Consumer Voice Interface Patterns

Dirk Schnelle and Fernando Lyardet
Telecooperation Group
Darmstadt University of Technology
Hochschulstraße 10
D-64283 Darmstadt, Germany
{dirk|fernando}@tk.informatik.tu-darmstadt.de
phone: +49 (6151) 16-4267
fax: +49 (6151) 16-3052

Abstract

Voice User Interfaces (VUI) are a relatively new concept, although voice applications have been around for several years now. The invisible and transient character of sound led to believe that there was no user interface. This misconception rapidly disappeared with the technology advances that triggered the development of novel and rich voice applications for corporate use as well as in cars, phones and everyday appliances. The new and more complex functionality became a whole area known as Voice User Interfaces to address the different design forces and complexities behind a voice-based user interface.

In the last years we began a pattern language for designing voice applications and documenting the aforementioned design tensions and common practices. In this paper, we continue with this work focusing on more complex issues such as authentication and consumer appliances interfaces.

1 Introduction

Voice User Interfaces are particularly difficult to build due to their transient and invisible nature. Unlike visual interfaces, once the commands and actions have been communicated to the user, they "are not there anymore". Another particular aspect of VUI is that the interaction with the user interface is not only affected by the objective limitations of a voice channel, but human factors play a decisive role as well: auditive and orientation capabilities, attention, clarity, diction, speed, and ambient noise.

The design patterns presented in [14] showed that it is possible to build a language to talk about VUI designs. We extended this language in [13] with a focus on the different dialog strategies, auditory design, and usage

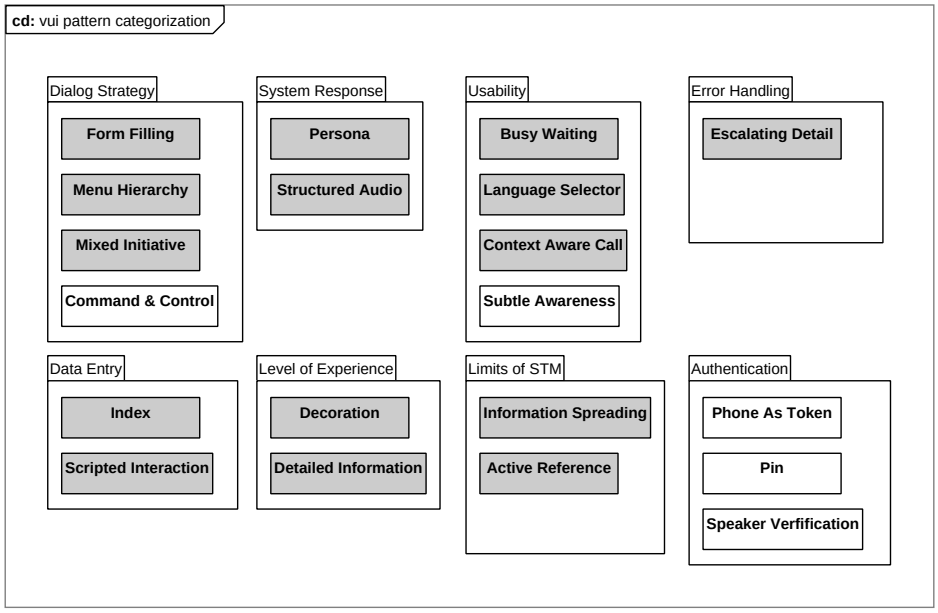


Figure 1: Categorization of Voice User Interface Design Patterns

scenarios and are now extending the language again with a focus on voice applications for corporate use.

The structure of the paper is as follows: we first provide an overview of the whole pattern language. Afterwards, we introduce the discovered patterns grouped by the VUI design aspect they address:

2 Integration Into the Existing Language

These new patterns expand the language in important areas of VUI design, closer to how the actual process of VUI design development actually takes place.

The patterns can be categorized according to their main purpose. An overview is given in figure 1, and connections between them are shown in figure 2. This pattern language is just a starting point and is not complete in all categories. There is a need to mine more patterns, for example to handle errors.

Figure 2 shows an overview of the language.

The goal is to document and share this knowledge with new designers (see figure 3) as well as providing a language to talk about *Voice User Interface* (VUI) designs. "Patterns and pattern languages offer an approach to design with much potential" [3]. The patterns described in this paper continue to build a pattern language to talk about VUI design. An example

