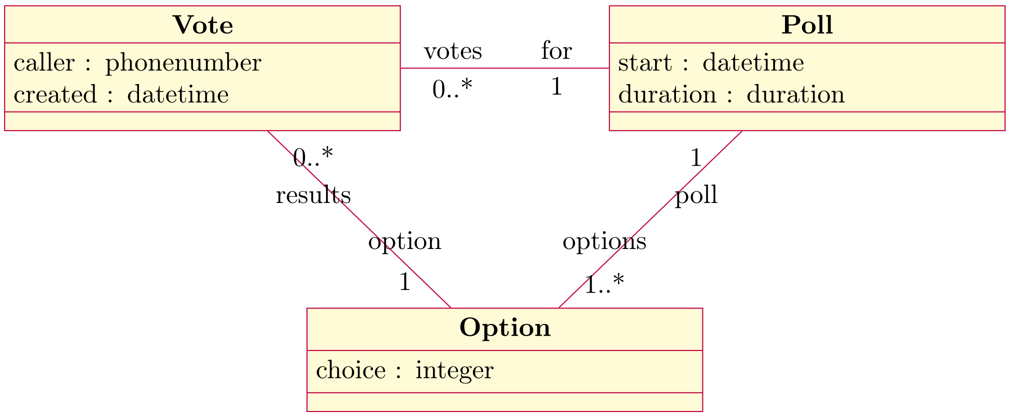


Our system consists of a database, a document server and an implementation platform. Whenever a call is made, the implementation platform receives the call and acts as user agent for the caller. A document is requested from the document server, and a VoiceXML document is served. The returned VoiceXML document contains a declarative description of the call: what audio to play, what menu choices the user has, or what form elements can be filled. Schematically, this is depicted on the right.

The document server is realized as an HTTP Server, on top of which we deploy an application that serves the back-end logic of our system. This allows us to dynamically generate VoiceXML documents with up-to-date and contextualized content, by interacting with the database component.

The voice dialogue design that has been given in the previous section is implemented as a set of linked Web resources. The links between web resources resemblences the call-flow structure.

Data Model An additional consideration for the system design is the database schema that specifies what information is to be stored. We specify it as follows, using an UML class diagram.



We now give a description of what the UML diagram depicts.

Each poll object consists of a start date and a duration. A poll is active if the current time is less than the start date plus the duration. To cancel a poll, the duration is assigned such a value that the poll no longer is active. To each poll there is a non-empty set of options associated. Every option has precisely one poll for which it counts as an option. A vote contains a caller identification, a timestamp, is related to precisely one poll and one option. The relation between vote and poll is implicitly transitively defined, since the poll it is related to can be infered by the relation: for = option ∘ poll. Similarly, the votes of a poll is also an implicit relation and can be defined as: votes = options ∘ results.

In addition, the votes relation filters out duplicate votes by the same caller. Clearly, if callers can call anonymously by disabling caller id, we can no longer track who called. All anonymous callers are counted only once, i.e. at most one vote is assigned to all anonymous callers.

We designed the models such that only the bare minimum amount of information is stored. This is for basically a cost consideration: a more complicated or sophisticated system has higher maintenance cost, and is less easily understood by peers who might want to continue work on it.

5 Implementation

5.1 Voice Dialog

The audio interface implementation uses the VSDK framework originally implemented by André Baart [7] and available on GitHub [8]. The VSDK framework is based on the Python web framework Django [9]. It enables developers to easily define a call flow by using high-level building blocks mapped directly to VoiceXML elements. The framework defines a certain set of VoiceXML elements as Django models. It means that instances of those elements are stored in the database, and can be easily modified without changing the code. What is more, links between those elements can be represented as foreign keys in the database.

For the designed call flow we had to implement a selection of new elements (called *models* in the Django nomenclature). Our elements extend the standard VSDK elements by at least one of the following:

- adding more audio recordings,
- adding more links to other elements,
- extending actions happening when the element is requested.

Our implementation is at `vsdk/polls/models/custom_elements.py`.

Besides the elements themselves, it was necessary to implement customized functions (called *views* in the Django nomenclature). They are responsible for handling the extended logic related to those elements. They can be found at `vsdk/polls/views.py`.

The most important custom elements are:

- `PollDurationPresentation` for telling how much is remaining,
- `AskPollDuration` for asking users for the duration of a new poll,
- `CreatePoll` responsible for actually creating a poll,
- `EndPoll` responsible for ending an active poll.

`PollDurationPresentation` is the most peculiar element, because it uses our algorithm for generating numbers. This is implemented as an extension to VSDK. Given a language, such as English, French or Bambara, the extension allows programmatic conversion of numbers into spoken representations. The representations consists of a large number of samples that are automatically combined to form large numbers.

