Creating a flexible voice service framework for low-resource hardware: extending the KasaDaka

January 2016

Bachelor Thesis
Computer Science
VU University Amsterdam

Author

André Baart (2504427)

abaart@gmail.com

Supervisor

Victor de Boer

v.de.boer@vu.nl

Abstract

In developing (rural) communities, the adoption of mobile phones is widespread. This allows information to be offered to these communities through voice-based services. This research explores the possibilities of creating a flexible framework for hosting voice services in rural communities. The context of the developing world poses special requirements, which have been taken into account in this research. The framework creates a voice service that incorporates dynamic data from a data store. The framework allows for a low-effort adaptation to new and changing use cases. The service is hosted on cheap, low-powered hardware and is connected to the local GSM network through a dongle. We validated the working and flexibility of the framework by adapting it to a new use case. Setting up this new voice server was possible in less than one hour, proving that it is suitable for rapid prototyping. This framework enables further research into the effects and possibilities of hosting voice based information services in the developing world.

Table of contents

1 Introduction

1.1 Context

In the 'Western world' we know today, Information and Communication Technologies are used by almost all of the population. These technologies have revolutionized the way of working and communicating in the modern world. The economies of countries with a high degree of digitalization are extremely dependent on ICT.

In developing countries and specifically in rural communities in those countries, this is not necessarily the case. There are challenges that prevent ICT from propagating in these areas include: poor (digital) infrastructure, lack of resources and poor education (illiteracy). Because of this *digital divide* (Fuchs & Horak, 2008), the gap in growth of economies between urban and rural areas has grown. This in turn causes a decline in welfare of the rural communities.

The research field ICT for Development (ICT4D) is aimed at researching the possibilities and effects of introducing ICT to the developing world.

The promoting of usage of 'traditional' ICT was the initial focus of ICT4D. These efforts were not very successful, as they turned out to be unsustainable in the developing world context. While these technologies thrive in highly developed countries, they do not necessarily satisfy the needs of a developing country (Toyama, 2010). The focus of ICT4D has since shifted to technologies that are already available and have a large user base in the developing world (Heeks, 2008). This is also known as ICT4D 2.0.

1.2 Motivation

A technology that has seen a significant growth in adoption in the developed world (Africa in particular) is the mobile phone. "mobile phone subscriptions on the continent have risen from over 16 million in 2000 to 376 million in 2008 – or one third of sub-Saharan Africa's population" (Aker & Mbiti, 2010).Because of this widespread adoption, mobile telephony is an interesting technology in the field of ICT4D.

Because of this widespread ownership and usage of mobile phones, voice based information systems are able to provide access to services and information to rural communities. This fits the scope of ICT4D 2.0 very well, as these systems use technology that has already been adopted by the population.

Past research has shown that rolling out voice services in rural Africa has proven to be successful. These services are a first step in bringing the World Wide Web to these communities. (Gyan et al., 2013) These voice services can be hosted locally, on affordable and low-end hardware. (Mantel, 2014) Unlike traditional web services, voice services also allow the illiterate to access the offered information. (Chhetri, 2013)

In this research, the existing voice platform (KasaDaka) will be extended to be suitable for many use cases, as found in a rural African context. The aim is to create a flexible framework that can easily be adapted to changing usage requirements.

1.3 Problem statement

This research is focused on extending the KasaDaka platform. KasaDaka is currently a platform to host a static and simple voice service. This service is implemented into the configuration files of the

telephony software (Asterisk). This severely limits the use of the service for different use cases. The aim of this research is to allow the platform to be used as a rapid-prototyping platform. This means that the platform is adaptable to new use-cases with low effort. This enables the KasaDaka to be modified *in the field* by local engineers. This makes it possible to use KasaDaka in many different situations, enabling more rural communities to get access to information and to get acquainted with ICT.

The aim of this research is to set-up an efficient and simple platform for providing independently hosted voice services. The platform will be used to extend the existing KasaDaka project, and will allow further research on various types of usage scenarios.

The KasaDaka project[1] is aimed at providing information to people living in rural communities. Most of the research is done at the Vrije Universiteit Amsterdam. The project builds on information acquired by the Web Alliance for Regreening in Africa (W4RA[2]). By setting up an affordable and low maintenance voice service, rural communities can maintain their own voice information service. Locals can call in to the service with their phones, and exchange information. The service is also available to low-literate users, as it is voice-based. This enables a cheap way of exchanging information in rural communities, which was not possible before.

