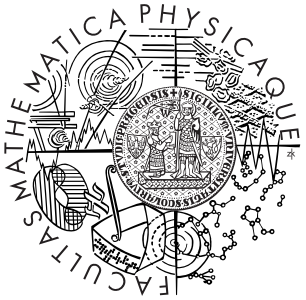
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Michal Ficek

JAWS framework pro ozvučování nestandardních aplikací

Katedra teoretické informatiky a matematické logiky

Vedoucí bakalářské práce: Mgr. Pavel Cejnar
Informatika: Programování

2006

Poděkování patří mému vedoucímu Mgr. Pavlu Cejnarovi za jeho pomoc při přípravě této práce.

Děkuji rodičům za jejich trpělivost a rodinné zázemí.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 25. července 2006 Michal Ficek

Obsah

Název práce: JAWS framework pro ozvučování nestandardních aplikací
Autor: Michal Ficek
Katedra (ústav): Katedra teoretické informatiky a matematické logiky
Vedoucí bakalářské práce: Mgr. Pavel Cejnar
e-mail vedoucího: pavel.cejnar@st.cuni.cz

Abstrakt: Aplikace JAWS (www.freedomscientific.com) je schopná zpřístupnit ovládání počítače nevidomým uživatelům. Aplikace s grafickými ovládacími prvky nahrazujícími standardní ovládací prvky Windows nebo aplikace, jež nemají plnou podporu ovládání z klávesnice, musí být dozvučeny přes skriptovací jazyk JAWS. V této práci popisujeme vytváření skriptu pro vybranou nestandardní aplikaci a analyzujeme, které části skriptu je možné generovat automaticky. Zároveň představujeme aplikaci JAWS framework, včetně návodu na použití, programátorské dokumentace a diskuze řešení vzhledem k specifikaci projektu.

Klíčová slova: JAWS, COM, automatické generování, skript

Title: JAWS Framework for Non-standard Applications
Author: Michal Ficek
Department: Department of Theoretical Computer Science and Mathematical Logic
Supervisor: Mgr. Pavel Cejnar
Supervisor's e-mail address: pavel.cejnar@st.cuni.cz

Abstract: The JAWS application (www.freedomscientific.com) is capable of allowing access of blind users to computer controls. Applications with graphical control elements that often substitute for standard controls of Microsoft Windows or applications that don't support full keyboard control have to be modified with JAWS scripting language to give out sounds and allow control of the sight impaired.
This project describes creation of a script for specific application and analyzes what parts of the script can be automatically generated. At the same time we introduce JAWS Framework - application with included users manual, technical documentation and discussion of solutions according to the specification of our project.

Keywords: JAWS, COM, automatic generation, script

Kapitola 1

Úvod

Používání programů běžících pod operačním systémem Microsoft Windows je díky jejich zažitému vzhledu uživatelského prostředí a známému chování ovládacích prvků velmi snadné. Programátoři mají k dispozici řadu vývojových nástrojů, kterými mohou během chvíle vytvořit příjemné ovládání aplikace. Málokdo z nich se ale zamyslí, zdali je takovéto ovládání použitelné i pro nevidomé uživatelé. Ti jsou často odkázání pouze na specializovaný software, který čte, co se na obrazovce nachází. V této práci se zabýváme aplikací JAWS (Job Access With Speech). Takový program nemůže být všemocný a tak často naráží na grafické popisky tlačítek, nestandardní ovládací prvky, nebo dokonce celé nestandardní ovládání aplikace. Jedná se o takové aplikace, na nichž ozvučovací software mlčí a neumí od sebe rozeznat jednotlivé ovládací prvky, pokud je vůbec nalezne.

V tuto chvíli nastupuje programátor ovládající skriptovací jazyk JAWS, který skriptem „vysvětlí“ ozvučovacímu programu, kde se v kterých místech nachází ovládací prvek aplikace, jak se s ním má zacházet, kdy je aktivní, jak na něj přistoupit a řadu dalších informací. Jistě ne všechny aplikace jsou programu JAWS do takové míry nepřátelské. Někdy stačí pouhých pár řádek skriptu, které upraví chování nějaké drobnosti v aplikaci.

V případě rozsáhlejších aplikací a tím i delších skriptů nastává často úmorná práce, kdy je třeba pečlivě ladit a zkoušet naprogramovaný skript, dopisovat a kopírovat již hotové části kódu na jiná místa skriptu apod. Pro ulehčení práce při skriptování jsme navrhli aplikaci JAWS framework. Ta pomocí přívětivého uživatelského prostředí umožní zadat vidomému uživateli informace o ozvučované aplikaci, oknech aplikace a ovládacích prvcích a potom vygeneruje z univerzální šablony výsledný skript. Ušetří tak řadu

drahocenného času programátorovi, který pak může výsledný skript pouze upravit na míru ozvučované aplikaci.

Přehled dalších kapitol

Tato práce je členěna na kapitoly, odpovídající postupu vytváření celého projektu JAWS framework. V následující kapitole představujeme stručný úvod do psaní skriptů pro aplikaci JAWS. Vysvětlujeme základní pojmy a příkazy používané ve skriptech, které jsou nezbytné pro porozumění příkladům uváděných v dalších kapitolách. Předpokládáme, že čtenář má alespoň základní znalosti práce s aplikací JAWS.

Třetí kapitola popisuje analýzu a implementaci skriptu pro vybranou nestandardní aplikaci. Ukazuje možné řešení problémů při skriptování a obsahuje několik příkladů nejzajímavějších částí skriptu.

Čtvrtá kapitola srovnává aplikaci JAWS framework se specifikací a odůvodňuje řešení použitá při implementaci. Objasňuje kroky, které nás vedly k výběru jednotlivých postupů z více možných.

Pátá kapitola se zabývá novým ozvučení nestandardní aplikace pomocí JAWS framework. Obsahuje výčet změn ve výsledném skriptu oproti skriptu původnímu. Poukazuje na vylepšení, jehož doznalo ovládání aplikace s novým skriptem.

Nedílnou součástí bakalářské práce jsou i přílohy. V příloze A naleznete specifikaci programu, podle které byla aplikace analyzována.

V příloze B naleznete obsah přiloženého CD-ROM s aplikací, naprogramovanými skripty, uživatelskou a programátorskou dokumentací a touto prací v elektronické podobě.

Kapitola 2

Základy skriptování

V této kapitole představujeme skriptovací jazyk JAMAL aplikace JAWS verze 5.0. V současné době existují verze vyšší (JAWS 6.2, JAWS 7.1), ale uvedené principy skriptování v nich zůstávají stejné.

Velká část aplikací je programem JAWS ozvučena korektně. Kdy a co má JAWS říct vzhledem k informacím na obrazovce se řídí souborem „standardních skriptů“ *default.jss*, který je vývojáři JAWS dodáván v každé verzi programu. Pro aplikace vykazující odchylky, s nimiž si JAWS při ozvučování neporadí, jsou určeny soubory „aplikačních skriptů“.

Soubor skriptů obsahuje skripty a funkce, které svými příkazy oznamují JAWS, jak se má v aplikaci orientovat a co má uživateli oznamovat, v závislosti na různých událostech. Modifikací existujících skriptů nebo napsáním nových je možné ozvučit i značně nestandardní aplikace.

K editaci a vytváření nových skriptů lze použít Manažer skriptů JAWS, který nabízí řadu nástrojů pro výběr a vkládání funkcí, vytváření nových funkcí a skriptů, provádí kompilování skriptů při uložení apod. Manažer skriptů spustíte vybráním položky *Manažer skriptů* v dialogu *Spustit JAWS nástroje*, který zobrazíte stiskem klávesové zkratky **Insert**+**F2** při běžícím programu JAWS.

2.1 Názvosloví

Skript je posloupnost příkazů, které vykonávají určitou činnost. Skript je aktivován klávesovou zkratkou.

Funkce je také posloupnost příkazů, které mají určitý úkol, ale není aktivo-

vána klávesovou zkratkou. Funkce jsou volány ze skriptů nebo z jiných funkcí.

Soubor skriptů je skupinou skriptů, funkcí a deklarací proměnných a konstant uložených v souboru s příponou *.jss* (JAWS script source).

Kompilace je akce, kdy se soubor skriptů převede na soubor s příponou *.jsb* (JAWS script binary), s nímž JAWS pracuje při ozvučování. Změny v souboru *.jss* bez následné kompilace nejsou JAWS rozpoznány.

2.2 Skripty a funkce

2.2.1 Aktivace skriptů

Skripty se aktivují klávesovými zkratkami. Přiřazení kláves jednotlivým skriptům je možné provést v Manažeru klávesnice programu JAWS, přístupným z dialogu *Spustit JAWS nástroje* po stisku **Insert+F2** při běžícím programu JAWS. Výstupem Manažeru klávesnice je mapovací soubor *.jkm* (JAWS key map). Ten obsahuje seznam názvů skriptů a k nim názvy klávesových zkratek a má stejné jméno souboru jako spouštěcí soubor aplikace.