For example 152 becomes one-hundred-fifty-two (4 samples) in English, and cent-cinquante-deux (3 samples) in French, and keme-ni-bi-duuru-ni-fila (6 samples) in Bambara.

For converting numbers into a sequence of voice samples, we investigate here French [10] and Bambara [11], and skip English. We assume that the reader already knows how to count in English.

In French, digits from 1 to 16 are specific words. The words for 17 until 19 are regular, namely 10+7 up to 10+9. The tens between 20 and 60 are specific words, and between tens we have regular words: concatenate the tens by the unit. For 61 until 99 a base 20 is used: for 61 until 79, we use 60 concatenated with 1 until 19. For 80 we use 2*40 ('quatre-vingts') and for 81 until 99 it is again 80 concatenated with 1 until 19. The word 'et' is prefixed whenever 1 is said last in a concatenation.

In Bambera, digits from 1 to 9 are specific words. The 10 and 20 are special, but then 30 up to 90 are the words for 3 up to 9 prefixed by a 10-multiplier. For 200 up to 900, we have another 100-multiplier prefix but this time it works also for 1 and 2: for 100, the prefix is the sole word. This works regular for thousand-multiplier and million-multiplier. Finally, each group (unit, tens, hundreds, et cetera) is separated by a word 'ni', most-significant groups first.

5.2 Web interface

The web interface is implemented using the Python web framework Django [9]. Django provides many generic web project requirements out of the box, including user management functionalities, security features and an admin panel. It is also highly portable, which is particularly relevant for an ICT4D project.

The VSDK (and our extension of it), explained in the previous section, is just a Django project. The Django-provided admin panel makes it very easy to create, modify and delete particular VSDK elements, which in a deployment can be stored in a PostgreSQL database.

The pie chart and time chart visualizations are additions to what is already provided by Django. The time chart is already integrated into the admin panel. The pie chart has not yet been integrated, but can be viewed at `viz.html`, located in the `visualization` project directory. To generate the visualizations, we use Highcharts, a JavaScript library that can be used for rendering a great variety of graphs. We chose Highcharts for a couple of reasons:

- JavaScript in general integrates well with our web framework.
- Highcharts is designed to be highly compatible with older browsers [12], which is especially relevant in the field of ICT4D. The Highcharts development even claims compatibility with Internet Explorer 6.0. If a browser does not support certain features (such as 3D rendering), Highcharts uses fallback mechanisms, so that Highcharts's high compatibility does not limit its design space. Compare this with e.g. D3, which is one of the most popular JavaScript libraries for visualization. D3 generates SVG images, which have only been supported by Internet Explorer since version 9.0, and even then there are scaling issues [13, 14].
- Highcharts is free (provided that it is used non-commercially).
- Highcharts supports many functional (and aesthetically pleasing) graphs right out of the box. The graphs also include additional useful features, such as zooming and exporting images in a variety of formats. By contrast, something like D3 is relatively low level and therefore not as suitable for our rapid prototyping approach.

An extensive overview and comparison of visualization tools is found in [13].

We have tried to make the graphs highly backwards compatible. To this end, we have used the following tools:

- *W3C Markup Validation Service* [15]: HTML5 and CSS3 are relatively new web technologies (2014 and approximately 2011, respectively). We have translated all HTML5 and CSS3 to HTML4 and CSS2, and validated the source code using the official W3C validators.
- *Babel.js* [16]: ECMAScript is the specification language which JavaScript implements. ECMAScript 6 was published in 2015, and it is supported by most of the newer browsers. Older browser versions, however, might only be compatible with ECMAScript 5, which was published in 2009. To improve browser compatibility of JavaScript source code (possibly containing ECMAScript 6 constructs), Babel.js transforms ('transpiles') any ECMAScript 6 *syntax* into syntax which is ECMAScript 5 compliant.

- *Polyfill.io* [17]: Apart from syntax, ECMAScript 6 might also introduce library functions that are not supported by ECMAScript 5. Such functions can be *polyfilled*, meaning that they are implemented in a way that is ECMAScript 5-compliant. Pollfill.io does exactly this.

In the end, however, the generated result (found at `visualization/viz-compatible.js`) unfortunately did not work even in Internet Explorer 11 on a Windows 7 system (to this end we used Sauce Labs [18], a cloud based service for cross browser testing). No debugging information was available either. We are not sure how much work is still required for making it functional, but we leave it as future work.

There is also an issue with displaying the correct time stamps in the time chart. We also designate resolving this bug as future work.

