IST 507427

SAFIR

Work Package 3

DELIVRABLE D.3.1.1 : eGovernment Domain Definition

19/09/05

Deliverable D3.1.1

Document name : WP3_D3.1.1 c

Release (Status): Draft **/** Provisional / **Final**
Date: 19-09-2005

Author(s):	Raphael Hérin, Wilfried Boes, François léger
Organization(s):	Voice-Insight
Contributor(s) autor(s):	
Document Reference:	WP3_D3.1.1_v01.0_final_eGovernment Domain.doc

Rules for Deliverable naming (Document Reference)
Originators of documents that are using the template will name it using the following structure,

WPx_Dx.x.x_vxx.rr_free.doc

Where

WPx	stands for work package number x,
D	means Deliverable
x.x.x	is the deliverable number (as mentioned in the technical annex),
Vxx	stands for **version** during the step of validation. Two digits for iteration (00 to n)
rr	stands for **release** during the step of document creation. Two digits for iteration (01 to n). Start the first release with the number V00.01
free	free text (for example, short title or subject)

Document Summary

Keywords:

Abstract: This document is presenting a definition of the e-Government domain within the VQL universe.
The document introduces 3 sub domains and explains how those have been implemented during the last 18 months.

Document History

Document Location: This document is only valid on the day it was printed.
The source of the document will be found in the Control section of the xxx Stage File (filepath).

Revision History: Date of this revision:
Date of next revision:

Version	Revision date	Summary of Changes	Organization	Name
01	19/08/2005	Creation of document	VI	R. Hérin
02	19/09/2005	Revision	VI	W. Boes
03	07/10/2005	Revision	VI	Ch. Kemper
04	14/10/2005	Revision	VI	W. Boes
1.00	15/10/+2005	Final first revision	VI	Ch. Kemper

Approvals: This document requires the following approvals.
Signed approval forms are filed in the project files.

Organization	Name	Signature	Date of Issue	Version
VI	François Léger		17/10/2005	1.00
VI	Charles Kemper		17/10/2005	1.00

Distribution: This document has been distributed to:

Organization	Name	Date of Issue	Version

Confidentiality: Only mentioned persons / Inside the WPx / Inside the consortium / Public

Version	Confidentiality level.
0X	Only mentioned persons / Inside the WPx / Inside the consortium / Public
1.00	Inside the consortium / NOT PUBLIC

Table of Content

1 Executive Summary

Introduction / Positioning in the Project

This deliverable aims at presenting a definition of the SAFIR e-Government domain.
This is needed technical element to allow the deployment for VQL and SAFIR e-Government applications.
As such it is an essential complement to the SAFIR SDK.
The document has been kept intentionally synthetic to focus on the concept.

Core of the document / Executive Analysis

The e-Government domain, as specified in work package one, and developed in WP 3 is composed of the following sub-domains:

- GIS (Geographical Information Systems) / LBS (Location Based Services)
- Document On Demand
- Form Entry

This document first describes the VQL domain concept. Then it describes how this was applied and developed to each of the sub-domains of the eGovernment domain.

VQL SAFIR SDK with the added eGovernment domain can now be used to develop voice interfaces to eGovernment applications in those sub domains.

The document describes some sample realizations and implementations that will be used in the pilot operations.

Although the system is of an universal nature, it was prototyped during SAFIR with two largely used commercial de facto standard ESRI ArcGIS for GIS applications and Adobe for Document and Form entry. Those were selected purposely for the obvious potential dissemination/exploitation potential.

This document however describes in a synthetic way the principles that could be applied, together with the SDK, to any comparable situation. Some recent development in GIS standardization in Europe could lead to promising synergies that must be further documented in the coming months.

Executive Conclusions

The e-Government SAFIR / VQL domain is now developed, at least in a first release that is ready to allow together with the SAFIR SDK to support the pilots that will be executed in the next phases of the project, mainly in:

- GIS (City Map updates, Sport Infrastructure asset management and more generally asset management, agriculture –Bulgaria-, transport –Bulgaria-, Computer Aided voice Dispatching –Greece-, city guide..)
- Fire, civil protection and Police
- Citizen pilots with STB

All those requesting in one way or the other a combination of LBS, form entry and document on demand.
As such it should support the potential in exploitation targeted by WP7-WP14.