JAWS rozeznává dva druhy mapovacích souborů – standardní a aplikační. Standardní *default.jkm* obsahuje klávesové zkratky přiřazené skriptům v *default.jss* (viz příklad 2.2.1). Při stisku klávesové zkratky v ozvučované aplikaci hledá JAWS přiřazení kláves ke skriptům v aplikačním mapovacím souboru. Pokud najde stisknutou klávesovou zkratku, spustí skript k ní přiřazený. Nenalezne-li, použije přiřazení definované ve standardním souboru. Pokud je shodná klávesová zkratka ve standardním i aplikačním mapovacím souboru, má přednost soubor aplikační.

Příklad 2.2.1 Mapovací soubor - fragment souboru *default.jkm*

```
JAWSKey+F4=ShutDownJAWS
ŠIPKA VPRAVO=SayNextCharacter
ŠIPKA VLEVO=SayPriorCharacter
ŠIPKA NAHORU=SayPriorLine
ŠIPKA DOLŮ=SayNextLine
CONTROL+JAWSKey+F2=SaySpecialWindowClasses
JAWSKey+B=ReadBoxInTabOrder
CONTROL+HOME=TopOfFile
ALT+ŠIPKA DOLŮ=OpenListBox
```

2.2.2 Umístění skriptů

JAWS obsahuje velké množství souborů skriptů, uložených v adresáři skriptů a nastavení. Hlavním souborem je *default.jss*, jenž obsahuje skripty a funkce pro práci s Windows aplikacemi. Tento soubor se načítá při každém startu JAWS a je aktivní až do ukončení programu JAWS. Řídí ozvučování a ve většině případů definuje co a kdy se má uživateli oznámit.

Pro aplikace, které potřebují upravit nebo nově definovat, jak mají být ozvučeny, slouží aplikační skripty. Jedná se o soubory s názvem shodným jako spouštěcí soubor aplikace a příponou *.jss*. Aby jej JAWS při startu aplikace načetl, musí být zkompilovaný a musí být umístěn v adresáři skriptů a nastavení JAWS. Po skončení aplikace je skript z paměti uvolněn.

Skript je možné spustit i z jiného skriptu nebo funkce uvedením klíčového slova `PerformScript` za kterým následuje jméno skriptu.

2.2.3 Spouštění funkcí a jejich typy

Spouštění funkcí se řídí názvem a umístěním funkce. Po každém zavolání funkce ze skriptu nebo jiné funkce je prohledán aplikační soubor skriptů. Obsahuje-li funkci s daným názvem, je funkce spuštěna. V opačném případě se název funkce hledá ve standardním souboru skriptů. Pokud ani ten funkci neobsahuje, hledá se název funkce mezi názvy vestavěných (built-in) funkcí.

Funkce se od skriptů liší tím, že vrací hodnotu a není aktivována klávesovou zkratkou. Může také přijímat parametry hodnotou nebo odkazem. Základní typy funkcí jsou:

Uživatelské funkce jsou funkce zapsané v aplikačních skriptovacích souborech nebo ve výchozím souboru skriptů *default.jss*.

Vestavěné (built-in) funkce jsou funkce připravené vývojáři JAWS a jsou určené pro použití ve skriptech a uživatelských funkcí. Jejich seznam a podrobný popis je možné nalézt v souboru *builtin.jsd* v adresáři skriptů a nastavení JAWS.

Událost (event) je funkce volaná JAWS, jakmile nastane určitá událost okna. Jsou velmi důležité a způsobují, že JAWS oznamuje změny na obrazovce automaticky. Události není možné přidávat, je však možné modifikovat stávající funkce událostí.

Hlavní události a jejich význam:

- `AutoStartEvent` je vykonána vždy při startu aplikace nebo když se aplikace stane aktivní.
- `FocusChangedEvent` nastává při změně focusu v aplikaci, dialozích nebo dialogových ovládacích prvcích.
- `KeyPressedEvent` je vykonána pokaždé při stisku libovolné klávesy.
- `WindowDestroyedEvent` nastává při uzavření okna.

2.2.4 Zápis funkcí a skriptů

Skript začíná klíčovým slovem `Script`, následuje název skriptu a prázdné závorky. Tělo skriptu obsahuje libovolné příkazy skriptovacího jazyka a volání funkcí. Skript končí klíčovým slovem `EndScript`.

Funkce začíná typem návratové hodnoty. Nevrací-li funkce žádnou hodnotu, uvádí se typ `void` nebo se vynechává úplně. Klíčové slovo `Function` následované názvem skriptu ukončuje seznam parametrů uzavřený do závorek. Tělo funkce končí klíčovým slovem `EndFunction`.

2.3 Proměnné a konstanty

2.3.1 Proměnné

Deklarace proměnných ve skriptu nebo funkci začíná klíčovým slovem `var`. Následuje typ proměnné a její název; jednotlivé deklarované proměnné jsou oddělované čárkou. JAWS rozeznává také globální proměnné, které si uchovávají svou hodnotu po celou dobu běhu skriptu aplikace i po jejím opětovném spuštění. Jejich deklarace začíná klíčovým slovem `globals` a smí být pouze na začátku skriptovacího souboru. Použití globálních proměnných je možné kdekoli v souboru skriptů, z libovolné funkce nebo skriptu.

Přiřazování hodnot libovolným proměnným začíná klíčovým slovem `let`.

Proměnné ve skriptovacím jazyce JAWS mohou nabývat čtyř typů:

integer – celočíselná hodnota. JAWS přiřazuje všem proměnným typu integer počáteční hodnotu nula. Klíčové slovo pro deklaraci proměnné typu integer je `int`.

string – řetězec znaků. Může obsahovat písmena, číslice, interpunkční znaménka a mezeru. Řetězec se uzavírá do apostrofů, prázdný řetězec je

dvojice apostrofů bez mezery mezi nimi. Výchozí hodnotou řetězce je prázdný řetězec. Deklarace proměnné typu string se provádí klíčovým slovem `string`.

handle – handle okna, klíčovým slovem pro deklaraci je `handle`.

object – ukazatel na objekt. Je možné jej vytvořit voláním funkce `CreateObject`, nebo funkcí `GetObject` vytvořit ukazatel na objekt již existující. Parametrem obou funkcí je jméno COM třídy, která vrací *automation* objekt. Použití metod a vlastností objektu je realizováno tečkovou notací (viz příklad 2.9.2).

Globální proměnné mohou být deklarované i v externím hlavičkovém souboru. Časté je použití standardních globálních proměnných ze souboru *HjGlobal.jsh*. Připojení hlavičkového souboru je možné provést zapsáním klíčového slova `include`, za kterým následuje jméno souboru v uvozovkách.

2.3.2 Konstanty

Konstanty ve skriptovacím souboru JAWS získávají svou hodnotu při deklaraci a nelze je později měnit. Deklarace konstant se provádí na začátku skriptovacího souboru a začíná klíčovým slovem `const`. Následuje název konstanty, rovnítko a hodnota konstanty.

Obvyklé je použití konstant z hlavičkových souborů programu JAWS *HjConst.jsh*.

2.4 Zprávy

Všechny texty, které se oznamují uživateli, je vhodné zapsat do souboru zpráv (messages). Jedná se o soubor s příponou *.jsm* (JAWS script message), ve kterém jsou všechny zprávy uloženy. Výhodou použití tohoto umístění je snadné opravování zpráv bez předchozího dlouhého vyhledávání všech výskytů v souboru skriptů. Hlavním přínosem je oddělení textů určených uživateli skriptu od zdrojového kódu, čímž se skript stává přehlednějším.

Soubor zpráv se skládá z bloků. Blok začíná znakem `@`, za ním následuje jméno zprávy, kterým se zpráva označuje a používá v souboru skriptů. Na dalším řádku je text zprávy a celý blok končí znaky `@@` na novém řádku. Příklad 2.4.1 je ukázkou části souboru zpráv.

Zprávy ze souboru *.jsm* je možné použít v souboru skriptů jako řetězcové konstanty. Pro oznámení zprávy uživateli slouží několik funkcí, doporučená je `SayFormattedMessage`.

Příklad 2.4.1 Soubor zpráv

```
Messages
@msgChecked
zaškrtnuto
@@
@msgNotchecked
nezaškrtnuto
@@
EndMessages
```

2.5 Virtuální prohlížeč

K oznámení delších textů uživateli slouží virtuální prohlížeč. Jedná se o součást JAWS – okno s bílým podkladem, na němž je zobrazen černý text fontem Arial. Při aktivaci virtuálního prohlížeče se stane aktivním i kurzor v jeho okně, kterým může uživatel text číst, označovat i kopírovat do schránky.