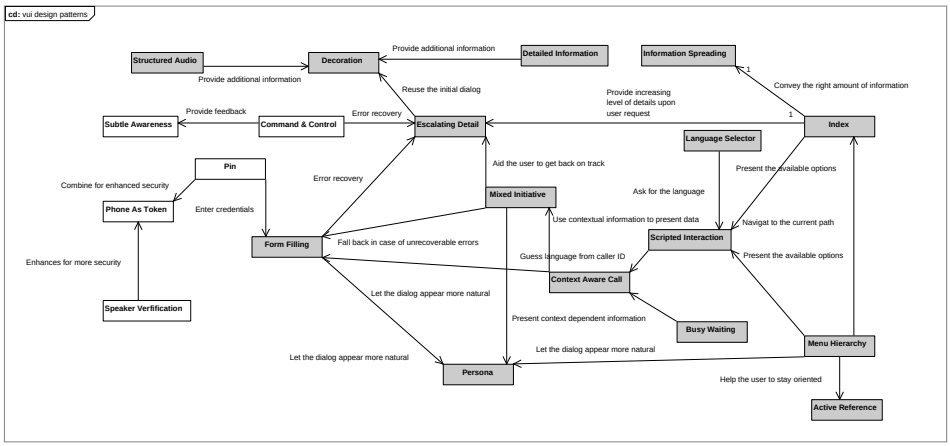


Figure 2: Relation of Voice User Interface Design Patterns

implementation of each pattern is given. The scenario of our examples is at a car inspection, where a worker processes a list of work items. We use VoiceXML for the example code. A specification of the tags can be found in [15].

3 The Difference with Audio

Development of audio based applications is different to development of graphical oriented applications. In [13] we grouped the challenges with audio into the two categories *technical challenges* and *audio inherent challenges*. It can be assumed that the technical problems, like speech recognition performance and speech synthesis quality, can be solved as technical progress is being made. The problems inherent to audio will be impossible to solve completely, but it is important to know them and to find workarounds, probably with the help of our pattern language.

3.1 Audio Inherent Challenges

Audio inherent challenges are *one-dimensionality*, *transience*, *invisibility*, and *asymmetry*.

One-dimensionality The eye is active whereas the ear is passive, i.e. the ear cannot browse a set of recordings in the same way as the eye can scan a screen of text and figures. It has to wait until the information is available, and once received, it is not there anymore. This is meant by one-dimensionality.

Transience Listening is controlled by the short term memory. Listening to long utterances has the effect that users forget most of the information that was given at the beginning. This means that speech is not an ideal medium for delivering large amounts of data. Transience has also the effect that users of VUIs often have the problem to stay oriented. They describe a phenomenon, which is called lost in space problem, which is also known in web based application.

Invisibility It is difficult to indicate to the user what actions she may perform and what words and phrases she must say to perform these actions. In contrast to graphical environments, where the means to enable user interaction are directly related to capturing the user input, the presentation of a voice user interface is completely independent to the evaluation of the entered data. Moreover, invisibility may also leave the user with the impression that she does not control the system. Note that there is a difference between *feeling to be* in control and actually *being* in control.

Asymmetry Asymmetry means, that people can speak faster than they type, but can listen much more slowly than they can read. This has a direct influence on the amount of audio data and the information being delivered. This property is extremely useful in the cases, where we have the possibility of using additional displays to supplement the basic audio interface. We can use the displays for delivering information, which is unsuitable for audio due to its length, and focus on using the audio device for interaction and delivering short pieces of information.

4 Additional Background Information

Some of the pattern descriptions feature a section about known uses. In general, this is a real telephony application which can be called. Some of the phone numbers may incur calling charges. It is also possible that they are not accessible from outside of Germany.

Figure 3 shows a simplified overview of the roles involved in VUI design. *Novices* are users that have not used the application before, and are neither familiar with the concepts of the application nor the grammar they can use. *Experienced users* have a systematic understanding of the application and its functionality. The *VUI designer* develops the application and the concepts on how to present it to the user.

The main problem with these roles is, that all actors have a direct relation to the core thing, the voice application, but all have different levels of understanding.

The VUI designer knows all details of the application. In order to design the application she also has to think like an experienced user and like a