2 VQL Universe

VQL (Voice Query Language), is a unique, automatic, dynamic syntax-centric, natural language technology, that enables users to directly request and manage information from any database application with simple, everyday language. This approach is in sharp contrast to other dialog-centric systems that provide access only through response to a limited, pre-programmed query list.

From a more technical point of view VQL offers a new approach between full natural language and grammar based speech solutions.

VQL makes it possible for users to use common off the shelve speech technology (both speech recognition and speech synthesis) to interact with existing Information Systems.

The main features of VQL that make this possible are:

- A Natural Language syntax based voice command interface.
- A domain concept that contains generic elements common to a particular class of applications (typically linking application topological relations with generic linguistic forms).
- Automatic dictionary and grammar creation based on information from the domain model, from the information system (system commands, usually actions) and from the managed data set (objects to which actions are applied).
- The dictionary is created at runtime and can be dynamically adapted to the application context, allowing to create the voice interface automatically on the fly.
- Seamless vocal access to multiple applications through multiple active dictionaries and grammars. This feature makes VQL a Vocal Application Browser.
- Integration of speech technology through industry standard interfaces and possible proprietary interfaces
- Dictionary Memory swapping management system.
- A toolkit or SDK to integrate the Information System with VQL.
- VQL multilingual and cross lingual features (allowing query in one language in a database written using another language). This feature will be possible with the ontology server that is being developed within SAFIR in WP5.

2.1 Natural Syntax Centric VQL Solution Principle

2.1.1 Generalities: Dialogs and Syntaxes

User interfaces can be either syntax or dialog based. Graphical user interfaces are dialogs based because at each selected menu level, the system responds with next available menu choices. Dialog interfaces can be very awkward, especially when many successive choices have to be made.

2.1.2 Natural Language

Natural language is syntax based. People understand a sentence because the language syntax is wired in their heads. Sentences can carry a whole lot of information at a cost of being very complex to analyze.

2.1.3 Domain Based

Vocal interfaces using natural language are best suited to a syntactical approach. VQL uses a domain based natural language syntax approach.

The purpose of the VQL solution is to allow a user to query or command existing information systems through spoken language. Commands involve the activation of information system functions. Those functions can in most cases be described as actions on objects managed by the information system.

The richness of natural language vocabulary and syntax allow actions to be requested in a multitude of different ways. The task of understanding natural language is a considerable one. Even if there would be no technical limitations in the speech recognition engines, it still would be very elaborate to describe the syntax and vocabulary people uses effortless while composing daily requests for action.
Complex natural language analysis would be needed to extract the meaning from the stream of words understood by the speech recognition engine. Only then the intention of the speaker could be translated into actions in the information system.

VQL basic uses the opposite approach. If one knows the set of objects and related functions that an information system makes available to a user (in a given context), commands that apply to these objects can be phrased in a “generic” way. Only those VQL phrases applicable to the particular user, the

particular context, the set of objects available to the user in the information system need to be made available at that particular time.
This principle reduces both the complexity of the grammar of allowable commands and the size of the dictionary of words that need to be understood. It also reduces the natural language understanding task to a simple grammar-parsing task. The cost of this simplification is a reduction in the freedom of the speaker. However, the speaker also gains in system responsiveness and better recognition under all circumstances. Furthermore, the VQL Natural Language or 4th VQL layer (not described here) enrich this basic concept, allowing to take care of synonyms and variations...

VQL phrases are thus built according to an application domain specific set of patterns or templates. For the SAFIR project, there will be several domains. The present document is describing the eGovernment domain. As stated in the introduction, there are three sub-domains that are included in this one:

- A VQL for GIS implementation
- A VQL for Document On Demand implementation
- A VQL for Form Entry implementation

2.2 The VQL Environment

The environment is composed of the following elements (See Figure 1):

- An application (a1) that provides the object class definitions and the actions to be applied to these object classes.
- A speech processing system (a2).
- A system analyst and linguist that provide the plausible phrases prototypes that will be used to exercise the applications actions on the object classes. For instance, the definition of the phrase ‘show <object name>’ in order to display an object.
- A user that will use the system (a4).
- The VQL system enabling the user to speak the phrases that the application must process (a5).