2.6 Rámce

K označení oblastí na obrazovce nebo oknech aplikace slouží rámce. Jedná se o průhledná okna pouze s rámečkem, jejichž velikost a umístění je možné určit v Manažeru rámců. Rámce mohou reagovat na události v části okna, které ohraničují: změnu textu, přítomnost kurzoru a jiné. Pozici rámců lze určit relativně k oknu aplikace, nebo absolutně vzhledem k obrazovce. K vypnutí nebo zapnutí jednotlivých rámců slouží pravidla pro aktivaci – titulek okna, třída okna, text v okně a další.

2.7 Řízení běhu skriptu

Skriptovací jazyk JAWS podporuje podmíněné skoky. K větvení běhu programu je možné použít konstrukce `if..then..else..endif`. Pro uve-

dení další větve lze využít konstrukce `if..then..elif..then..endif`, kde `elif..then` větví může být libovolný počet.

K řízení běhu programu je možné taktéž využít cyklus `while`.

Jelikož skript běží rychleji, než se mohou stát změny v aplikaci, které jsou skriptem vyvolány, používá se při skriptování funkcí pro pozdržení běhu skriptu:

- Funkce `Pause` pozastaví běh skriptu do doby, než ostatní aplikace ukončí své úlohy.
- Funkce `Delay` pozdrží běh skriptu na dobu určenou parametrem funkce v milisekundách.

2.8 Zpracování klávesových zkratek

Je-li v ozvučené aplikaci nějaké klávesové zkratce přiřazen skript, je po jejím stisku skript spuštěn. Důležitou vlastností však je, že klávesová zkratka není do aplikace předána. Má-li skript zajišťovat pouze dodatečnou funkcionalitu ke klávesové zkratce, jejíž stisk aplikace vyžaduje, je nutné do skriptu přidat funkci pro „propuštění“ klávesové zkratky. Vestavěná funkce `TypeKey` propustí do aplikace klávesy uvedené ve svém parametru a aplikace je obdrží aniž by poznala, že je uživatel nestiskl na klávesnici.

Po funkci pro propuštění klávesových zkratek je zpravidla nutné použít některou z funkcí pro pozdržení běhu skriptu, aby se změny aktivované klávesami v ozvučované aplikaci mohly projevit. Příkladem může být čekání na zobrazení dialogu pro otevření souboru po volání funkce `TypeKey("Ctrl+O")`.

2.9 Ukázka skriptu

Příklad 2.9.1 je klasickým úvodem do skriptování. Postup vytvoření tohoto skriptu krok za krokem je možné nalézt v [BS].

Příklad 2.9.1 Skript „Hello world“

```
Script HelloWorld ()
SayString ("Hello␣world!")
EndScript
```

Složitější příklad 2.9.2 demonstruje většinu obratů skriptovacího jazyka JAWS vysvětlených v předchozích odstavcích.

2.10 Další informace

Podrobnější informace o základech skriptování naleznete v [BS]. Rozsáhlý manuál skriptování včetně metodiky psaní skriptů, řady příkladů i úloh pro procvičení skriptování je [Go]. Řadu cenných informací pro vytváření skriptů můžete najít v [JS].

Příklad 2.9.2 Základní obraty skriptování - fragment souboru *JAWSframework.jss*

```
include "hjconst.jsh"    ;konstanty JAWS
include "hjglobal.jsh"   ;globalni promenne JAWS

globals                  ;deklarace globalnich promennych
  object JF,
  int g_iNacistKlavesovouZkratku

Void Function JFNavazatSpojeni ()
  if !JF then
    ;prirazeni do promenne JF vysledku volani vestavene funkce
    let JF = CreateObject ("Component.JFcomponent.1")
  endif
EndFunction

Script JFNazevSkriptu ()
  JFNavazatSpojeni()
  ;volani metody objektu JF
  JF.PutString(GetActiveConfiguration ())
EndScript

Script JFNazevOkna ()
var string vystup,
    handle hRealWindow,
    string sText
  let hRealWindow = GetRealWindow(GetFocus())
  let sText = GetWindowTextEx (hRealWindow, false, true)
  JFNavazatSpojeni()    ;volani uzivatelske funkce
  let vystup = GetWindowName(hRealWindow) +
;znaménko + spojuje retezce
       "|" + GetWindowClass (hRealWindow) +
       "|" + StringLeft (sText, 200)
  JF.PutString(vystup)
EndScript

Int Function JFShortcutHook (string scriptname, string frame)
  JFNavazatSpojeni()
  JF.PutString(GetCurrentScriptKeyName ())
  RemoveHook (HK_SCRIPT, "JFShortcutHook")
  TrapKeys(FALSE)
  return FALSE
EndFunction
```

Kapitola 3

Skript JAWS pro aplikaci Bílé stránky 2004-2005

Aplikace Bílé stránky 2004-2005 (dále jen Bílé stránky) jsme pro ozvučení vybrali pro její velkou míru nestandardního chování. Je naprogramována v jazyce Visual FoxPro verze 6. Její uživatelské rozhraní neposkytuje žádné podrobné informace o ovládacích prvcích, všechny mají shodný identifikátor (controlID) „0“ a třídu „eltes6c000000“. JAWS neumí v takovém případě rozlišovat ovládací prvky a na celé aplikaci pouze mlčí.

Následující části nejsou ukázkou postupu při vytváření nového jednoduchého skriptu. U většiny aplikací stačí ve skriptu napsat pár funkcí zlepšující chování ozvučené aplikace, ale pro Bílé stránky podobné řešení nestačí. Skript námi vytvořený je značně složitý a komplikovaný, protože zcela emuluje uživatelské rozhraní. Téměř ke všem významným klávesám je přiřazen skript, jenž v aplikaci provádí změny. Příkladem za všechny je změna ovládacích prvků klávesami **Tab** a **Shift**+**Tab**, které není možné přímo propouštět do aplikace, ale ve skriptu se určuje, který ovládací prvek je tím dalším nebo předchozím.

Řešení námi navržená jsou určena autorům skriptů k inspiraci, co lze použít, nebo čemu se naopak vyhnout. Použité skriptovací obraty mohou sloužit k odstranění případných problémů během ozvučování nebo alespoň minimalizaci jejich dopadů.

Všechny soubory skriptu naleznete na přiloženém CD v adresáři `Skripty pro Bílé stránky\Původní skripty\`.

3.1 Analýza

Prvním krokem ozvučování bylo zjišťování, kterým prvkům je přiřazena třída nebo identifikátor (controlID). Klávesovou zkratkou **Ctrl**+**Insert**+**F1** (při spuštěné aplikaci JAWS) je možné získat control ID, třídu a handle okna. Protože tyto hodnoty (až na handle) mají všechny ovládací prvky stejné, zjišťovali jsme, který text v aplikaci umí JAWS přečíst. Klávesovou zkratkou **Ctrl**+**Insert**+**W** se zobrazí veškerý text na okně. U většiny hlavních oken byl jediným přečteným textem titulek okna.

Tyto informace o aplikaci vedly k rozhodnutí, že skript pro ní určený bude stavového charakteru - bude si ve svých proměnných pamatovat aktivní ovládací prvek, a převezme kontrolu nad většinou kláves, které uživatel zadává. Soubor skriptů tak obsahuje pro běžné klávesové zkratky skripty, které na základě aktivního prvku buď konají akci, nebo je klávesa propuštěna do aplikace. Uvedené řešení přináší jisté obtíže (viz část 3.6.4), ale je prakticky použitelné a v mezích skriptovacího jazyka JAWS proveditelné.

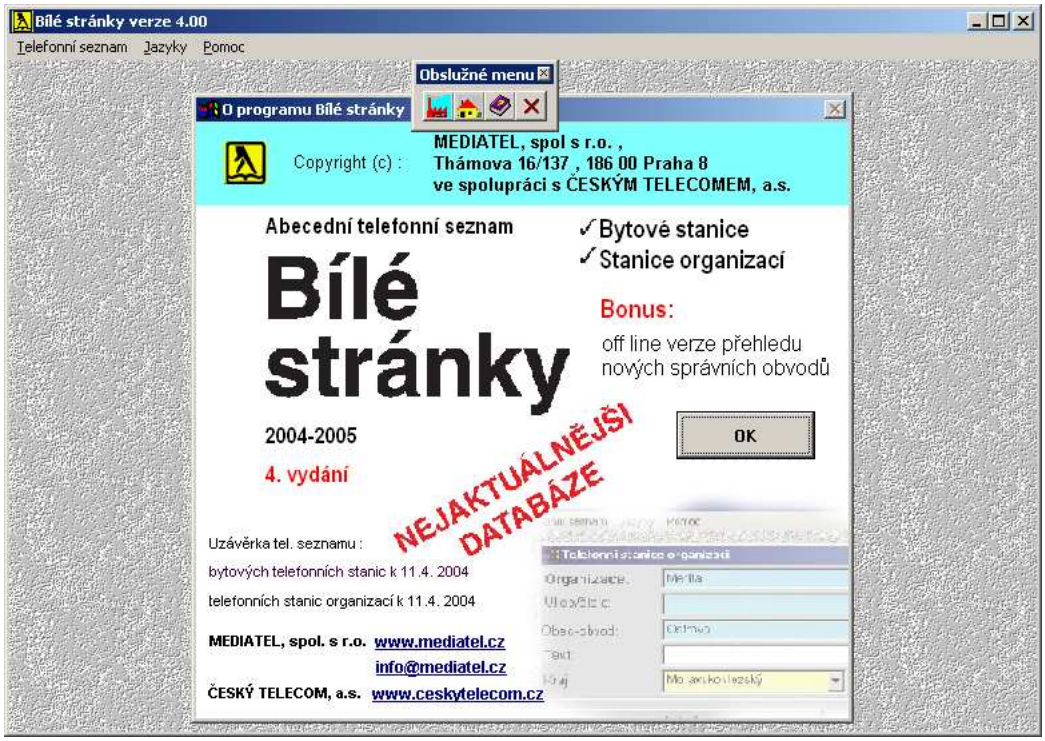
Potěšitelným zjištěním bylo, že menu aplikace Bílé stránky umí JAWS přečíst a tak není nutné jej ozvučovat. Následovalo vyhodnocení, které nabídky menu jsou pro základní práci s aplikací relevantní. Na nabídky, které uživatel nemá používat, je upozorněn v nápovědě k aplikaci, kterou zobrazuje skript po svém startu ve virtuálním prohlížeči. Například změna jazyka na nabídce *Jazyky* způsobí mimo jiné přejmenování titulků oken a skript pak již nerozezná okna, jejichž ovládací prvky jsou ozvučené.