5.3 Deployment

The actual application is intended to run on the same Raspberry Pi 2B as the KasaDaka system. However, as part of our prototyping experience, we have deployed the application on a Heroku virtual server instance.

In a real deployment setting, the KasaDaka system must be extended with multiple telephony dongles to allow multiple vote options, one associated with a unique phone number for voters to call. This requires an external USB hub, that also requires external power supply, since the Rapberry Pi board does not have enough power to drive two or more of such dongles.

6 Demonstration

As the process of installation of our system is not trivial, although reproducible, we have provided user access to a running instance of BipVote at `http://bipvote.ml`. In the sections below, we present detailed instructions of how to configure and test it.

Installation instructions for Windows can be found in a YouTube video:
`https://www.youtube.com/watch?v=4Ox5RJkMnq0`

A video walk through of the web administration system and recognizing a bip can be found in a YouTube video:
`https://www.youtube.com/watch?v=mxtRCMht0qg`

To perform the configuration, a running KasaDaka instance is necessary. We assume that the configuration of the KasaDaka instance can be done in the same way as during the course.

6.1 Configuration and administration

Configuration can be split into 2 parts: configuring the KasaDaka instance, and configuring the voice service itself.

KasaDaka configuration Configuration of the KasaDaka instance can be done at `http://ict4d.kasadaka.com`. The login is: `mali`, and the password is: `bamako`. The configuration process requires two URLs specific to our service: the main URL of the service and the "bip" URL. You can find them here:

- VoiceXML URL: `http://bipvote.ml/vxml/start/3`
- Bip URL: `http://bipvote.ml/polls/bip/3`

Voice service administration Accessing the service from the administrator's perspective can be done via `http://bipvote.ml`. The admin username is: `admin`, and the password for this user is: `AhNgotheiM6wu9j`. (The dot is not part of the password.)

The administrator has access to everything that is in the database. In particular, it includes changing the call flow by editing elements in relevant sections, and accessing polls, vote options, and votes.

A special account for a radio host can be created by creating a new user, setting its "staff status" to active, and adding him to the "Operator" group. All this can be done by clicking "Add" next to the "Users" category in the main screen of the admin panel. The current configuration already contains a radio host account. The username is: `bill`, and the passwords is: `radiohost7`. (Again, the dot is not part of the password.)

Such an account has a very restricted access to the system. It means that only managing polls and vote options is possible. Votes can be seen only as a live graph, and modifying them is not allowed.

6.2 Using the voice interface

After the KasaDaka configuration is complete, the voice service can be accessed by calling this landline number: `+31 (0)20-3697664`.

The voice interface will first asks to choose a language, and then redirects to the main menu of the service. The call flow can be found in the voice interface design section.

6.3 Bips

To see a live presentation of how bips work, an active poll has to exist. This can be done via the web interface or via the voice interface. We recommend using the voice interface, because it gives a very constrained list of options that can be done, and makes sure everything is in place.

After making sure that KasaDaka is configured and there is an active poll, it's possible to make a bip. It means calling the aforementioned number, and hanging up as soon as the calling signal can be heard. Unfortunately, as we have only one available number, making a bip means casting a vote for a random option.

The bip will be recognized in around 60s by the system. This means that a new vote will be created from the number of the caller. The new results will be available right away through both interfaces. We recommend keeping the poll's edit page open while making bips, because the graph is updated dynamically whenever a new vote is cast (i.e. a bip is made).

It is worth keeping in mind that only the last vote from a certain number is taken into account. It means that the second or later bip from one number can make no difference, because the chosen vote option can be exactly like the previous one.

7 Discussion

To ensure high quality of our application, a handful of automatic tests were introduced. Those tests try to automate checking whether some non-trivial parts of our system are behaving correctly.

Our system depends critically on one assumption: the KasaDaka system should implement bip recognition. We have identified a number of issues with the underlying platform, which have to be resolved over time for *BipVote* to become usable.

- The KasaDaka system has a high burden of system administration work to set up. This makes it hard for developers to reproduce the system and improve it. Our recommendation is to configure a disk image that can run in a virtual machine, which allows developers to access and test the system locally. This image should not require an actual phone line or GSM dongle, but could work with a softphone such as Ekiga.
- The KasaDaka system is deployed on an Raspberry Pi 2B board, which does not have enough power to support multiple GSM dongles. We need one GSM dongle for each phone number that is supported. To solve this problem, we should look into using different boards that allow for a higher power supply (e.g. a Banana Pi) or an externally powered USB hub.
- KasaDaka employs many open source components. This allows developers to independently improve and develop the system, free from licensing restrictions and free from dependency on properietary support. This includes the Django framework, the Python runtime environment, SQLite database, the Asterisk private telephone exchange, the Ekiga softphone package, and the Raspbian operating system. However, a critical component of the system, the VoiceXML interpreter implemented as an Asterisk module, is not provided for free: the source code is not open, and thus restricts developers from improving that part of the system. This imposes a long-term risk of the project. A potential solution is suggested in [19].