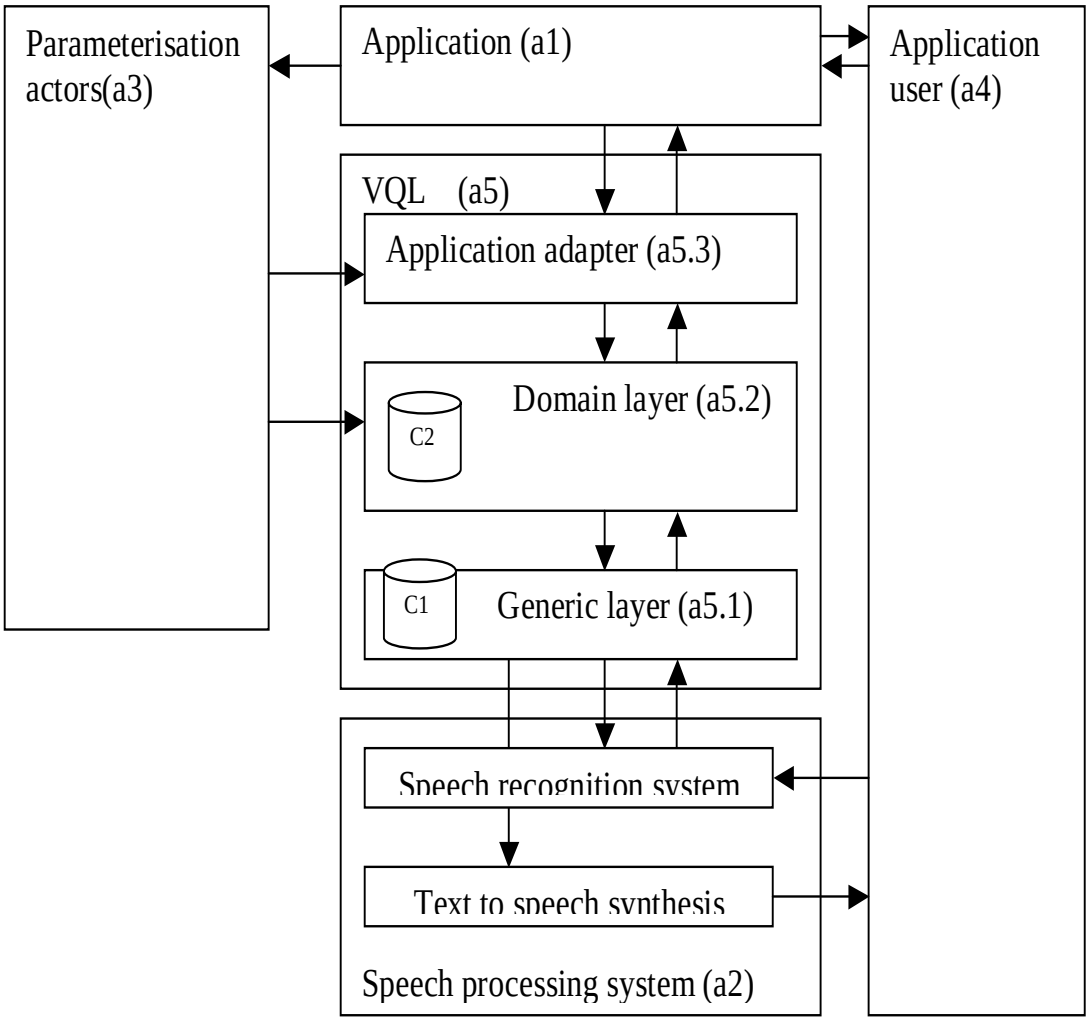

Fig 1: Voice query language system environment (principle schema)

2.2.1 The speech processing system

The speech processing system (a2) consists of two parts:

- The speech recognition system, a software transforming phrases spoken by the user into a computer representation of text
- The 'text to speech' synthesis software that can read aloud a computer representation of text.