3.2 Start aplikace

Po startu aplikace (obrázek 3.1) je zobrazen plovoucí panel nástrojů *Obslužné menu*, který nevidomý uživatel nevyužije. Není totiž přístupný žádnou zkratkou z klávesnice a všechny funkce na něm jsou stejně obsaženy v menu. Navíc překrývá některé ovládací prvky na oknech aplikace. Uzavření tohoto panelu nástrojů a zobrazení úvodního textu ve virtuálním prohlížeči zařizuje skript v události `FocusChangedEvent`.

Ke zjištění, zda je aplikace spuštěna poprvé nebo zda došlo pouze k přepnutí mezi jinými programy klávesami **Alt**+**Tab** není možné použít příznaku v globální proměnné skriptu. I když se spolehneme na fakt, že proměnná obsahuje po startu skriptu nulovou hodnotu a po zpracování úkolů po startu bude skriptem nastavena na nenulovou hodnotu, JAWS si uchovává hodnoty globálních proměnných i po ukončení aplikace. Nový start aplikace

Obrázek 3.1: Bílé stránky po startu



zjistí skript umístěním kurzoru vestavěnou funkcí `MoveTo` do míst na plovoucím panelu nástrojů, kde se nachází ikony nápovědy. Je-li v tomto místě fialová barva ikony, skript plovoucí panel uzavře a zobrazí uvítací text. Protože není žádný způsob jak panel nástrojů znovu vyvolat z menu aplikace nebo jinak, jedná se o vyhovující řešení.

Úvodní text aplikace obsahuje doporučení a pravidla pro práci s programem. Seznamuje také uživatele s kroky v aplikaci, kterých se má vyvarovat, protože vedou k nestabilitě ozvučení (rychlé psaní textu) nebo k pádu aplikace vinou jejích chyb.

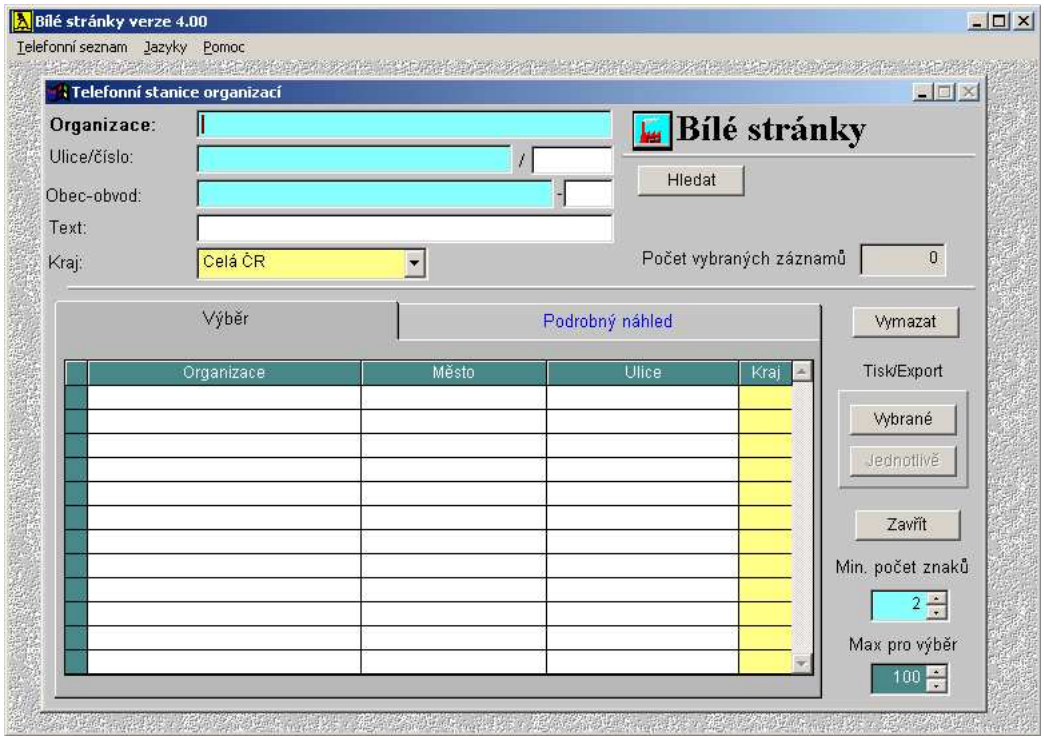
3.3 Práce s okny

Zjištění všech používaných oken a rozhodnutí, na kterých se budou ozvučovat které prvky, bylo nedílnou součástí návrhu skriptu. Dvě hlavní okna *Telefonní stanice organizací* (obrázek 3.2) a *Bytový telefonní seznam* mají skriptem všechny své prvky ozvučeny. Okna *O programu* a *Důležitá čísla* obsahují grafický text, který je na nich ve formě obrázku nakreslen. Pro použití těchto oken jsme text zobrazený v okně přepsali do zprávy v souboru zpráv a je zobrazen skriptem ve virtuálním prohlížeči při každém otevření okna.

Titulek každého ozvučeného okna jsme uložili do souboru zpráv. Titulek je skriptem porovnán s titulkem reálného okna, které získá v události `FocusChangedEvent` focus. Shoduje-li se nadpis reálného okna s některým z uložených, přepíná se ve skriptu na ovládací prvek na okně ležící – buď první, nebo naposledy aktivní. Není-li titulek shodný, nastavuje se aktivní ovládací prvek v aplikaci (viz část 3.4) na nulovou hodnotu a zobrazené okno je programem JAWS ošetřováno standardním způsobem. Toto chování je výhodné pro případ dialogů s chybovým oznámením, dialogů pro otevření souboru a dalších, které jednoduše není možné i při podrobném zkoumání aplikace postihnout všechny. Navíc tyto dialogy bývají typicky aplikaci JAWS přátelské – mluví na nich.

Každému oknu jsme ve skriptu určili inicializační funkci, která se volá po otevření okna v aplikaci nebo návratu do okna z jiného, které v aplikaci není ozvučeno. Inicializační funkce nastavuje proměnné skriptu s vlastnostmi ovládacích prvků a určuje aktivní ovládací prvek na okně.

Obrázek 3.2: Okno *Telefonní stanice organizací*



3.4 Implementace ovládacích prvků

Každému z ovládacích prvků jsme ve skriptu přiřadili unikátní celočíselný identifikátor. Podle něj se ve skriptech a funkcích pozná, který prvek je aktivní, a určuje se další běh skriptu. Přiřazení je tvořeno konstantou s názvem prvku a hodnotou identifikátoru podle pořadí, ve kterém jsme prvek do skriptu přidávali. Tento způsob se v pozdějších fázích ozvučování stal značně nepřehledným, protože žádným způsobem neoznačuje prvky stejného typu nebo ležící na stejném okně.

Identifikátor aktivního ovládacího prvku je po celou dobu běhu skriptu uložen v proměnné `iAktID`. Prvek se stane aktivním poté, co se do něj uživatel přepne klávesou **Tab** – skript `Tab` na základě aktivního prvku volí následující a přepíná do něj voláním funkce `VstupDoOvl`. Je důležité podotknout, že změnu ovládacího prvku nemusí uživatel v aplikaci vůbec zpozorovat (viz 3.6.4). Přepínáním prvků se rozumí změna hodnoty proměnné ve skriptu, která určuje, ke kterému ovládacímu prvku v aplikaci budou směřovat uživatelem stisknuté klávesy.