There is still quite some work to do. In a continuation of this project, the following issues should or could be addressed:

- Bambara samples should be added to the system.
- The pie chart visualization should be integrated into the admin panel.
- The visualizations of the web interface could be made more backwards compatible with older web browsers.
- The time stamps in the time chart are currently not correct: the responsible bug should be found and fixed.
- Poll information could be stored in an open data format, such as JSON-LD. This data can then be made publicly accessible. Relatedly, a poll import functionality could be added to BipVote, so that a poll held at one station could be repeated at another station. By collecting the results of such linked polls, the coverage of a poll can effectively transcend the level of individual radio stations.

8 Conclusion

We have presented BipVote, an ICT application that supports Malinese radio hosts in hosting polls over a radio broadcast. More specifically, BipVote allows radio listeners to cast their vote in a cost-free way by sending bips to designated phone numbers representing vote options, and the BipVote system will automatically process these bips and make the poll results available to the radio host. BipVote is designed to run on the flexible KasaDaka hardware system.

Radio hosts can either access BipVote through a web interface or through a voice interface. The web interface exposes all functionalities of BipVote, including poll histories and informative visualizations of poll results. The voice interface is limited in functionality (for the sake of usability), and is intended especially for (digitally) illiterate radio hosts.

While BipVote is usable in its current state, it still has some limitations. Suggestions for improvements include adding Bambara audio samples, refining the visualizations and making the system compliant with linked open data requirements. If development is continued, we also urge that the application is tested with prospective users, to ensure that it matches the needs and skills of the user. As of yet, there has been no opportunity to perform such tests.

9 References

[1] A. Bon et al. Use cases for ICT4D course, 2017.

[2] DC: Central Intelligence Agency Washington. The world factbook 2018. https://www.cia.gov/library/Publications/the-world-factbook/geos/ml.html, 2018. Accessed: 03-05-2018.

[3] Freedom House. Mali: Freedom of the press 2016. https://freedomhouse.org/report/freedom-press/2016/mali, 2016. Accessed: 03-05-2018.

[4] CountrySTAT. CountrySTAT: Mali (key indicators). http://mali.countrystat.org/key-indicators/. Accessed: 03-05-2018.

[5] Ethnologue. Mali. https://www.ethnologue.com/country/ML/languages. Accessed 03-05-2018.

[6] Susan Elizabeth Gagliardi. Antelopes and queens: Bambara sculpture from the Western Sudan: A groundbreaking exhibition at the Museum of Primitive Art, New York, 1960. https://www.metmuseum.org/toah/hd/bamb/hd_bamb.htm, 2016. Accessed: 23-05-2018.

[7] Kasadaka: a sustainable voice-service platform. https://w4ra.org/wp-content/uploads/2017/12/Master-thesis-andre-baart_Dec2017.pdf.

[8] Vsdk implementation. https://github.com/abaart/KasaDaka-VSDK.

[9] Django. https://www.djangoproject.com/. Accessed 25-05-2018.

[10] French numbers – of languages and numbers. http://www.languagesandnumbers.com/how-to-count-in-french/en/fra/. Accessed 05-05-2018.

[11] Bambara numbers – of languages and numbers. http://www.languagesandnumbers.com/how-to-count-in-bambara/en/bam/. Accessed 05-05-2018.

[12] Highcharts: compatibility. https://www.highcharts.com/docs/getting-started/compatibility. Accessed 26-05-2018.

[13] Scott Murray. *Interactive Data Visualization for the Web: An Introduction to Designing with D3*. O'Reilly Media, Inc., 2017.

[14] Can I use... https://caniuse.com/#feat=svg. Accessed 26-05-2018.

[15] W3C markup validation service. https://validator.w3.org/. Accessed 26-05-2018.

[16] Babel. https://babeljs.io/. Accessed 26-05-2018.

[17] Polyfill service. https://polyfill.io/. Accessed 26-05-2018.

[18] Sauce labs. https://saucelabs.com/. Accessed 26-05-2018.

[19] Lerato Lerato, Maletšabisa Molapo, and Lehlohonolo Khoase. Open source voicexml interpreter over asterisk for use in ivr applications. *Proceedings of the SATNAC*, 2009.