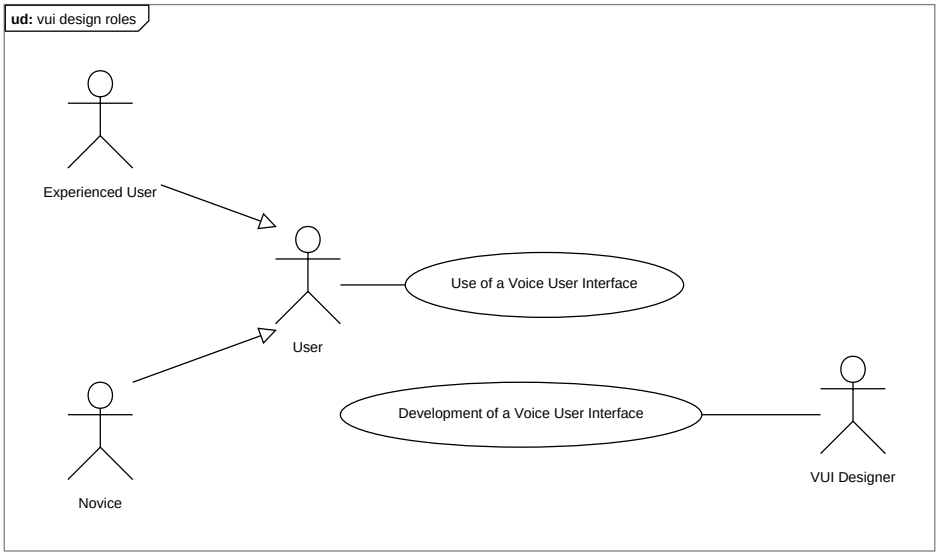


Figure 3: Simplified view on roles involved in VUI design

novice. Since experienced users also have a detailed understanding of the application, it is relatively easy for the designer to take the experienced user's view. The role of the novice is more severe to understand for the designer. She has to provide the user with means to learn the concepts and structure of the application to become an experienced user. Additionally the designer has to take care that both, experience user and novice find a comfortable way to interact with the application. Hence, we have a set of core forces for the development process that are shared among all patterns

- Dialogs have to to be efficient and short
- Dialogs have to be clear and structured
- Novices have to be guided. They need to know what kind of information to provide
- Experienced users know what to say and need a fast way to enter the data
- Designers knows all aspects of the application
- Designer have to know the role of novices and experienced users by heart

For some of the patterns we suggest to use a *wizard-of-oz* experiment [6]. This is a well known strategy taken from the field of human-computer interaction. Test persons interact with an application which they believe to

behave autonomous. Actually it is operated by an unseen human being, the *wizard*. Instead of real voice recognition, the user's input is manually entered into the system and processed as a text stream rather than an audio stream. These experiments should reveal information about use and effectiveness of the application.

The name of the experiment comes from *The Wonderful Wizard of Oz* story, in which an ordinary man hides behind a curtain and pretends to be a powerful wizard.

DTMF (**D**ual **T**one **Multiple** **F**requency) describes the possibility to allow the user to use the telephone number pad to enter numbers as touch tones.

PBX (**P**rivate **B**ranch E**x**change) is the abbreviation of a telephone system.

4.1 VoiceXML

We use VoiceXML for the example code and a specification of the tags can be found in [15]. VoiceXML is a language in the XML format that has been developed by the VoiceXML forum. The forum was founded by IBM, Motorola, and AT&T to access networked services via a telephone. Version 2.0 of the VoiceXML specification has been release in February 2003. In 2004, it was standardized by the W3C.

The goal of VoiceXML was to have a means for easy development of voice based applications. The main domain is telephony based applications. However, the concept can also be used to serve mobile devices.

VoiceXML documents consist of several dialogs (`<form>` or `<menu>`). These dialog elements describe the interaction with the user. Besides there are tags to support input fields, control elements, variables, event handler, and blocks containing procedural logic.

5 VUI Design Patterns

5.1 Dialog Strategy

This section names the four main concepts of voice based interaction based on the major requirements for voice user interfaces.

These major requirements are specified by ETSI in [4]. Voice based interaction must be

- Easy to learn,
- Easy to remember, and
- Natural.

For the speech recognizer it is also important that the commands are acoustically different to reduce recognition errors.

There are four main approaches, also know as *dialog strategy* [2], to use voice as an input medium:

Command & Control In Command & Control environments the application can handle a voice command menu that contains voice commands. This can be used to enable the user controlling the computer without the need of a mouse or keyboard.

Menu Hierarchy If the user has to provide data that can be gathered through a selection process and the options to be presented to the user are interrelated in a hierarchy, or can be made to appear that way, the application can prompt her with a set of options from which she may choose one.

Form Based This is the simplest and most common type. Form items are executed exactly once in sequential order to collect data from the user as if she was filling out a form. The computer directs the conversation, prompting the user for the next field to be filled.

Mixed Initiative In mixed initiative dialogs, both the computer and the human direct the conversation. The user can speak freely. Input items are set and the corresponding actions are taken in response. This dialog strategy requires natural language understanding (NLU) capabilities, confronting us again with the vision of the computer as a conversational counterpart.

In command & control environments, the user is the active part, controlling the computer by voice. This is why it is also called *user initiative*.

Applications that we find today are most of the kind of menu hierarchy and form based, or a combination of both. In these environments, the

computer directs the dialog, while the user can only react. These dialog strategies are also called *system initiative*.

Some applications using *mixed initiative* exist, but since this requires a higher programming effort (having a direct relation to the money being paid for development) they are not very common. However, this dialog strategy is the most promising to be accepted by users.

Patterns for MENU HIERARCHY, FORM FILLING, and MIXED INITIATIVE have been introduced in [13]. Here we extend those dialog strategies with the missing dialog strategy.

Command & Control

Also Known As

Also know as
User Initiative

Intent

Provide a means for the user to control an application using voice.

Context

In Command & Control environments the application can handle a voice command menu that contains voice commands.