Funkce `VstupDoOvl` podle typu ovládacího prvku volá tzv. *vstupní funkci*, například `EdbVstup`, `LstVstup` apod. Vstupní funkce pro konkrétní typ je pro všechny ovládací prvky stejného typu pouze jedna, protože prvky se chovají stejně a tak i jejich zpracování ve skriptu bude probíhat shodným způsobem. Funkce zajišťuje nastavení proměnných skriptu s identifikátorem ovládacího prvku, provádí inicializaci nově aktivovaného prvku a oznamuje jeho název a typ. Na příkladě 3.4.1 jsou vysvětleny všechny důležité kroky.

Aktivace a přepínání ovládacích prvků pomocí vstupních funkcí přinesla výhodu ve snadném přidávání nových prvků stejného typu. Není-li pro ovládací prvek třeba speciálního zacházení, dá se použít funkce pro ošetření daného typu bez jakýchkoli úprav. Zároveň však lze funkci snadno doplnit a na základě identifikátoru aktivovaného ovládacího prvku přizpůsobit běh skriptu „na míru“.

Pozici ovládacích prvků pozná skript podle rámců, kterými jsme polohu prvků na okně aplikace určili. Rámce slouží skriptu pro umístění kurzoru na ovládací prvek, kliknutí na tlačítko, přepínání karet, zvyšování hodnot číselníku a další. Vstup kurzoru do rámce prvku je realizován funkcí `KlikNaOvl`. Pokud by měla aplikace standardní podporu ovládání z klávesnice, nebylo by rámců zapotřebí. Pro Bílé stránky jsme však museli emulovat i uživatelské rozhraní, a tak přesuny kurzoru z jednoho ovládacího prvku na jiný zajišťuje pouze skript.

Příklad 3.4.1 Funkce `EdbVstup` ze souboru *VFP6R.JSS*

```
Void Function EdbVstup (int iID)
    ;parametrem je ID nového aktivního prvku
    ;akce pro opuštění stávajícího aktivního prvku
  OpustPredchoziOvl(iAktID)
    ;určení nového aktivního prvku
  let iAktID = iID
    ;kliknutí na prvek - umístění kurzoru
  KlikNaOvl (iAktID)
    ;uložení typu aktivního prvku
  let sAktTyp = VratTyp (iAktID)
    ;načtení obsahu editačního pole
  let sAktObsah = VratObsahEdb ()
    ;hodnoty pro práci s textem editačního pole
  let iAktObsahDelka = StringLength (sAktObsah)
  let iAktPos = iAktObsahDelka
  let iAktOznaceno = 0
    ;oznámení popisu, názvu a obsahu aktivního prvku
  OhlasitAktualniOvl (VratPopis(iAktID),
                      VratNazevTypu(sAktTyp),
                      sAktObsah)
EndFunction
```

3.4.1 Editační pole

Editační pole bylo z hlediska času stráveného při skriptování nejsložitějším ovládacím prvkem. Základním problémem je naprostá absence hlasového výstupu programu JAWS. Ten nejen nepřečte jednotlivá písmena při změně pozice textového kurzoru, nýbrž ani vysvícený (označený) text. Hlavními body ozvučení editačního pole se staly

- načtení textu z editačního pole do proměnné skriptu JAWS
- práce s textem v editačním poli

3.4.1.1 Načítání textu

Pro načtení textu z editačního pole do proměnné skriptu jsme využili zřejmě jediného možného způsobu – schránku Windows. Zjistili jsme, že text je možné stiskem klávesové zkratky **Ctrl**+**A** označit a následně zkopírovat do schránky. Vlivem blíže neurčené chyby programu JAWS verze 5.0 nebo aplikace Bílé stránky se ne vždy po stisku klávesové zkratky **Ctrl**+**C** zkopíruje

text do schránky a ne vždy se obsah schránky funkcí `GetClipboardText` načte. Proto jsme ve skriptu implementovali smyčku realizovanou cyklem `while` s počtem deseti opakování, kdy v každé iteraci proběhne pokus o zkopírování textu editačního pole do schránky a načtení obsahu schránky do proměnné.

Aby bylo možné zjistit, zda byl načten text z editačního pole a ne ten, který již ve schránce byl, je na začátku funkce `VratObsahEdb` vložen do schránky řetězec „chyba“. Proběhne-li všech deset iterací cyklu, aniž by ze schránky bylo načteno cokoli jiného než kontrolní řetězec, je uživateli skriptem oznámena chyba při načítání textu z editačního pole.

Protože je nutné vzít v úvahu i prázdné editační pole, je před označením textu vložena skriptem do editačního pole v aplikaci funkcí `TypeString` značka s řetězcovou hodnotou „ëöä“. Tuto hodnotu jsme zvolili kvůli svému značně nepravděpodobnému výskytu v textu pole. Je-li do proměnné skriptu načtena ze schránky pouze tato značka, signalizuje to, že textové pole je prázdné. Použitý obrat přináší potřebu z neprázdného načteného textu ve proměnné skriptu značku vyjmout – značka může být umístěna kdekoli v textu a její pozice závisí na pozici kurzoru v editačním poli při volání funkce `TypeString`. Značku není možné přidat pokaždé na začátek nebo na konec načítaného textu, protože stisk kláves **Home** nebo **End** v editačním poli aplikace přesunuje kurzor do jiného ovládacího prvku. Je to nestandardní chování a chyba aplikace.

3.4.1.2 Práce s textem

Ovládání editačního pole vykazuje v aplikaci Bílé stránky nepřeberné množství chyb. Stiskem klávesy **Home** při textovém kurzoru umístěném na začátku editačního pole se kurzor přesune do předchozího ovládacího prvku. Podobně po stisku klávesy **End** je kurzor umístěn do následujícího ovládacího prvku. Při psaní přechází kurzor po vyčerpání délky textového pole plynule do dalšího editačního pole, do úplně jiného ovládacího prvku. Stejně tak i při mazání znaků klávesou **Backspace** se kurzor nezastaví na začátku editačního pole, ale přemístí se do předchozího textového pole, ve kterém také vymazává znaky. Při označování textu s použitím klávesy **Shift** je kurzor umístěn oproti standardu ve Windows na začátku označeného textu, takže zrušení označení například klávesou **Šipka vpravo** umístí kurzor jinam, než by uživatel čekal. Rovněž nefunguje přeskakování slov klávesovou zkratkou **Ctrl**+**Šipka vpravo** nebo **Ctrl**+**Šipka vlevo** – kurzor se přemis-

ťuje na okolní ovládací prvky.

Z těchto důvodů jsme do souboru skriptů implementovali řadu skriptů a funkcí pro nejběžnější editaci textu v poli. Klávesám pro práci s textem (kurzorové klávesy, shift a control s kurzorovými klávesami) jsou přiřazeny skripty, které volají funkce pro úpravu textu v editačním poli. Základní myšlenkou je uložení pozice kurzoru v aktivním textovém poli do globální proměnné `iAktPos` a jeho obsahu do `sAktObsah`. Stiskne-li uživatel v aplikaci klávesu pro přesun kurzoru o jednu pozici, je skriptem do aplikace klávesa propuštěna. Při stisku kláves pro přesun o více písmen nebo pohyb po slovech je do aplikace skriptem propuštěna klávesová zkratka pro přesun o jedno místo tolikrát, kolik činí rozdíl mezi aktivní a požadovanou pozicí kurzoru. Při označování textu je navíc ve skriptu nastaven příznak `iAktOznaceno`, podle kterého se určuje, kolik písmen je označeno a směr, ve kterém byl text označován.

K docílení správné funkce pomocných skriptů pro práci s textem je nezbytné aktualizovat globální proměnné s informacemi o textovém poli. Toho jsme dosáhli funkcí události `KeyPressedEvent` ve skriptu Bílých stránek, která po každém stisku klávesy znovu načítá text, pokud je aktivním prvkem editační pole.

Použité řešení neustálého načítání textu po stisku kláves je častým zdrojem mrzutostí v ozvučeném skriptu. Pokud uživatel píše pomalu a čeká na oznámení každého nového písmene, funguje celá posloupnost funkcí správně. Při velmi rychlém zadávání textu jsou nastavené prodlevy a časové konstanty skriptu pro načítání kopírovaného textu pole ze schránky delší, než prodleva mezi psanými písmeny. To má za následek chybné načítání textu do proměnné skriptu, nevymazávání pomocné značky z kopírovaného textu, která pak zůstává v editačním poli aplikace apod.

Faktem však zůstává, že oproti stavu úplného mlčení a chaosu při přesouvání kurzoru během editace textu, je implementovaný způsob stále velmi dobře použitelný.

3.4.2 Tlačítko

K ovládání tlačítek není ve skriptu potřeba žádné speciální zacházení. Pro každé tlačítko, pro nějž jsme požadovali přímou aktivaci klávesovou zkratkou, jsme vytvořili skript `KlikNa...`. Tečky v názvu skriptu představují název tlačítka. Tomuto skriptu je přiřazena aktivační klávesová zkratka tlačítka.