Small-scale experiments with KasaDaka have been done in Ghana (Lô, 2014) , Mali[3] and Burkina Faso. Past use-cases deployed on the KasaDaka platform include a marketplace service (de Boer et al., 2015), and diagnosing animal health (DigiVet). The marketplace service has also been combined with a traditional radio broadcast. Users were able to call the service and post an offer of products they had for sale. All the current offers were then combined in a communiqué, which was broadcast on local radio. (de Boer et al., 2012)

In this research, we will first determine the requirements (section 2). Then we will decide which system components are needed to meet the posed requirements (section 3). The framework will then be implemented and validated (section 4). After checking whether the framework meets the set requirements, we will place the research in context, and discuss the current limitations and possible future work to extend this research (section 5).

[1] http://kasadaka.com/
[2] http://w4ra.org
[3] https://videbo.wordpress.com/2015/10/15/inspiring-tmt-workshop-in-bamako/

2 Requirements

As this research extends the KasaDaka project, it shares some of the same restrictions and requirements. Key problem factors concerning ICT in rural development areas include poor connectivity, poor infrastructure, lack of resources, shortage of expertise and poor education (Lô, 2014). To ensure the KasaDaka project fits well within the ICT4D 2.0 specifications, the project has to involve these problem factors in the system design.

Maintained by the rural communities

As there is a shortage of expertise and infrastructure to rural communities is lacking, reliable and high-speed connections to datacenters are not realistic. This means that the system cannot be run from a data-center or from abroad. By promoting local ownership, the service becomes more integrated into the community.

Run on low-powered, cheap hardware

Farmers in rural communities have low incomes. The system has to be affordable to be applicable to rural communities. The costs of the hardware thus must be minimal.

Electricity supply is unreliable, thus a battery backup is necessary. In the African environment, solar power combined with battery backup is preferable to ensure independent operation. This is only possible when the system has a low energy consumption.

No internet access

GSM (Global System for Mobile Communications) is often is the only possible way of (affordable) telecommunication in rural areas. This means that the system will not be assumed to always be connected to the internet. The system has to be able to run solitarily indefinitely. Remote maintenance through an internet connection is also impossible.

Has to use free software and open standards

As the KasaDaka project has to be as sustainable and affordable as possible, there is no budget to spend on commercial software. As maintenance has to be kept to a minimum, open standards are preferred, as they make it easier to maintain the system.

Voice based

The system has to be accessible by the entire rural population. A large percentage of the rural population suffers from a lack of education. To ensure the system is usable by the illiterate, the user interface has to be entirely voice based.

Language independent

Local tribes and communities often have their own languages, that are not spoken anywhere else. The system has to be accessible in many languages, and has to be extendable to include new languages with minimal effort.

Adaptable to new use-cases

The system has to address the needs of every individual community. Each community has different demands in terms of functionality. To minimize set-up cost, the system has to be adaptable to new use-cases with minimal effort.

3 System architecture

The design of the system builds on the previous research done for the KasaDaka project[4].

The aim is to build the system using known and supported standards where possible. This way the system has a modular design, making the system independent from any single vendor and highly flexible. The current KasaDaka system is a low-powered PC that is connected to the local GSM network. The user can use his existing phone to call the number belonging to the KasaDaka. The system then answers the call and provides the voice service.

The resulting system has to comply to the limitations and problems as described in section 2.