Problem

How to enable the user controlling an application by voice?

Forces

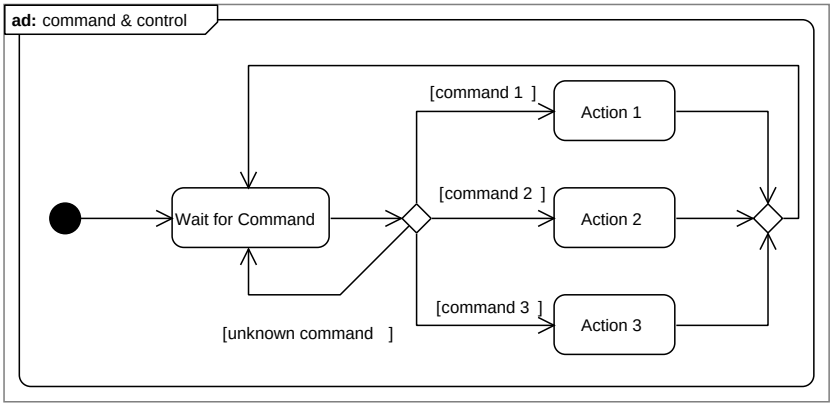
- The user has to learn a special vocabulary
- Does not scale
- Commands may sound similar and are hard to distinguish by the recognizer
- Commands should be easy to learn and natural

Solution

Determine the commands that the user should be able to initiate by voice commands. Conduct wizard-of-oz experiments to determine commands that are natural or use a predefined command set like the one from ETSI [4]. The ETSI command sets are selected to be acoustically different and natural. Associate a command for each action to perform. Invoke the action, if the user said the word. Expect the next command.

Structure

This is illustrated in the following figure.



After the start, the application waits for a command. If the user enters a command by voice, the appropriate action, e.g. *Action 1* is invoked. After invocation, the application waits for the next command.

Consequences

- ☺ The principle is easy to understand.
- ☹ Hard to recover, if an unwanted action was triggered.
- ☹ The force to learn a special vocabulary remains unsolved.
- ☹ The force that the user has to learn a special vocabulary remains unsolved.

Known Uses

The new Windows Vista allows to control the desktop using voice commands, e.g., the command *open notepad* launches the notepad application.

Katalavox [9] offers a voice control for phocomelic drivers. Drivers can control all functionalities, like turning signals or controlling the whiper by voice commands.

Related Patterns

Sample Code

The following code expects the commands *start* and *stop*. Depending on the command, the application executes the corresponding method in a Java archive *application.jar*. Afterwards it returns expecting the next command.

```
<form id="entercommand">
        <field name="command">
    <grammar src="http://grammarlib/commands.grxml"
```

```
        type="application/srgs+xml"/>
      <filled>
        <if cond="command=='start'">
          <goto next="#startapp"/>
        <if cond="command=='stop'">
          <goto next="#stopapp"/>
        <else/>
          <reprompt/>
        </if>
      </filled>
    </field>
</form>

<form id="startapp">
  <object name="start"
      classid="method://start"
      data="http://application.jar"/>
  <goto next="#entercommand"/>
</form>

<form id="stopapp">
  <object name="start"
      classid="method://start"
      data="http://application.jar"/>
  <goto next="#entercommand"/>
</form>
```

5.2 Authentication Over the Telephone

Many voice application have the need to authenticate the user. Garfinkel [7] names three points how authentication is described in computer systems by presenting:

- Something that you know, e.g., a password
- Something that you have, e.g., a phone
- Something that you are, e.g., biometric data

In other words, most systems actually fail because they rely on

- Something that you have forgotten
- Something that you have lost
- Something that you no longer are

However, there are many cases, where authentication is required. Most of what is written concentrates on web based authentication or have a graphical user interface in mind. The only technology that explicitly takes respect to voice are based on biometric data. It turned out, that this technology is even less accurate than speech recognition and plays only a minor role.

Delivering critical information over audio is a crucial factor. The audio domain does not guarantee secure delivering of information. Since security and privacy is an issue for several applications, we extend our pattern language to these aspects. Patterns for authentication are known for graphic oriented applications. These patterns are hard to transfer to audio, since these patterns do not take respect to the special characteristics of this medium.

In addition to the challenges named in section 3 designers of voice based applications in telephony based environments have to master the following challenges:

Conversation If the user is speaking to another person, or if a person that drops in or passes by addresses the user by saying something, the recognizer has no clue to distinguish these utterances to other persons from commands to control the system. In contrast to noisy environments, which is part of the recognition performance challenge, the risk of unwanted triggering of the recognizer is higher, since the user may use words, which are valid input but have the same source.

Privacy Being on the move, other persons may be around the user, while she is interaction with the system. Both, audio-input and -output should be hidden from these persons. In practice, this is a problem, which is impossible to solve. The only workaround is, not to deliver critical information via mouth & ear devices.