Je-li tlačítko aktivním ovládacím prvkem ve skriptu a uživatel stiskl **Enter** nebo **Mezerník**, zajistí skripty těmto klávesám přiřazené přesun kurzoru do rámce určující polohu tlačítka a kliknutí na tlačítko voláním funkce `LeftMouseButton`.

Protože stisknutí tlačítka může vyvolávat v aplikaci nějaké změny, na které je třeba reagovat, implementovali jsme funkci `ButtonAkcePoStisku`. Jejím parametrem je identifikátor stisknutého tlačítka. Funkci volají skripty pro mezerník nebo enter, eventuelně skript `KlikNa...` po každém stisknutí tlačítka. Podle předaného parametru se provede požadovaný kód, reagující na změnu v aplikaci. Jedná se obvykle o inicializaci proměnných skriptu jiných ovládacích prvků (po stisku tlačítka *Vymazat*) nebo oznámení o stavu a výsledku vyhledávání (po stisku *Hledat*).

3.4.3 Rozbalovací seznam

Rozbalovací seznam v aplikaci Bílé stránky umožňuje procházení svých položek klávesami **Šipka nahoru** a **Šipka dolů**, nebo vyhledání položek podle počátečních písmen pouze v rozbaleném stavu.

Proto je každé klávese pro pohyb v rozbalovacím seznamu přiřazen skript, který se stiskem klávesy aktivuje. Skript volá funkci `ListBoxChangeItem`, která seznam nejdříve rozbalí a provede propuštění klávesy do aplikace. Tím se přejde na další prvek seznamu. Funkce následně seznam sbalí a z editačního pole seznamu načte a oznámí novou hodnotu. Rozbalení a sbalení je realizováno ve skriptu kliknutím na rámec ohraničující tlačítko vedle editačního pole seznamu funkcí `ListBoxDropdown`.

Kvůli častému načítání textu skriptem ze seznamu je vhodné zadávat v aplikaci klávesy pro pohyb pomalu a další stisknout až ve chvíli, kdy je oznámena nová položka. Pro urychlení práce s rozbalovacím seznamem by bylo možné poznamenat si všechny hodnoty seznamu do souboru zpráv, pamatovat si v proměnné skriptu aktivní položku a po přechodu na jinou oznámit její hodnotu pouze zprávou. Toto řešení ale v aplikaci Bílé stránky není použitelné, neboť po vyhledání záznamů neplatí, že položky v seznamu jsou vždy stejné.

3.4.4 Grafický text

Oznámení o průběhu a výsledku vyhledávání je v aplikaci zobrazeno ve formě textu s různě barevným podkladem. JAWS tento text přečíst neumí a tak

jsme do souboru skriptů přidali funkci, která využívá rozdílné barvy pozadí textu v oznámení k určení zobrazovaného stavu hledání. Kódy barev pozadí jsou uloženy na začátku souboru skriptů v řetězcových konstantách, reprezentující složky RGB:

Příklad 3.4.2 Deklarace konstant barev - fragment souboru *VFP6R.JSS*

```
cColNotFound = "255000000", ;červená barva pozadí "Nenalezeno"
cColFound = "0000128128",   ;tyrkysová barva pozadí "Vybráno"
cColFinding = "255255000",  ;žlutá barva pozadí "Hledám..."
```

Po stisku tlačítka *Hledat* zobrazí aplikace vedle tlačítka text „Hledám…“ na žlutém podkladu. Skript funkcí `ButtonAkcePoStisku`, spuštěné po stisku tlačítka *Hledat*, volá funkci `CekaniNaVysledek`. Ta v cyklu načítá barvu z místa, kde se nachází podklad textu. Dokud je žlutý, oznamuje skript uživateli hlasovým výstupem „Hledám“ v intervalech určených konstantou `cFindDelay`. Jakmile text zmizí a barva se změní na původní šedou, vrací funkce hodnotu signalizující konec hledání. Poté je funkcí `OznamitVysledek` načtena barva pozadí textu v místě, kde aplikace zobrazuje oznámení o výsledku hledání. Tím je text „Vybráno“ s tyrkysovým podkladem nebo „Nenalezeno“ s červeným podkladem.

Uvedené řešení funguje velmi dobře, avšak opět nevhodným navržením aplikace je občas problematické. Bílé stránky po začátku hledání nezobrazí text „Hledám…“ ihned, nýbrž až ve chvíli, kdy je připravena mechanika CD-ROM s diskem dat aplikace. Vyhledávání tak ještě nezačalo, ale skript načte šedou barvu v místě, kde se má zobrazit informace o průběhu. Z toho učiní chybný závěr, že vyhledávání již skončilo. Pozdržení běhu skriptu, než se objeví text *Hledám…*, zajišťuje časová konstanta. Tu však není vhodné nastavit tak, aby pozdržela běh skriptu do doby, než se stihne roztočit CD v mechanice. Při velké hodnotě časové konstanty by bylo vyhledávání zbytečně zpomalováno, protože by CD mohlo být dávno připraveno pro čtení, například při opakovaném vyhledávání stejných dat. Navíc doba pro roztočení CD v mechanice a zahájení načítání potřebných dat se jistě bude na různých počítačích lišit.

3.4.5 Počet vybraných záznamů

Počet vybraných záznamů je vidomému uživateli v aplikaci zobrazen ve stejnojmenném textovém poli, které JAWS opět přečíst neumí. Jedná se však

o důležitou informaci - zjištění počtu nalezených záznamů je jediným použitelným způsobem, jak oznámit skriptu počet řádků tabulky vybraných záznamů (3.4.6).

Způsob načtení grafického textu z aplikace do proměnné skriptu, v tomto případě čísla složeného z cifer 0..9, je podobný rozeznávání textu v OCR programech: načte se barva pixelů na předem určených pozicích v rámci ohraničujícím výskyt jednoho znaku a podle kombinace vybarvených nebo nevybarvených pixelů se určí zobrazený znak. Využili jsme toho, že font číslic je neproporcionální a tak je možné načítat postupně jedno číslo za druhým. Výhodou této metody je malý počet pixelů, jejichž barevnou hodnotu je třeba načíst.

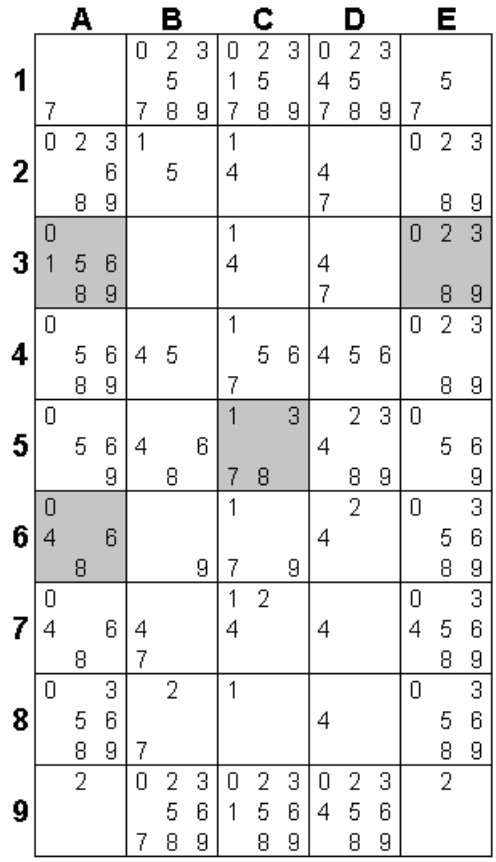
Do čtvercové sítě 5×9 (A1..E9), která představovala prostor pro umístění všech znaků jedné číslice, jsme pro každou cifru vyznačili pixely, jež při svém zobrazení vyplňuje (obrázek 3.3). Pro určení deseti hodnot potřebujeme minimálně čtyři dvoustavové hodnoty - tedy sektory, které svou hodnotou vybarveno/nevybarveno přesně určí zobrazené číslo. Po chvíli hledání jsme nalezli potřebné souřadnice - E3, C5, A3 a A6.

V souboru skriptů načítá funkce `PrecistCislo` (příklad 3.4.3) od pozice předané svým parametrem barvu pixelů z okna aplikace na pozicích číslice, odpovídacích vybraným sektorům. Je-li pixel vybarven požadovanou barvou, zde černou, je do pomocné proměnné `iOut` s počáteční hodnotou nula přičteno číslo 1. Poté je proveden bitový posun vlevo násobením dvěmi. Po přečtení všech čtyř pozic tak získáme číslo v rozmezí 0..15. To je v podmíněných skocích vyhodnoceno podle tabulky analýzy čtení znaků a funkce vrací hodnotu cifry, která byla načítána.

Funkce `VratitPocetZaznamu` provádí načítání třech cifer umístěných za sebou trojím volání funkce `PrecistCislo`, které předává souřadnice levého horního rohu místa ohraničující jednotlivé znaky cifer. Metodou Hornerova schématu z nich sestaví jedno celé číslo. Funkce `PrecistCislo` vrací hodnotu nula i pro prázdnou hodnotu všech pixelů, a tak je možné bez obav číst znaky i z místa, kde zobrazené nejsou. Tímto způsobem je vráceno jedno až trojciferné číslo představující počet vybraných záznamů.