2.2.2 The VQL system

The core VQL system (a5) consists of three parts / layers:

- A low level interfacing part (a5.1) that converts a language grammar ('identified words lists' and 'words list combination rules') into a format that can be used by the speech recognition software and vice-versa, converts the result of a speech recognition into a phrase defined by a grammar. This low level interface is also called VIVA and can be use separately
- A high level interfacing module (a5.2) that allows a software component to provide object type definitions, their semantics in a particular application domain, attributes definitions

for these object types, object instance's names and existing known values for attributes corresponding to existing or imaginary entities in an application (a1). The module combines these definitions with related action phrase prototypes to produce the grammar needed by the point a5.1 above. The module can take a phrase produced by a5.1 after recognition happened and generates an action on specified objects identified by specific attribute values and transfers it to the application adapter (a5.3).

- A interfacing software (a5.3) that is created for a specific application that constructs the information needed by the software described under the point a5.2 and is able to invoke the functions of the application to perform the action requested by the user (a4). This software layer is called the application adaptor.

The core VQL layer is complemented by a 4th and 5th layer, but this is out of the scope of this deliverable, even though it would complement this deliverable results.

2.2.3 The sequence of events

2.2.3.1 Preliminary tasks to be implemented

The preliminary tasks to be done in order to create the vocal interface are described here after.

- A domain is identified (market research, commercial agreement for a given application...). A domain regroups the common semantics of a class of applications. This defines the target of the parameterisation. (a1 selection)
- The analyst and linguist define the object classes and related action phrases (templates) required by the domain.
- The information is encoded into a formal definition that is usable by the software referenced as a5.2 above
- A software to be used in a5.3 is implemented for a specific application a1 that belongs to the domain identified in p1

2.2.3.2 Application domain template definition

In order to create the domain layer of the VQL system, several things are to be defined.

2.2.3.2.1 The domain Identification

The domains define configuration groups that can be easily added or removed from the system.

2.2.3.2.2 The object classes belonging to this domain

Each domain is composed of object classes. Each class represent a set of objects with a common characteristic.
These object classes do not necessary identify completely a real object: a real object will most probably be part of several object classes.

2.2.3.2.3 The various actions that can be applied to the objects classes

Associated with each object class, one can define, for a given domain, a set of action that can be applied to these object classes: an object belonging to the displayable object class (in other word, a displayable object) can be displayed or hidden, similarly, a selectable object can be selected or deselected, a navigable object can be moved to a selectable object, two geographical spatial objects can intersect…

2.2.3.2.4 The attribute classes that can be attached to the objects

As we have defined object classes to identify specific characteristics of objects, we define attribute classes to categorize attributes with a common meaning toward the objects such as object's naming attribute, instances' naming attribute, attribute's naming attribute …

2.2.3.2.5 The attribute types that relate the attribute classes to the objects classes

In order to more precisely relate an action to certain instances of an object class, we need to identify the objects of specific object classes. For this, one needs to select an instance-naming scheme for objects of each particular class.

2.2.3.2.6 The various phrases that can be used to request the execution of the actions

Associated with each class action, one can define a set of phrases templates that can be used to request the class action to be done.

2.2.3.2.7 The sequences of word groups that compose a vocal command

Each command is composed of a succession of term elements, either static or dynamic, that will be assembled in sequence to generate pronounceable phrases.
Each of these elements can be flagged as optional; this means they can be forgotten from the speech without impairing the meaning of the phrase, such as politeness expressions. Each of these elements can be flagged as repetitive as for composing numerical values or identification codes. They can also be associated with a specific decoder as for converting automatically dates, times, numeric values …

Finally, each of these static elements can refer other static or dynamic elements.

A typical example is the following sequence:

[-1] [2] [+3] [-4]

Where: [-1] mark an optional term list composed of the terms ‘please’ and ‘can you’
[2] marks an obligatory term list composed of ‘pan to the’ or ‘move to the’
[+3] marks a repetitive term list composed of the terms ‘north’, ‘east’, ’west’, south’
[-4] mark an optional term list composed of the terms ‘please’

The following are valid expressions according to that definition:

“pan to the north east”
“can you move to the south south east please”

2.2.3.2.8 The elements marking the presence of dynamically defined text group in the commands

In the example phrase above, all the elements where defined statically. If we take another example such as:

[1] [2] [3] and

[1] [4] [3] [5] [2]