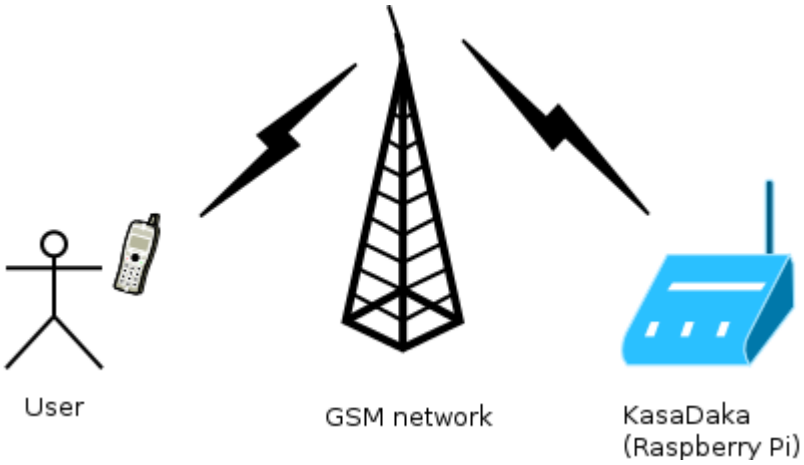

Figure 1: Model of the KasaDaka system

3.1 Design decisions

Hardware: Raspberry Pi 2 and Huawei E169

For the hardware, a Raspberry Pi 2[5] is used. Past research (Mantel, 2014) has shown that running a voice server on low-powered hardware is possible. In this past research a Raspberry Pi 1 is used. The second iteration of the Raspberry Pi offers about 2 to 4 times more performance[6], enabling the implementation of more resource intensive use-cases. The Raspberry Pi 2 is also affordable (approximately 40 EUR) and available worldwide. Because of the popularity of the Raspberry Pi, there is a large amount of accessories and add-ons for sale. These include cases, small TFT screens, solar panels and batteries. These add-ons make it possible to adapt the system to the circumstances often found in rural areas. The Raspberry Pi has plenty of USB ports to connect a GSM dongle, and has a low power consumption[7]. Power is supplied through a micro-USB connection; a lot of standard

[4] https://github.com/WorldWideSemanticWeb/KasaDaka
[5] https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
[6] https://learn.adafruit.com/introducing-the-raspberry-pi-2-model-b/performance-improvements
[7] http://raspi.tv/2015/raspberry-pi2-power-and-performance-measurement

mobile phone chargers can be used for supplying power. The Raspberry Pi can run various operating systems, of which Linux is the OS best suitable for the project. Raspbian (a derivative of Debian) is tailor made for the Raspberry Pi, and supports a lot of packages.

To provide a GSM connection, a GSM dongle is used. There are a lot of dongles available, but not all are supported by Linux and Asterisk. A USB GSM dongle that is well supported, is the Huawei E169 (approximately 20 EUR).

Telephone exchange (PBX): Asterisk

To receive and further process calls coming in through the GSM dongle, telephony software is required. Asterisk[8] is an open-source Private Branch Exchange (PBX) telephony software program. It allows connected phones to call each other using VoIP, and also supports external connections (in this case the GSM dongle). It is well supported and has a large amount of possible functions. The program is designed to be used on Linux. Asterisk is suitable to run on very low-end hardware, such as consumer routers. The program is extendible by adding modules that can provide a specific function that is not included in the base system.

Voice application document standard: VoiceXML

In the past iteration of KasaDaka, the Asterisk configuration files were used to create the voice application. The application was created by playing audio files in a pre-defined order, and on a certain user-input, a subdialog was triggered. This meant that the application had static audio, and was largely set-up in a linear fashion. In order to allow for more advanced use-cases, the program logic has to be dynamic, and linked to a data store. In order to support this, a development language is needed. VoiceXML[9] is a document standard for voice applications, designed by the World Wide Web Consortium (W3C). The working of VoiceXML is comparable to the way a web page (in HTML) is written. The format is based on the Extensible Markup Language (XML). The standard supports interactive voice dialogs between the computer and the user.

VoiceXML gateway: I6NET VoiceXML IVR (VXI)

In order to connect our VoiceXML document files to the Asterisk system, we need an gateway. The gateway interprets the VoiceXML files and presents it to the user in the form of a call through the telephony system (Asterisk). In order to do this, it may include Text to Speech and Automatic Speech Recognition software. There are various systems that can fulfill this task. Most of these programs are commercial and not available for free, which would increase the cost of the KasaDaka significantly. A suitable open-source interpreter that is compatible with Asterisk is Voiceglue[10]. It is based on the open-source OpenVXI interpreter. Implementing Voiceglue proved to be very difficult, as the Voiceglue project has been abandoned and not maintained for a long time. Because of this, Voiceglue cannot be installed on recent versions of Linux, and is only supported by old versions of Asterisk. There is no solution planned for the foreseeable future. This makes Voiceglue a suboptimal option to base a new project on, which has to be supported for a long time. Reviving and maintaining Voiceglue is out of the scope of this research.