Použité řešení je sice časově náročné na analýzu a implementaci, je však velmi rychlé a spolehlivé při běhu skriptu. Může být vodítkem pro rozpoznávání dalších neproporcionálních textů, složených i z více znaků.

Obrázek 3.3: Analýza čtení znaků



3.4.6 Tabulka

Pro správnou funkci emulovaného uživatelského rozhraní pro práci s tabulkou nalezených záznamů je nezbytné znát počet řádků tabulky po vyhledání dat. Pro jeho zjištění připadaly v úvahu dvě metody – postupný průchod tabulkou a počítání záznamů, nebo přečtení počtu nalezených záznamů z textového pole *Počet vybraných záznamů*. Procházení všech řádků tabulky skriptem je možné pouze propouštěním kurzorových kláves do aplikace, po kterém musí následovat načtení textu buňky do proměnné skriptu. Podle té se určí konec tabulky a tak i počet řádků – je-li načtena stejná hodnota znamená to, že po klávese **Šipka dolů** nedošlo k přesunu aktivní buňky v aplikaci o řádek níže a nacházíme tak na konci tabulky. Tento přístup

Příklad 3.4.3 Funkce `PrecistCislo` ze souboru *VFP6R.JSS*

```
int Function PrecistCislo(int iX, int iY, string sBarva)
var int iOut
  let iOut = 0
  if ColorToRGBString (GetColorAtPoint (iX+4, iY+2)) == sBarva
    then let iOut = iOut + 1  ;E3
  endif
  let iOut = iOut * 2
  if ColorToRGBString (GetColorAtPoint (iX+2, iY+4)) == sBarva
    then let iOut = iOut + 1  ;C5
  endif
  let iOut = iOut * 2
  if ColorToRGBString (GetColorAtPoint (iX, iY+2)) == sBarva
    then let iOut = iOut + 1  ;A3
  endif
  let iOut = iOut * 2
  if ColorToRGBString (GetColorAtPoint (iX, iY+5)) == sBarva
    then let iOut = iOut + 1  ;A6
  endif
  if iOut == 14 || iOut == 15 then return 8  ;1110 nebo 1111
  elif iOut == 12 || iOut == 13 then return 3;1100 nebo 1101
  elif iOut == 11 then return 0              ;1011
  elif iOut == 10 then return 9              ;1010
  elif iOut == 8 || iOut == 9 then return 2  ;1000 nebo 1001
  elif iOut == 6 || iOut == 7 then return 1  ;0110 nebo 0111
  elif iOut == 4 || iOut == 5 then return 7  ;0100 nebo 0101
  elif iOut == 3 then return 6               ;0011
  elif iOut == 2 then return 5               ;0010
  elif iOut == 1 then return 4               ;0001
  ;OCR selhalo, uživatel pravděpodobně zadal špatné souřadnice
  ;nebo na místě číslice nic není
  elif iOut == 0 then return 0               ;0000
  endif
EndFunction
```

je pomalý a značně nespolehlivý, protože ne vždy je text buňky načten a také počet nalezených záznamů může být velký. Proto jsme zvolili metodu „OCR“ popsanou v části 3.4.5.

Tabulka pro zobrazení nalezených záznamů obsahuje zaškrtávací políčka, která se nachází v řádku u nalezených záznamů v prvním sloupci. Jejich stav (zaškrtnuto / nezaškrtnuto) JAWS přečíst neumí a tak je nezbytné si jej pamatovat. Za tímto účelem je po vyhledání záznamů v aplikaci do

řetězcové proměnné skriptu `sTblVyberOrgCbx` zapsána posloupnost znaků `O|` v délce počtu řádků tabulky. Stiskne-li uživatel klávesu **Mezerník** při aktivním ovládacím prvku tabulka a umístěném kurzoru v prvním sloupci, je změněn v proměnné skriptu znak `O` na pozici aktivního řádku na `X`. Ten značí zaškrtnutí políčka. Tento formát řetězce jsme použili pro následné jednoduché využití funkce `StringSegment` pro dotaz na stav tlačítka podle čísla řádku, na kterém se nachází.

Problém tohoto řešení se objeví při seřazení řádků tabulky podle jiného sloupce po kliknutí na jeho záhlaví. Znaky v proměnné `sTblVyberOrgCbx` totiž nejsou s řádky svázány jinak než absolutní pozicí a ta se po seřazení řádků podle jiného klíče změní. Po krátké rozvaze jsme však od řešení této skutečnosti upustili - aplikace vykazuje značné nedostatky při řazení tabulky (viz 3.7) a tak by správná funkčnost stejně zajištěna nebyla.

Do souboru skriptů jsme implementovali funkce a skripty pro všechny běžné operace pro práci s tabulkou – přesun mezi buňkami, přesun na první a poslední buňku sloupce nebo řádku, čtení aktuálního řádku a sloupce, řazení tabulky podle aktivního sloupce a mnohé další.

Kvůli absenci horizontálního posuvníku u tabulky a proměnlivé vzdálenosti posledního řádku tabulky od jejího spodního okraje je pro správnou funkci skriptu nezbytné vždy při opuštění tohoto ovládacího prvku tabulku „zarovnat“ – tedy přesunout se kurzorem do prvního sloupce prvního řádku. To zajišťuje skript při opuštění tabulky. Bez zmíněného zarovnání by tabulka zůstala tak, jak ji uživatel naposledy opustil, a při novém přístupu by oznamování čísel řádků a názvů sloupců počítané od levého horního okraje tabulky nesouhlasilo se skutečnou polohou aktivní buňky v tabulce.

3.5 Klávesové zkratky

Součástí dobrého návrhu ozvučované aplikace je i rozvržení klávesových zkratek. V případě zkratek Windows nebo JAWS by se nemělo lišit od běžného nastavení, na které jsou uživatelé zvyklí - aktivace tlačítek, přepínání mezi ovládacími prvky, vyvolání nápovědy k ovládacím prvkům apod.

Pro rychlé ovládání aplikace je doporučujeme přiřadit nejčastěji používaným ovládacím prvkům klávesové zkratky, které způsobí přístup na prvek, případně jeho aktivaci. Pro Bílé stránky jsme zvolili zkratky pro vkládání údajů - umístění kurzoru do editačního pole *Organizace*, kliknutí na tlačítko *Hledat*, kliknutí na tlačítko *Vymazat* a několik dalších. Jejich přehled je možné zobrazit standardní klávesovou zkratkou JAWS pro vyvolání ná-

povědy klávesových zkratek aplikace – **Insert**+**H**. Pokud se s nimi uživatel seznámí, velmi si usnadní práci. Nemusí totiž přeskakovat klávesou **Tab** spousty ovládacích prvků, které využívat nechce, ale zahájí rovnou požadovanou akci.

Vhodně zvolené klávesové zkratky jsou také polovinou úspěchu. Není třeba je umisťovat blízko sebe, je lepší volit snadno zapamatovatelné, například podle prvních písmen: **Ctrl**+**Shift**+**H** pro *Hledat*, **Ctrl**+**Shift**+**V** pro *Vymazat* apod.

3.6 Omezení skriptu, problémy při ozvučování a jejich řešení

3.6.1 Umístění rámců

Rámce ohraničující ovládací prvky by měly být vztaženy k oknům, na nichž prvky leží. Aktivační podmínkou rámce se tak stává titulek nějakého okna aplikace. Při ozvučování Bílých stránek jsme narazili na chybu programu JAWS verze 5.0, který ačkoli správně načte titulek okna *Telefonní seznam organizací* nebo *Bytový telefonní seznam*, neumí za běhu tyto nadpisy použít pro aktivaci rámců na oknech.

Proto jsme polohu rámců definovali vzhledem k hlavnímu oknu aplikace s titulkem *Bílé stránky verze 4.00*. Tím však přicházíme o možnost přesouvat okna pro vyhledávání v telefonních seznamech - poloha rámce definovaná k aplikačnímu oknu se totiž s přesunutím nezmění. Na toto omezení je uživatel upozorněn v úvodním textu k aplikaci.

3.6.2 Pozdržování běhu skriptu a časové konstanty

Skript Bílých stránek obsahuje řadu míst, kde je vyžadováno pozdržení běhu skriptu do doby, než aplikace zareaguje na nějakou akci – typicky stisk klávesy vyvolaný v nějakém skriptu funkcí `TypeKey`. Použití funkcí `Pause` a `Delay` je problematické, na což upozorňuje již přehled základů skriptování pro JAWS [BS]: doba, za kterou je aplikace schopná zareagovat, velmi závisí na rychlosti počítače, na němž běží.