Service availability While the user is walking around using a mobile phone, some services may become no more available because the network is not reachable any more.

Phone As A Token

Intent

Automatically recognize the user to customize services and user experience.

Context

Voice applications can better customize the service and user experience when they can identify the user. In some services that users may call often or under difficult conditions such as time pressure or emergency, key/password entry is not an option. A mechanism for identifying and locating the user automatically is necessary, with a certain degree of confidence.

Problem

How to identify a user without asking her for credentials?

Forces

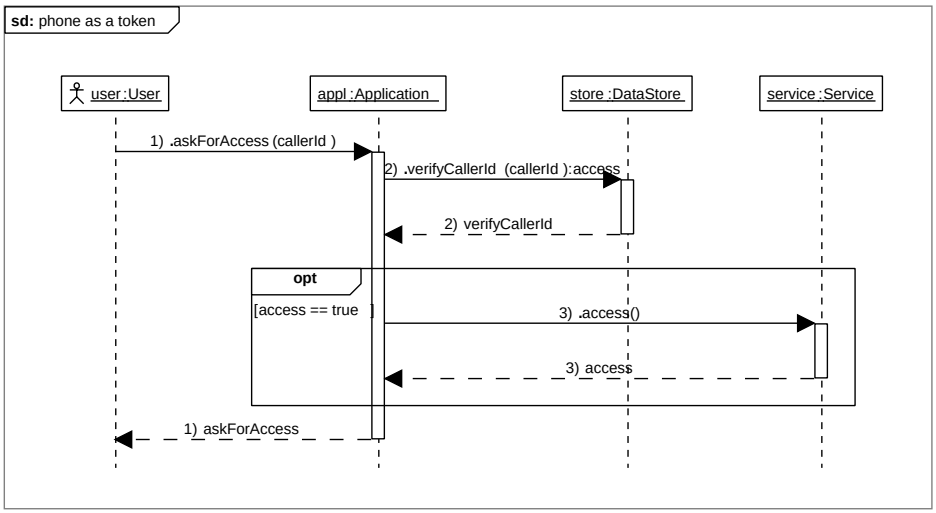
- The business rules require an authentication to access the service
- The permission has to be conveyed from the caller to the service provider
- The user does not want to authenticate each time
- Users might want to access services from multiple telephones
- The telephone number is based on something that the user has
- The user and the service provider trust one another
- The service provider can validate the telephone number of the user against an identity store

Solution

Obtain the caller's telephone number. Check if this number is privileged to access the service. If yes, grant access, if no refuse the access.

Structure

This is illustrated in the following figure.



Consequences

- ☺ User does not need to authenticate each time she wants to access the service.
- ☺ No one can hear the authentication
- ☹ Phone is just a token. There is no guarantee, that the current caller is identical to the person that registered for this service.
- ☹ The force that users may want to use another telephone or even looses her telephone remains unsolved.

Known Uses

A very popular application is the Memobox implemented by communication providers, which uses the user ID in order to automatically enable the access to the private recorded messages. In this case, the telephone (either fixed or mobile) becomes an identification (not authentication) token.

Related Patterns

Authorative Source of Data from Romanosky [10] describes how to recognize the source of data.

Token from Fernandez [5] describes a similar pattern more general, while we focus on the audio context.

Context Aware Call makes use of the caller id to personalize services.

Speaker Verification can be used in conjunction to enhance security.

Sample Code

The following code checks, if the callers telephone number is stored in the data store. If is present, the user can access the service. If they not, the user is disconnected.

```
<form id="authenticate">
  <object name="access"
      classid="method://validateNumber"
      data="http://application.jar"/>
    <param name="number" expr="session.callerid"/>
  </object>

  <block>
    <if cond="access == true'">
      <goto next="#service"/>
    <else/>
      <goto next="#deny"/>
    </if>
  </block>
</form>

<form id="deny">
  ...
  <hangup/>
</form>

<form id="service">
  ...
</form>
```

PIN

Intent

Authenticate a user

Context

A user needs to access a telephony based service. The service requires the client to present credentials that are randomly chosen for authentication so that additional controls such as authorization and auditing can be implemented.

Problem

How to verify the credentials that are presented by the user?