---

[8] http://www.asterisk.org/
[9] https://www.w3.org/TR/voicexml20/
[10] https://github.com/voiceglue/voiceglue/wiki

VoiceXML IVR[11] (VXI) is a commercial VoiceXML gateway made by I6NET[12]. Although this software is not open-source, there is a free version available for Raspbian, the Linux distribution of the Raspberry Pi. The free version is limited to one simultaneous voice connection[13]. For the use case of the KasaDaka this is sufficient. VXI is still supported, however the package for Raspbian is one version behind the current release. It supports a recent version of Asterisk (but not the latest).

Development Framework: Flask (Python)

In order to support a wide range of complex use-cases, it is necessary to be able to dynamically generate VoiceXML files, incorporating data from a data store. It is important that is is relatively simple to create the dialog structure and to receive and send data from/to the data store. As VoiceXML files are similar to HTML files, a web development framework is suitable for this purpose. Flask[14] is a micro framework based on the Python[15] programming language. It works by having a method for each document. This method renders a template of a VoiceXML document, inserting the dynamic data into the VoiceXML document. This separation of the structure of the VoiceXML and the processing of data, ensures that modifying the code to fit a new use-case is relatively simple, while not limiting the complexity of the use-case. It also allows expansion of the project to also include regular web pages, for instance to administrate the system.

Data store: ClioPatria (SWI-Prolog RDF toolkit)

To store our data, almost any kind of storage can be used. The 'usual' choice is a SQL relational database, such as MySQL. For the KasaDaka, a triple store was chosen. A triple store has advantages over relational databases in the case of the KasaDaka. Triple stores have a flexible schema, which allows simple adaptation of properties of objects, and thus less effort to adopt a new use-case. Writing queries is simpler due to the nature of a triple store. Triple store queries are typically written in SPARQL[16] and sent over HTTP, which allows easy integration in our development framework. A major point for triple stores over relational databases, is the possibility of knowledge sharing. In a triple store, data is linked to other data, allowing for complex relationships. Combining data can result in new information. The Semantic Web[17] (Berners-Lee, Hendler, & Lassila, 2001) houses an enormous amount of knowledge that can be made accessible (however offline) through the KasaDaka (Nieland, 2013). Using a triple store allows for more possibilities in the sharing of information for the user. Semantic Data is lightweight, allowing for possible transfer of data through SMS, as described in the DigiVet use-case (Lô, 2014).

There are many triple stores available. ClioPatria (Wielemaker, Beek, Hildebrand, & van Ossenbruggen, 2015) is an open-source Semantic Web server. It is written on top of SWI-Prolog (Wielemaker, Schrijvers, Triska, & Lager, 2012), an open-source implementation of the Prolog programming language. It provides a web-interface for managing data and configuration. It provides a HTTP handler for SPARQL queries.

HTTP server: apache2

---

[11] http://www.i6net.com/technology/voicexml-ivr/
[12] http://www.i6net.com/
[13] https://wiki.i6net.org/doku.php/vxi_installation_guide:license:start
[14] http://flask.pocoo.org/
[15] https://www.python.org/
[16] https://www.w3.org/TR/rdf-sparql-query/
[17] https://www.w3.org/standards/semanticweb/

For hosting the dynamic VoiceXML files and the static audio files, a web server is required. Apache[18] is one of the most used web servers worldwide, and has a module for hosting Python based web pages.

3.2 Implementation

The installation of the modules described in 3.1 consists of several steps. First, the KasaDaka code and configuration files are retrieved from GitHub[19]. Second, Asterisk and VXI have to be downloaded and installed. Asterisk will be configured by using the configuration files provided in the KasaDaka git repository. These set-up Asterisk to answer calls, and presents the KasaDaka VoiceXML file generated by the Python code. The KasaDaka Python code files are then installed in the Apache web-server. This enables (local) web-access to the VoiceXML files, which will be generated (using data from the triple store) when requested by Asterisk (VXI). Finally, the ClioPatria triple store is installed and configured, allowing the use of dynamic data retrieved from the data store.