Where : [1] is either ‘show’ or ‘display’
[2] is the related object instance names
[3] is the spatial object names
[4] is ‘the’
[5] is ‘of’

Then, if at run time we have two spatial objects called city and state and two instance names called New-York and Washington for the city and New-York and Idaho for the state, then the combination generation will generate the following templates:

[1] [2a] [3a]
[1] [2b] [3b]
[1] [4] [3a] [5] [2a]
[1] [4] [3b] [5] [3b]

Where: [2a] will be either New-York or Washington
[3a] will be city
[2b] will be either New-York or Idaho
[3b] will be state

[3a] will be city
[2b] will be either New-York or Idaho
[3b] will be state

Enabling commands like:
Show New-York city
Or
Display the State of Idaho

[2] and [3] are placeholder (dynamic) term lists that are used to create extraneous phrase templates at run time. These term lists refers to attributes types, thus to object classes and to attribute classes that set the way the term lists will be generated.

2.2.3.2.9 The elements marking the presence of statically defined text group in the commands

These are straightforward list of terms, which elements can be predetermined during the domain definition and thus simply need to be defined.

2.2.3.2.10 The text composing the statically defined text groups

These represent the different words group that can be used interchangeable within a given phrase.

2.2.3.2.11 The various elements that are required to identify precisely an action and/or the objects subject to an action

Associated with an action, a means should be provided to specify the function and the arguments to be called in the application adapter (a5.3) so that the application can perform the action. The arguments to be returned will be a modifier to the primary action (such as a pan direction in a windows: up, left, right, down), or an object's identification, an object's attribute or object's attribute value…

2.2.3.2.12 The various type of value that can be handled by the system

In addition to the above definition, present for usage in the domain interface, but define statically are the various definitions required to generate the interfacing of the attribute type values either by the natural language or the spelling of the values. Associated with these definitions are the related processes required to rebuild a proper definition (a number, a date...).

3 The eGovernment Domain

Now that the definition of a domain in the VQL environment has been made, we will concentrate, in this part of the document, on the sub-domains that form the eGovernment domain.
The result of WP1 presented at M6 and finalized at M12 clearly indicates that in the scope of SAFIR an e-Government domain should handle :

- Geographical Data
- Document / Form On Demand
- Form Entry

3.1 VQL for Geographical information System (GIS) sub-domain

The interest in geographic information (GI) in general and GIS in particular has increased tremendously in the last two decades. As societies are becoming increasingly aware and concerned about governing their activities, there is an increasing appreciation for the role geographic information can play in this process. Geographic Information Systems (GIS) technology can provide great value for empowering public organizations.

The aim of the SAFIR project is to give special user groups an equal opportunity to access and modify at any time interactively existing up to date e-government information thru easy devices as TV, wired and wireless devices (GSM/PDA), from anywhere by usage of own voice in his daily language. For that matter, the SAFIR project is also dedicated to public organizations of all member states of the EU. Its aim is to provide them a variety of tools that could ameliorate the services provided to the citizens of the EU.

Geographic Information Systems (GIS) technology can provide great value for empowering public organizations. It enables its users to intelligently create, manage and manipulate their geographic-based data. In the SAFIR project, several pilots are going to be implemented with GIS application. This is for example the case of the Bulgarian, the Belgian and the Greek pilot. (See work package 6 for the pilot preparation deliverables associated with those pilots).

3.1.1 The Belgian Pilot

In order to describe the VQL for GIS sub-domain, a description of the Belgian pilot will be done.

3.1.1.1 The UrbIS Database

"**Brussels UrbIS**" (**Brussels Urb**an **I**nformation **S**ystem) is a group of geographic and alphanumeric data of the Brussels-Capital Region. One of them is "Urbis-Topo" which is a vector database constructed by photogrammetric plotting with aerial photographs. Through this process, several types of objects are digitized and added to the database at a scale of 1:500. Fieldwork is then undertaken to compare the work carried out in offices to reality and to complete the database with objects that could not be added with photographs analysis.

The features contained in the database represent limits of different types like front or rear façade, common walls, hedges, parcel limits, road sides, parking plots limits, etc.
Street furniture is also represented in the database (sewer manholes, benches, dustbins, mailboxes,…). Some of these features have to be created in the database because they were not itemized up to there and others need to be repositioned due to incorrect photo-interpretation.