Forces

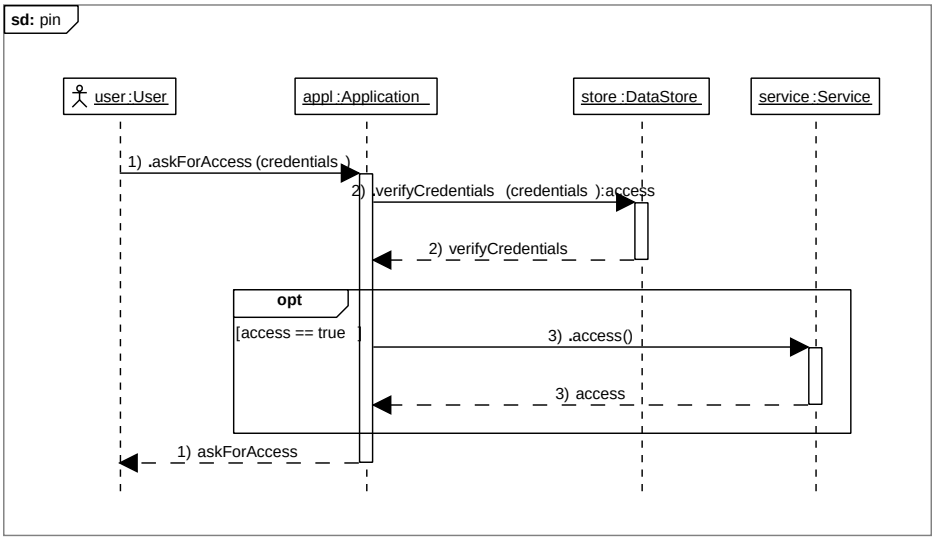
- The credentials that the user presents are based on shared secrets
- The user and the service provider trust one another
- The service provider can validate the credentials from the user against an identity store
- Others can hear the credentials
- Some telephones do not support DTMF tones or require the user to enter a code to bypass the PBX
- The users wants to access service from multiple telephones
- The recognizer may misrecognize correctly entered credentials
- Random words are hard to recognize
- Random words that are chosen by a user are easier to remember
- Numbers can be recognized with an accuracy close to 100%
- Numbers are hard to remember

Solution

Use PASSWORD ALGORITHM to create a sequence of numbers, store it in a secure data store and deliver the sequence of numbers to the user. Ask the user for this password, before accessing the service. Expect the user to enter the credentials using speech or the telephone number pad. Validate the credentials against the credentials stored in the database and grant access if the credentials match. If the credentials do not match give the user another try to enter her credentials. It is common to allow the repetition for at most three times.

Structure

This is illustrated in the following figure.



Consequences

- ☺ Can be used from any telephone
- ☺ Only users knowing the credentials can access the service
- ☺ The service can be accessed from any telephone
- ☹ The force that numbers are harder to remember than words chosen by the user remains unsolved

Known Uses

The German Postbank offers a telephone banking service that requires the user to enter a pin before she can access the service. The application can be called at +49 (180) 30 40 700.

Related Patterns

PASSWORD ALGORITHM from Riehle [11] can be used to generate credentials.

Can be combined with PHONE AS A TOKEN to enhance security.

SPEAKER VERIFICATION can be used to select a word that is easier to remember.

FORM FILLING can be used to enter the credentials.

Sample Code

The following code retrieves the stored credentials from the data store and saves it in the variable *pin*. Then the user is asked to enter her pin. If the entered credentials and the pin match, the user can access the service. If they do not match, the user has two additional chances before she is disconnected.

```
<form id="entercredentials">
    <var name="count" expr="1"/>

    <object name="pin"
       classid="method://retrievePin"
       data="http://application.jar"/>

        <field name="credentials">
      <prompt>
        Please enter your pin.
      </prompt>
      <grammar src="http://grammarlib/numbers.grxml"
        type="application/srgs+xml"/>
      <filled>
        <if cond="credentials == 'pin'">
          <goto next="#service"/>
        <elseif cond="count == 3"/>
          <goto next="#deny"/>
        <else/>
          <assign name="count" expr="count + 1"/>
          <reprompt/>
       </if>
     </filled>
   </field>
</form>

<form id="deny">
  ...
  <hangup/>
</form>

<form id="service">
  ...
</form>
```

Speaker Verification

Intent

Authenticate a user

Context

A user needs to access a telephony based service. The service requires the client to present credentials that are chosen by the user for authentication so that additional controls such as authorization and auditing can be implemented.

Problem

How to authenticate a user?

Forces

- The business rules require an authentication to access the service
- The permission has to be conveyed from the caller to the service provider
- The user does not want to authenticate each time
- Voice based biometrics is an imperfect technology
- Random words that are chosen by a user are easier to remember
- Biometric data of the voice may vary, e.g., if the user caught a cold
- The user wants to access the service from multiple telephones
- Authentication must be secret