The system is set up, such that it is functional by just powering up the Raspberry Pi. After initial set-up, no further work is needed. By default, the Radio Marché use-case is implemented (section 4.2). When another usage scenario is desired, the Python code, data-store and audio files have to be adapted (section 4.3).

3.3 Call handling

When a call is placed to the system, all components work together to deliver a smooth experience to the user. Invisible and unnoticeable to the user, all the components are delivering the service. An example of the experience to the end-user is on YouTube[20].

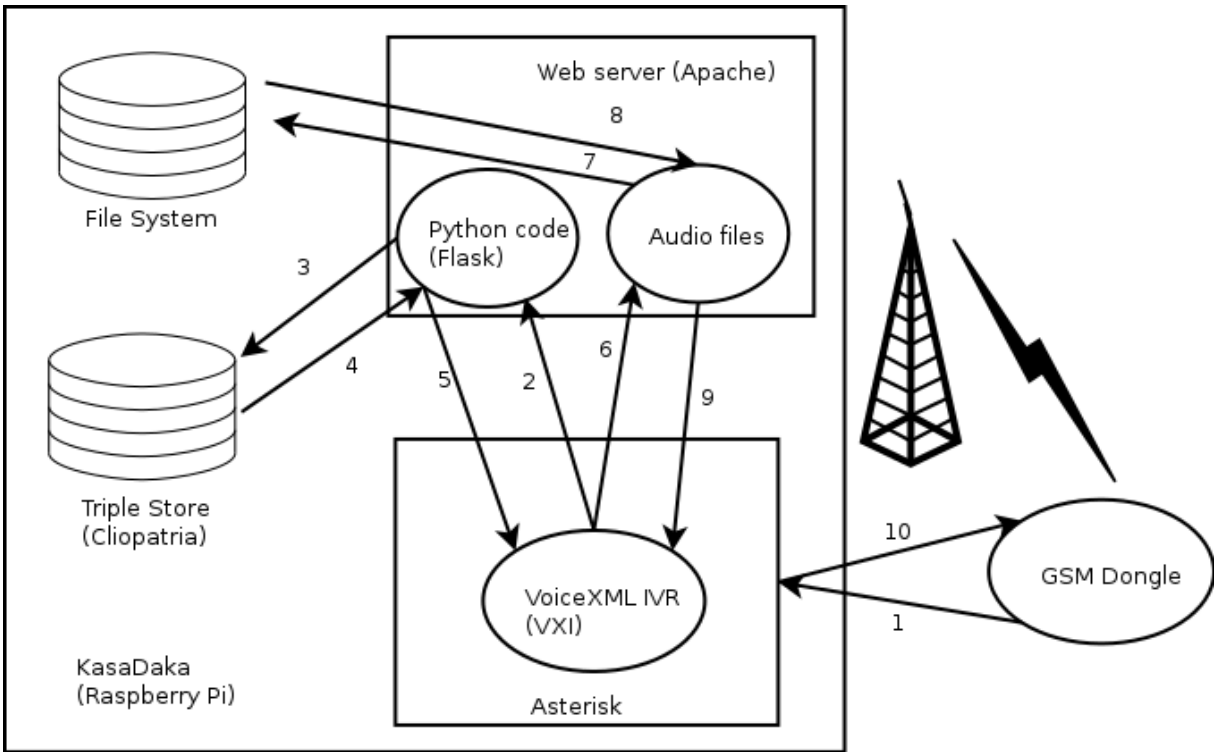

Figure 2: The system architecture

[18] https://httpd.apache.org/
[19] https://github.com/abaart/KasaDaka
[20] https://youtu.be/Bp-BFDQLhy0

When a call is placed, the system executes the following steps (see figure 2):

Text in *italics only takes place when setting up the call.*

1. *Asterisk receives the call from the GSM dongle, answers the call, and connects it to VXI.*
   Asterisk receives the user's input and forwards it to VXI.
2. *VXI requests the configured VoiceXML document from Apache.*
   VXI requests the configured VoiceXML document from Apache. Together with the request, it sends the user input.
3. Apache runs the Python program (based on Flask), in which data from the triple store has to be read or written. Python sends the SPARQL query to ClioPatria.
4. ClioPatria runs the query on the data present, and sends the result of the query back to the Python program.
5. Python renders the VoiceXML template. The dynamic data is now inserted in the VoiceXML document, and it is sent back to VXI.
6. VXI starts interpreting the VoiceXML document. In the document there are references to audio files. It sends requests to Apache for the referenced files.
7. Apache sends a request for the file to the file system.
8. The file is read from the file system.
9. Apache responds with the requested audio files.
10. VXI puts all the audio files in the correct order and plays them back sequentially, sending the audio to the GSM dongle.