The UrbIS database is a regional standard and is used by more then 120 authorities and private companies in Belgium.

3.1.1.2 Voice as Effective Tool for the revision of the UrbIS Database in the Field…

Cartography is an extremely useful discipline, essential in today's world… However, a map that is not up to date is almost worth nothing! It is consequently essential to bring and keep all the contained information of the Urbis database up to date regularly.

Part of this work can certainly be carried out in offices by observing the aerial photographs but the final validation can very often be done only by sending people in the field. It's a meticulous work but thanks to the vocal contribution, this task can not only be more pleasant and user-friendly but faster and more accurate since data collection and information requests really become more efficient and easier to use. In addition to that, the ability to work directly in the field on the different objects by means of the voice could initiate the digital flow in the field and save much time for the database's update processes. Indeed, considerable work in offices after fieldwork to incorporate the changes could be avoided as it could be done directly in the field. Very often also when field work is replaced by office work, new verifications in the field are requested.

The field work will consist of scanning any modification in the field or any new objects to insert in the database, compared to the map displayed on his tablet PC equipped with a GIS application.

An example of inside dialog and reflexion could be the following one:

- There is a sewer manhole in the middle of the road; it is not present in the map…
- I have to create it based on it's position from other existing features present in the database
- There should be a parcel limit for this parcel here…I will have to trace it
- There should be a wall here instead of a simple parcel limit…
- I have to transfer it to another information layer and thus change it's properties
- On the left side of the road, there's a sports center…
- I should label this building as "sports center"

And so on…

3.1.2 VQL for GIS Description

3.1.2.1 The object classes

The typical object classes specific to the GIS domain are surface class, linear class and punctual class. In a GIS, map data contains the location and shape of geographic features. Maps use three basic shapes to present real-world features: points (symbolizing cities, dustbin, manholes,…), lines (symbolizing streets limits, frontage limits, rivers,…) and areas (called polygons and symbolizing countries, regions). Here, only object classes that are very specific to a GIS application are described but we could think of anything else, even virtual objects. For example the graphical object classes (representing Aerial Photography, layers of information and maps,…). The object classes specific to the GIS sub-domain are created in the VQL for GIS definition to represent those features and to have the ability to access those features vocally.

These object classes do not necessary identify completely a real object: a real object will most probably be part of several object classes.
In the GIS sub-domain, a typical object such as a city street will belong to the following classes: displayable object class, selectable object class, spatial object class and linear object class... The displayable object classes that can be added to a map, the selectable object classes that can be selected, the spatial object classes that can be projected on a map using various projection rules, and the linear object classes that can be moved along for a certain distance to provide a specific location.

3.1.2.2 Actions associated to the object classes

Associated with these object classes, one can define a list of actions. An object belonging to the displayable object class (in other word, a displayable object) can be displayed or hidden, similarly, a selectable object can be selected or deselected, a navigable object can be moved to a selectable object, two geographical spatial objects can be intersected…

3.1.2.3 The attribute classes that can be attached to the objects

Attribute (tabular) data is the descriptive data that GIS links to map features. Attribute data is collected and compiled for specific areas like states, census tracts, cities, and so on and often comes packaged with map data.

As we have defined object classes to identify specific characteristics of objects, we define attribute classes to categorize attributes with a common meaning toward the objects such as object's naming attribute, instances' naming attribute, attribute's naming attribute.

For instance, when referencing the linear features that are of type 4 (representing frontages in Urbis), type 4 is compared to the line type attribute of the line features which name is "linetype". Linetype is the attribute's type attribute.

3.1.2.4 The attribute types that relate the attribute classes to the objects classes

In order to more precisely relate an action to certain instances of an object class, we need to identify the objects of specific object classes. For this, one needs to select an instance naming scheme for objects of each particular class. Typical examples of such a definition are the spatial linear objects instances name such as street names.