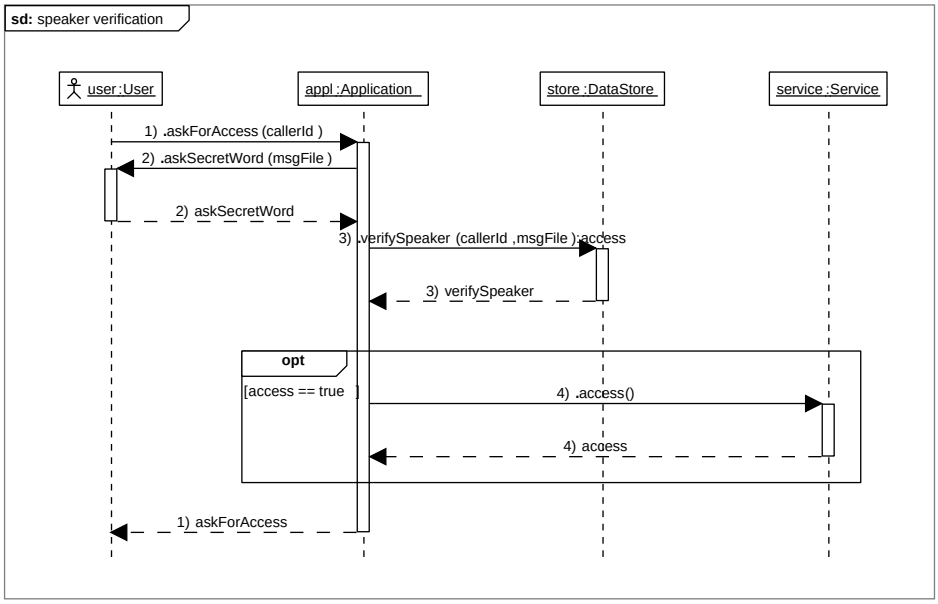
Solution

Use the user's phone number as an identifier. Alternatively use something to verify against, e.g., an account number. Ask the user to enter a word. Determine the biometric factors of the utterance using speaker verification technology and store them in a database together with the identifier.

If the user wants to access the service, determine the identifier and ask her for this word. Determine the biometric factors and compare them with the stored biometric factors. Grant access if they match. Deny access if they do not match.

Structure

This is illustrated in the following figure.



Consequences

- ☺ Can be used from any telephone
- ☺ Only the user can access the service
- ☺ Other persons who heard a user entering the word, can not access the service
- ☹ A recording of the authentication process can be used to bypass the authentication process
- ☹ Voice based biometric is imperfect. If the user, e.g. caught a cold, the determined data differ from the original recording

Known Uses

Related Patterns

Enhances PHONE AS A TOKEN to obtain more security.

Sample Code

SPEAKER VERIFICATION can not be implemented with the current version 2.1 of VoiceXML. However, the discussion of the upcoming 3.0 release includes the speaker verification feature as a module [16]. Currently the implementation is possible, but has to go the way of an intermediately stored file.

```
<form id="entercredentials">
    <var name="count" expr="1"/>

        <field name="credentials">
      <prompt>
        Please enter your account number
      </prompt>
      <grammar src="http://grammarlib/numbers.grxml"
        type="application/srgs+xml"/>
     </filled>

     <record name="msg" beep="true" maxtime="10s"
       finalsilence="4000ms" type="audio/x-wav">
       <prompt timeout="5s">
         Say your secret word after the beep.
       </prompt>
       <noinput>
         I didn't hear anything, please try again.
       </noinput>
     </record>
   </field>

  <object name="access"
     classid="method://verifySpeaker"
     data="http://application.jar"/>
    <param name="number" expr="session.callerid"/>
    <param name="number" expr="msg"/>
  </object>

  <block>
    <if cond="access == true'">
      <goto next="#service"/>
    <else/>
      <goto next="#deny"/>
    </if>
  </block>
</form>

<form id="deny">
  ...
  <hangup/>
</form>

<form id="service">
  ...
</form>
```

5.3 Voice and Sound in Consumer Devices

Adding electronics to control devices, tools and appliances has been a common practice for almost two decades now. These products embed different functionality and operation modes that rely on the ability of storing information or internal "state" . This information is not always visible from the outside, either because the device is very simple or because the user may not be able to look at it. Therefore, sound and voice play a major role in how we perceive, react and operate a device.

In this section we begin with a new area regarding the use of sound and voice in everyday products, to analyze and discover how sound and voice interfaces are being used.

Subtle Awareness

Intent

Raise the user's awareness over the current status of the device, application or task.

Context

In VUIs, users are usually unaware of events and state changes that take place, since no visual clues may be available about the system's state. For instance, let's consider the case of a typical answering machine: How does the owner know without looking at the machine if there are pending messages? One possibility is of course an explicit action from the user, but in many scenarios such engagement is time-consuming and expensive.

Problem

How to make the VUI user aware about relevant information without forcing explicit verification?