This cycle repeats until the call is terminated.

4 Validation and testing

4.1 System set-up

Setting up a KasaDaka can be as simple as copying an image to the SD card. As all Raspberry Pi's have the same hardware, a preconfigured image will work on all Raspberry Pi 2's. This enables many KasaDaka's to be quickly rolled out.

Manually setting up the system involves configuring Linux, installing and compiling the necessary components (see system architecture) and their dependencies. The configuration files and code can be copied, after which the system is functional. A complete guide is posted on the GitHub repository[21]. Using this guide, it is possible to set up a system from scratch in a couple of hours. This makes setting up a KasaDaka a relatively simple task, as stated as a requirement in section 2.

4.2 Use Case: marketplace (Radio Marché)

The included code in the repository, is derived from the Radio Marché (de Boer et al., 2012; Dittoh, van Aart, & de Boer, 2013) project. It uses the Radio Marché dataset[22], to offer a basic marketplace service. It is possible to look up offerings of products (placed by others) and to place new offers. This demonstrates running basic queries on the triple store. Posting a new offering consists of giving input on all the required properties of an offering. These input options are also retrieved from the triple store. After all the necessary input is given, the offering is inserted into the triple store and will become findable by others.

Radio Marché use case

The Radio Marché use case is a service that provides a marketplace for offering products. This enables the users (rural farmers) to place an offer of their produce or other products, such as seeds or other food products. In the original use case, all offers were combined in to a communiqué, which was broadcast on the local radio. For our use case, we use the telephone connection for both offering and looking up offers.

When the user calls the service, first a choice of languages is offered. When the language has been chosen, the user is offered the two functions of the service, being looking up current offerings of products, and posting a new offering.
When the first function is chosen, the user has to choose which product he/she would like to look up the current offerings of. The system then plays back all the current offerings of this product that are available.
If the user chooses the second function, the user is requested to input the necessary information in order to place the product offer in the data store. This information includes: the identity of the user, the kind of product offered, the quantity of the product offered, the location, the price and desired currency.

[21] https://github.com/abaart/KasaDaka
[22] https://github.com/abaart/radiomarche/

4.3 Adapting to new use case

Crop Fertilizer lookup use case

This use case offers an information service. The user (rural farmers) can look up the best fertilizer to use for a crop. This is a use case that was the result of a local demonstration session of the KasaDaka project in Mali.

The set-up of this use case is fairly simple. The user is first prompted to choose a desired language to use in the application. The user is then offered only one function, which is to look up fertilizers for a certain crop. The user is given a choice of crops, and after the choice has been made, offers the fertilizers that are suitable for this crop.

Adapting the platform to host a voice service based on another (new) use case is not very difficult. In a test it was possible to implement a new use case (looking up what fertilizer to use for a certain crop) in around an hour. It proved to be possible to reuse a lot of the code in implementing another use case, speeding up the process. Most of the times (for simple use cases), changing the SPARQL query was sufficient to change to the desired functionality. More work however, is populating the starting dataset with the required information for basic functioning of the application. For every new object and property, audio has to be recorded in all languages. This can be a (relatively) long process of recording, editing, saving and referencing (in the triple store). However this depends on the complexity of the use case and the amount of used data in the triple store. In the case of the crop fertilizer lookup use case, the whole process came down to changing 2 methods in the Python code and recording 9 audio files. Also, a turtle (RDF) file had to be created and added to the triple store.

In conclusion, the framework can be changed to serve a new use case with relative ease. The Development Guide[23] makes this a simple process, and guides the administrator in adapting the framework to his/her needs. This fulfills the requirement as stated in section 2. With this addition, KasaDaka has become more versatile and can be deployed in many different use cases.