3.1.2.5 The various phrases that can be used to request the execution of the actions

In order for the system to understand a particular phrase, the syntax has to be defined. This is how when the user says "display xyz", the class action is requested. While the specific *'xyz'* cannot necessarily be defined during the creation of these templates phrase, the type of the attribute (see 3.1.2.4 above) used to identify the object on which the action will be performed can be referenced. This will be populated by the existing values to be discovered during the software activation. Of course, these phrase templates need to be defined for each language supported. Furthermore, words like '*display'* can have synonyms that could easily be exchanged with the original terms. This means that multiple variants of a same phrase can exist without many changes in its grammatical structure.

3.1.3 Deliverables

- Object Classes 'Linear', 'Spatial', 'Punctual'
- The associated command templates for object classes 'Linear', 'Spatial', 'Punctual'
- Definition of a GisCallbackInterface to implement the actions associated to the command templates for object classes 'Linear', 'Spatial', 'Punctual'
- Belgian Gis Pilot

3.2 VQL for document on demand

Of all the interactions between a private person and the government, the exchange of information in the form of documents is the most frequent. Such a transaction happens in two phases:

- First, the user searches for the document (form) through a service locator.
- Second, the user fills out the document and submits it to the government.

Also, the request for receiving a document from the government happens generally in the same way, a request form is obtained, filled in and submitted to obtain the requested document.

VQL for document on demand provides support for constructing a vocal interface for this process.

3.2.1 Service

The act of looking for a document can be abstracted to the act of looking for a service i.e. the service being the process of requesting the document or filling and submitting the document.

A service, after it has been located with the help of a service locator, can be started and stopped. Also one can switch between active services.

This can be expressed through vocal commands as : use form, start form, stop or exit form, switch to form.

A service may have a vocal interface or not. A vocal interface of a service can be enabled or disabled.

A service may be graphical or not i.e. the service may support a GUI or be only vocal. A graphical service can be shown, hidden or printed on request.

A service locator is a service to locate other services. A service locator name can be used in a help command e.g. 'Help me with defence documents' to start the service locator for 'defence documents'

The classification of objects as service, vocal, graphical or service locator triggers the automatic generation of command templates associated to each classification.

3.2.2 Service Manager

A Service Manager is an application that manages the access to services. A service manager has a vocal interface that is configured by specifying the various types of services that it manages. Through this configuration, the service related command templates are added to the vocal interface of the service manager.

Of course, when a command has been recognized, the service manager has to implement the execution of this command.

The VQL document on demand domain has predefined a number of callback interfaces that will be called when a related vocal command was used.

e.g. the user says 'List all tax documents'.

The fact that this command was understood implies that:
- A object class was configured with the associated text 'tax document'

The ObjectCallbackInterface.QueryAllObj function will be called. It is up to the ServiceManager implementation of this function to show a list of tax documents on screen.

e.g the user says 'Select car tax form'.

The fact that this command was understood implies that:
- A object class was configured e.g. 'tax document' with a identifying attribute e.g. 'document name' that has among its known values the value 'car tax form'. Therefore the command is understood as a request to select an instance of the class 'tax document' with 'car tax form' as value for the 'document name' attribute.
- During the initialisation of the service manager, VQL called the ObjectCallbackInterface.InitAllAttrVal function in order to get the list of all known 'document name' values. It is up to the Service Manager to supply this information to VQL. This implies that the Service Manager has knowledge about the existance of a document catalogue and about ways to access that catalogue.

The ObjectCallbackInterface.QueryObjInst will be called by VQL to allow the ServiceManager to lookup information about the 'car tax form' and to make this service the currently selected service.

e.g. the user says 'Open the form'

This command implies that there is a currently selected service. VQL will call the ServiceCallbackInterface.StartService function. The service manager will do what is needed to open the form (start the service). If this service is a voice-enabled service, the service manager will call the AdapterInterface AdapterEnable function on the service to enable the vocal interface of the service.

3.2.3 Service locator

The Service Manager can implement the standard VQL query commands. These commands provide syntax for simple composite queries e.g. query objects (in this case services) based on their attribute values.

Example:

With issuing department equal to department of economic affaires.
List all newsletters.

For more advanced searches, a real Service locator can be build. This can be a question-answer (dialogue) based interface to narrow down the choice of applicable forms. Also an ontology-based service locator is probably a more user-friendly way of locating a needed document. All the standard features of VQL allows such an interface..

A service locator can be started with the 'help' command templates.