Forces

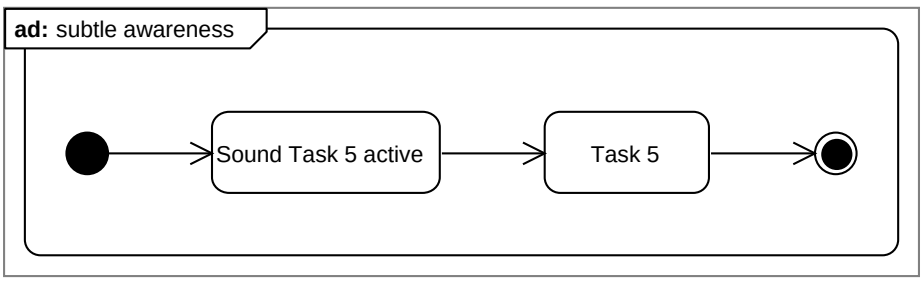
- No visual cue is available or visible.
- Querying the device through a VUI interface takes time, cumbersome or expensive.

Solution

Encode each event class with a specify tone or sound that users can identify and play that sound when the relevant task is active.

Structure

This is illustrated in the following figure.



Consequences

☺ Users become aware of changes and events without having to explicitly request the information.

☺ Very efficient when users operate the application or device remotely.

☹ Cognitive overload may occur when many different events are encoded with sounds. This overload tends to increase as further devices encode their own events using different sound cues

☹ The feedback about the internal state (e.g. "There are new messages")is available to all people accessing or using the device.

Known Uses

This is a pattern available in most answering machines today: after or before the welcome message, the device will play a particular tone or sequences of tones to quickly let the caller know whether the device has unread messages stored.

Related Patterns

A related pattern from the hypermedia/web design is BEHAVIOR ANTICIPATION [12], which conveys information about the destination of a link so the user may decide whether to follow a particular link or not. In a sense, SUBTLE AWARENESS plays a similar role by letting the user know without requiring further/explicit action.

Can be used to provide feedback in COMMAND & CONTROL environments.

6 Summary

This paper continues the previous work in the area of VUI patterns, presenting new and more specialized patterns, as well as introducing three specific areas of VUI design such as *Authentication* and *Data Entry*.

COMMAND & CONTROL completes the dialog strategy patterns.

The domain of authentication opens a new field, that is widely explored for GUI oriented applications, without taking respect to the special problems inherent to the audio domain. PIN, PHONE AS A TOKEN, and SPEAKER VERIFICATION are representatives for the three main aspects of authentication something that you know, something that you have, and something that you are. They show that some of the results of security patterns can be transferred to the audio domain. However, the use of audio adds new forces, like *Others can hear what you are saying*, because voice can not be hidden from others, like the representation of a user's password as stars in a password field.

SUBTLE AWARENESS investigates the use of audio clues as an additional means to provide information.

References

[1] Alice Chatterbox. `http://www.alicebot.org/`.

[2] Michael H. Cohen, James P. Giangola, and Jennifer Balogh. *Voice User Interface Design*. Addison-Wesley, Boston, January 2004.

[3] Andy Dearden and Janet Finlay. Pattern Languages in HCI; A Critical Review. *Human-Computer Interaction*, 21, 2006.

[4] ETSI. Human factors (HF); user interfaces; generic spoken command vocabulary for ict devices and services. Technical report, ETSI, April 2000. `http://www.etsi.org`.

[5] E. B. Fernandez and Günther Pernul. Patterns for Session-Based Access Control. In *Conference on Pattern Languages of Programs, PLoP 2006*, 2006.

[6] Norman M. Frazer and G. Nigel Gilber. Simulating speech systems. In *Computer Speech and Language*, volume 5. Academic Press Limited, 1991.

[7] Simson R. Garfinkel. *Design Principles and Patterns for Computer Systems That are Simoultaneously Secure and Usable*. PhD thesis, Massachusetts Insitute of Technology, 2005.

[8] Jabberwacky Chatterbox. `http://www.jabberwacky.com/`.

[9] Katalavox. Voice Control in Automobiles. `http://www.katalavox.com`, 2006.

[10] Sasha Romaosky. Enterprise Security Patterns. In *Conference on Pattern Languages of Programs, EuroPLoP 2002*, 2002.

[11] Sasha Romaosky. Password Patterns. In *Conference on Pattern Languages of Programs, EuroPLoP 2002*, 2002.

[12] Gustavo Rossi, Daniel Schwabe, and Fernando Lyardet. User interface patterns for hypermedia applications. In *AVI '00: Proceedings of the working conference on Advanced visual interfaces*, pages 136–142, New York, NY, USA, 2000. ACM Press.

[13] Dirk Schnelle and Fernando Lyardet. Voice User Interface Designt Patterns. In *EuroPLoP 2006 Conference Proceedings*, 2006.

[14] Dirk Schnelle, Fernando Lyardet, and Tao Wei. Audio Navigation Patterns. In *EuroPLoP 2005 Conference Proceedings*, 2005.

[15] VoiceXML. `http://www.w3.org/TR/voicexml20/`, March 2004. accesses on 08/28/2006.

[16] Sneak preview: VoiceXML 3.0. `http://www.w3.org/Voice/2006/voicexml3.pdf/`.

[17] Joseph Weizenbaum. Eliza A computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, 1966.