[23] https://github.com/abaart/KasaDaka/raw/master/Developmentguide.docx

5 Limitations and discussion

On the whole, the KasaDaka project has become a flexible rapid prototyping platform for voice services. There are almost an unlimited number of use cases that can be implemented. With sufficient funding, it is now possible to deploy KasaDaka's in many developing countries. Further research can be conducted on the consequences of introducing rapid exchange of information in the developing world.

There are some problems however. Implementing the system in many (niche) languages causes a lot of work. All possible objects, properties, numbers, et cetera, need to be recorded. This poses a lot of manual work, especially when the number of different possibilities increases. This also has to be done for every supported language. One area where there is definitely room for improvement, are numbers. Currently, each number has its own audio file. This means that there are possibly hundreds of thousands of numbers that need to be recorded. A possible solution would be to build a 'number audio file generator', which generates an audio file from the spoken numbers 0 until 9, 10s, 100s, 1000s, etc. This generator should use the grammar of a certain language to formulate larger numbers. In the Radio Marché (de Boer et al., 2012) project, such a generator was implemented. A good solution would be to include this generator to the KasaDaka project.

Another possible danger is abuse of the system. The usefulness of the system is dependent on the quality of the presented information. It could be in the interest of a certain party to degrade this information, for example by spamming or entering incorrect information. Currently, there is no authentication implemented. Reporting false information, and detecting spamming or other abuse should be considered before deploying KasaDaka on a large scale.

The issue of a lack of expertise does unfortunately also still remain. The farmers themselves can run the KasaDaka. But changing the functionality still requires programming knowledge. Because of the poor education in these areas, this is not likely to change soon. Changing the software also requires a screen and keyboard, reducing the affordability of the KasaDaka. True independent knowledge sharing is thus still not realistic.

One of the biggest questions however, is for how long these voice-based systems will remain relevant. As technology advances, it is only a question of time, until a large part of the developing population will possess smartphones with an internet connection. The illiteracy rate will probably not change as fast, still leaving a demand for voice based services, but their use may become limited when the voice based systems are not kept connected to their web based counterparts. This could of course be solved by providing the KasaDaka with a cellular internet connection, and connecting the data stores of both services.

5.1 Future work

As mentioned in the discussion, there are still challenges involving the language support and the possibility of abuse of the system. Other than these functional improvements, there are also many improvements possible in the area of user friendliness. The current system is largely based on program code and configuration files. Building a graphical user interface (possibly web based in Flask), would greatly improve the amount of effort needed to administrate the system. Building on this idea, a relatively simple improvement would be to build a backend to assist in recording the necessary audio files.

Further research could be conducted on expanding the capacity of the KasaDaka. The Raspberry Pi 2 is quite powerful and more than one simultaneous call should not be a problem. The licensing of VXI

is. It could be, that I6NET would be willing to support the KasaDaka project by providing a less restrictive license for non-profits.

Another interesting possibility is to have multiple KasaDaka's exchange information with each other. This would need to happen through SMS messages, as there is no internet connection. The cost of transferring data through SMS is high, so efficient encoding and compression is very important.

An advantage of using a triple store, is the possibilities of knowledge sharing. Research could be done on the possibility of browsing parts of the Semantic Web through voice based services. This would enable a form of self-education in the developing world. This research would also be of interest of making the Semantic Web accessible for the blind.

The social and economic impact of introducing these voice based information systems is an area where a lot of research can still be done. As KasaDaka's are low cost devices, they can be deployed on a large scale to study the impact and change caused in developing rural communities.

6 Conclusion

In rural developing areas, such as the sub-Saharan parts of Africa, information and communication technologies are not as widespread as in developed countries. The information services offered by these technologies, can however offer a great advantage to these rural communities. The adoption of mobile phones is relatively high in these areas. Offering voice based information systems to these communities can improve communications and shrink the *digital divide* in these communities.

The goal of this research was to create a flexible framework for hosting voice based services in rural developing areas. The framework has to meet the challenges posed by a developing world context. Thus allowing these services to be run and maintained by the communities themselves.

The framework proposed by this paper meets all the set requirements. It allows for simple adaptation to many use cases, and incorporates the implementation of data from a data store. This allows the framework to be extended to include many use cases, and it can be adapted to changing requirements with minimal effort.