3.2.4 Deliverables

- Object Classes 'Service','Service Locator','Vocal'
- The associated command templates for object classes 'Service','Service Locator','Vocal'
- Definition of a ServiceCallbackInterface to implement the actions associated to the command templates for object classes 'Service','Service Locator','Vocal'
- Definition of a VocalAdapterInterface to manage the cooperation of multiple vocal interfaces.
- A sample implementation of a Service Manager, Service and Service locator

3.3 VQL for form entry

As argued above, a document request or submittal can be considered (process view) as a service.
The services can potentially have a visual and/or a vocal interface. A service where both visual and vocal interfaces can be very productive is the filling of forms.

Today, reader programs exist as an assistive technology for visually impaired users. These programs use text-to-speech technology to read the content of a document or form to the user.

VQL for form entry is a pilot program to extend the functionality of a reader to the filling of fields in a form. The pilot is currently limited to one de-facto standard i.e. Adobe PDF forms.

3.3.1 VQL Dialogues

The pilot is based on standard technology present in VQL. VQL allows the creation of multi-modal interfaces through:

- Definition of semantically tagged speech grammars. When a user speaks, the captured information is passed as semantic tag-value pairs to a callback function.
- Definition of dialogues
  - Subdivided into individual interactions called dialogue items
  - Background commands are represented by a speech grammar that is always active. This allows the user to say un-asked-for things that can change the flow of the dialogue at the users initiative.
  - The control and data flow between dialogue items is controlled through callback functions
  - Each item defines:
    - 1-3 prompt messages that are automatically repeated at pre-defined intervals
    - a semantically tagged speech grammar or sub-dialogue (what the user can say)

- The vocal interface can in addition to speech events also process so called 'external' events. These events can originate in GUI actions, scanning events, keyboard events. The external events are treated by callback functions, just as the speech events. This permits multimodal interfaces where the

same action can be accomplished either through speech, keyboard, mouse or other action. Also external events can change the normal flow of the dialogue.

3.3.2 Forms entry

The process of filling a form is very similar to the execution of a dialogue. Each field on a form needs a corresponding prompt(s) and a speech grammar covering the possible inputs. The default fill order of the form corresponds to the default control sequence of the dialogue. This default sequence can be changed through events like :

- user clicking with the mouse in a new field
- user switching vocally the focus to another field by saying e.g. 'Quantity (field) is 20'

In order to be able to generate automatically a form filling dialogue for a given form, extra information has to be present in the form itself (tags or annotations on the form) or as a collateral to the form (metafile per form or a metadatabase for multiple forms). That information must contain:

- speakable text describing the form
- for each field
  - speakable text describing the field
  - prompt and help messages for the field
  - speech grammar for allowed inputs
  - for buttons (e.g. submit button), only the speakable text is needed
- default sequencing of the filling process
- the above information may be present in multiple languages

3.3.3 Activation

When a form is selected, the service manager enables the vocal interface of the form if present.
The form filling interface will dynamically create the form filling dialogue based on the meta-information.

3.3.4 Favorites

Very often the same information has to be given when filling a form. Often, this information is difficult or impossible to recognize correctly. Examples are telephone numbers, addresses, names, membership id's.
It would be very useful for the service manager to allow the definition of vocal favorites or aliases for this personal information. This way you can say 'Name is my spouse's maiden name' to fill a name field. 'My spouse's maiden name' is an alias for the actual name.

3.3.5 Deliverables

- Pilot implementation of vocal form filling based on Adobe PDF forms.

4 Conclusion

Some key concepts needed to develop voice interfaces to applications in the eGovernment domain were identified and integrated into the VQL universe. Together with the SAFIR SDK this should enable faster pilot implementations of voice driven eGovernment applications as the pilots should prove it in the next phase.

The concepts explained were implemented as proof of the concept with ESRI ARCGIS for the GIS part and with ADOBE PDF. This choice dictated by the market position of both systems should not be misinterpreted: the system proposed through the SAFIR initiative is of a generic nature. Standardisation initiatives in the GIS and e-Government arena could be combined together with the principles explained in this document, this will most probably be an axis of research, dissemination and exploitation for the SAFIR consortium in the coming months as the pilots will be unfolding.