This research thus enables further exploring of the possibilities and effects of offering voice based information services to rural and developing communities.

8 Literature

Aker, J. C., & Mbiti, I. M. (2010). Mobile Phones and Economic Development in Africa. *Center for Global Development Working Paper*, *211*(June 2010), 1–43. http://doi.org/10.1257/jep.24.3.207

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, *284*(5), 34–43. http://doi.org/10.1038/scientificamerican0501-34

Chhetri, D. (2013). Voice User Interface Design for m-Event Organizer. *Vrije Universiteit Amsterdam Master Thesis*, 1–28. Retrieved from http://www.few.vu.nl/~vbr240/teaching/Deepak_MScThesis_final.pdf

de Boer, V., De Leenheer, P., Bon, A., Gyan, N., van Aart, C., Guéret, C., … Akkermans, H. (2012). RadioMarche: Distributed Voice- and Web-interfaced Market Information Systems under Rural Conditions. *Advanced Information Systems Engineering*, *7328*, 518–532.

de Boer, V., Gyan, N. B., Bon, A., Tuyp, W., Van Aart, C., & Akkermans, H. (2015). A dialogue with linked data: Voice-based access to market data in the sahel. *Semantic Web*, *6*(1), 23–33. Retrieved from http://www.few.vu.nl/~vbr240/publications/swj2013_deboer_dialogue.pdf

Dittoh, F., van Aart, C., & de Boer, V. (2013). Voice-based marketing for agricultural products: a case study in rural Northern Ghana. In *Proceedings of the Sixth International Conference on Information and Communications Technologies and Development: Notes-Volume 2* (pp. 21–24). Retrieved from http://udsspace.uds.edu.gh/bitstream/123456789/152/1/Voice-Based Marketing for Agricultural Products .pdf

Fuchs, C., & Horak, E. (2008). Africa and the digital divide. *Telematics and Informatics*, *25*(2), 99–116. http://doi.org/10.1016/j.tele.2006.06.004

Gyan, N. B., de Boer, V., Bon, A., van Aart, C., Akkermans, H., Boyera, S., … Allen, M. (2013). Voice-based web access in rural Africa. *Proceedings of the 5th Annual ACM Web Science Conference on - WebSci '13*, 122–131. http://doi.org/10.1145/2464464.2464496

Heeks, R. (2008). ICT4D 2.0: The next phase of applying ICT for international development. *Computer*, *41*(6), 26–33.

Lô, A. G. (2014). The power of knowledge sharing : innovative ICTs for the rural poor in the Sahel. *Bachelor Thesis Vrije Universiteit Amsterdam*, (2523988).

Mantel, S. (2014). Small hardware solutions for voice services. *Vrije Universiteit Amsterdam Bachelor Thesis*, (June). Retrieved from http://www.few.vu.nl/~vbr240/teaching/stephan_mantel_scriptiefinal.pdf

Nieland, R. (2013). Talking to Linked Data : Comparing voice interfaces for general-purpose data.

Vrije Universiteit Amsterdam Master Thesis. Retrieved from https://videbo.files.wordpress.com/2014/07/eindversie-paper-rianne-nieland-2057069.pdf

Toyama, K. (2010). Can technology end poverty. *Boston Review*, *36*(5), 12–29. Retrieved from http://cs.bennington.edu/courses/f2013/cs2108.01/05-Can Technology End Poverty_ _ Boston Review.pdf

Wielemaker, J., Beek, W., Hildebrand, M., & van Ossenbruggen, J. (2015). ClioPatria: A SWI-Prolog infrastructure for the Semantic Web. *Semantic Web*, (Preprint), 1–13.

Wielemaker, J., Schrijvers, T., Triska, M., & Lager, T. (2012). Swi-prolog. *Theory and Practice of Logic Programming*, *12*(1-2), 67–96. http://doi.org/10.1017/S1471068411000494

9 Appendix

A demonstration of the KasaDaka platform is on YouTube:

https://youtu.be/Bp-BFDQLhy0

All the necessary information and files to set up a KasaDaka are located on GitHub.

https://github.com/abaart/KasaDaka

There is a guide on setting up the KasaDaka system, as well as a guide to adapt the platform to other use cases.

The website of the KasaDaka project also provides useful information and links.

http://kasadaka.com