Řešením je zavedení „časových konstant“ – celočíselných konstant, jejichž deklarace se nachází na začátku souboru skriptů, a které určují důležité prodlevy v aplikaci (příklad 3.6.1). Pro správnou funkci skriptu po přenesení

na jiný počítač je vhodné vyzkoušet chování aplikace a časové konstanty v případě potřeby pozměnit.

Příklad 3.6.1 Deklarace časových konstant - fragment souboru *VFP6R.JSS*

```
const
  cAppSwitchTime = 6, ;čas v ms, doba na jakou se plánují
                      ;funkce, které se mají vykonat po
                      ;přepnutí do aplikace BS
  cFindDelay = 10,    ;čas v ms, časový interval mezi
                      ;průběžným informování o výsledku
                      ;hledání
```

3.6.3 Reakce na chybová hlášení

Skript nedokáže zachytit chybové hlášení o hodnotě číselníku *Min. počet znaků* a *Max pro výběr* mimo povolený rozsah. Toto hlášení se zobrazuje formou grafického textu v horní části okna aplikace po zadání chybné hodnoty. Bylo by možné po každé zadané číslici do číselníku zjišťovat ve skriptu funkcí `GetColorAtPoint` barvu části okna v místě, kde se zobrazuje chybové hlášení a podle jeho šedé barvy upozornit uživatele na chybný rozsah, nebo rovnou opravit hodnotu v číselníku. Toto řešení však přináší značné zdržení a nutnost kontrolovat každý stisk kláves s čísly.

Pro zamezení zobrazování chybového hlášení aplikací umožní skript uživateli zvyšovat nebo snižovat hodnotu číselníku pouze kurzorovými klávesami. Tím je z vlastnosti ovládacího prvku zabráněno překroční povolené hodnoty.

3.6.4 Focus ovládacích prvků

Má-li ovládací prvek „focus“, znamená to, že je aktivní. Tlačítko je zvýrazněno orámováním, v editačním poli je kurzor pro psaní, název karty je orámován apod. Korektně naprogramovaná aplikace umožňuje přístup na všechny své ovládací prvky pomocí klávesy **Tab**. Pokud tomu tak není, je nutné klávese **Tab** přiřadit skript, jenž se stará o správné přepínání mezi ovládacími prvky.

Zde je však schován nemalý problém: focus je v aplikaci umístěn na nějakém ovládacím prvku (například editační pole), ale nevidomému uživateli se oznámí, že je aktivní jiný – například tlačítko. Vše je v pořádku, pokud

uživatel stiskne jen **Mezerník** nebo **Enter** pro aktivaci tlačítka. Těmto klávesám jsou přiřazené skripty, které v případě, že je ve skriptu označen typ aktivního prvku tlačítko, přesunou kurzor do rámce ohraničující tlačítko a provedou kliknutí. Začne-li však uživatel psát nějaký text nebo se jen omylem „přepíše“, jsou jednotlivé klávesy skriptem propouštěny do aplikace a text se objevuje v editačním s „fyzickým“ focusem v aplikaci.

Tomuto chování se dá zabránit filtrováním kláves v události `KeyPressedEvent`. Jedná se však o metodu značně pomalou a tak jsme navrhli díky chybě v aplikaci Bílé stránky jiný způsob.

Vypozorovali jsme, že po několikerém stlačení klávesy **Tab** v aplikaci přechází focus po ovládacích prvcích, načež najednou „zmizí“ a není možné jej vyvolat ani zpětným přesunem klávesami **Shift**+**Tab**. To má výhodu v tom, že stisky kláves, psaní textu, nebo jiná podobná akce klávesnice nemá na ostatní prvky aplikace žádný vliv.

Toto slepé umístění focusu je implementováno ve funkci `UklidFocus` (příklad 3.6.2).

Příklad 3.6.2 Funkce `UklidFocus` ze souboru *VFP6R.JSS*

```
Void Function UklidFocus ()
var string sOkno
    ;načtení názvu reálného okna
  let sOkno = GetWindowName (GetRealWindow (GetFocus ()))
  if sOkno == msgCapOrg then
    KlikNaOvl (cLstTextKrajOrg)
  elif sOkno == msgCapByt then
    KlikNaOvl (cLstTextKrajByt)
  else
    return
  endif
  TypeKey ("SHIFT+TAB")
EndFunction
```

3.6.5 Nápověda aplikace

Nápovědu v aplikaci Bílé stránky neumí JAWS přečíst. I změna barvy pozadí nebo pokusy o čtení vysvíceného textu s jinou barvou pozadí vyzněly naprázdno.

V tomto případě bylo nejjednodušší nápovědu přepsat ke konkrétním ovládacím prvkům ve formě kontextové nápovědy, zobrazované skriptem

`ScreenSensitiveHelp`. Tento skript aktivuje uživatel v aplikaci klávesovou zkratkou **Insert+F1** při spuštěném programu JAWS. Na tuto skutečnost upozorňujeme uživatele v uvítacím textu aplikace.

3.7 Chyby v aplikaci Bílé stránky

Při důkladném zkoušení všech funkcí a vlastností aplikace, nezbytném pro správný návrh ozvučovacího skriptu, jsme narazili na řadu chyb. Některé byly lehčího rázu a způsobovaly „pouze“ horší ovladatelnost aplikace, jiné byly kritické a měly za následek její pád. Nastává ale otázka, co je chybou aplikace a co její vlastností. Je-li víceřádkové editační pole určeno pouze pro čtení a přitom do něj lze vkládat text, není to ničemu na škodu, ale nevidomého uživatele to může zmást.

Je důležité poznamenat, že zatímco vidící uživatel programu snadno vyhodnotí chybu a často bez problémů najde jiné řešení, nevidomý nemá jinou možnost než spolehnout se na hlasový výstup programu JAWS. Oznámí-li JAWS, že je aktivní tlačítko a ono je vinou chyby v aplikaci neaktivní, jen těžko lze podobné chování podchytit nebo ošetřit. Je tedy zřejmé, že stabilita a pečlivé odladění aplikace je pro její zpřístupnění nevidomému uživateli stěžejní. Jakékoli chyby totiž komplikují její ozvučení a v konečné fázi mohou uživatele od používání programu také zcela odradit.

Jedinou obranou proti známým chybám v aplikaci je jejich zmínění ve skriptu v kontextové nápovědě k ovládacímu prvku, který chybu způsobuje.

Následující výčet zahrnuje nejzávažnější chyby v aplikaci Bílé stránky:

- Ihned po startu programu jsou na kartě *Podrobný náhled* zpřístupněna tlačítka pro výběr dalšího a posledního záznamu. Jejich stisknutím dojde k chybě v aplikaci.

- Po uzavření všech oken pro vyhledávání a následném otevření libovolného nového jsou na kartě *Podrobný náhled* nepřístupná všechna tlačítka a to i při správně vyhledaném nenulovém počtu záznamů. Není tak možný průchod mezi podrobnými informacemi.

- Při stisknutí tlačítka *Storno* na dialogu pro uložení souboru ve formátu Microsoft Excel 5.0, vyvolaném z dialogu *Tisk*, se jako jméno souboru pro uložení vybere **.xls*. Potvrzením tohoto názvu dojde k chybě aplikace.

- Řazení záznamů v tabulce na kartě *Výběr* není funkční u sloupce *Telefon*, řazením podle jiných sloupců se ztrácejí z tabulky záznamy, nefunguje řazení podle české abecedy.

3.8 Zhodnocení

Práci na skriptu pro Bílé stránky 2004-2005 je možné charakterizovat jako neustálé vymýšlení a zkoušení postupů pro překonávání potíží při skriptování. Zjištění startu aplikace, načítání textu z editačního pole, oznamování vyhledávání nebo práce s tabulkou byly těmi nejtěžšími. Po naprogramování hlavní kostry skriptu nastala fáze dlouhého ladění a zkoušení funkčnosti při nejrůznější práci s aplikací. Přidávání zbylých ovládacích prvků bylo naopak jen rutinním opisováním kusů kódu a vepisováním identifikátorů do správných funkcí a skriptů.

Výsledný skript není zdaleka ideálním řešením jak z hlediska funkčnosti, tak i naprogramování. Práce na něm však byla vynikající školou skriptovacího jazyka JAWS a přípravou pro navržení šablony pro automatizované generování skriptů. Skript obsahuje několik slabých míst a nevhodných obratů, které však byly v šabloně JAWS framework odstraněny nebo upraveny.

Kapitola 4

Analýza

V této části seznamujeme čtenáře s důvody, které nás vedly k řešením navržených ve specifikaci a ke krokům použitých v implementaci JAWS framework. Doporučujeme seznámit se před dalším čtením se specifikací (příloha A) a programátorskou dokumentací (na přiloženém CD).

4.1 Architektura aplikace

Architekturu aplikace jsme zvolili na základě požadavků na JAWS framework kladených:

- minimální zásah do souboru *default.jss*
- načítání dat z JAWS do JAWS framework bez jejich přepisování
- práce parseru se stejnými daty nad více soubory
- oddělená šablona skriptu, kterou je možné jednoduše nahradit jinou

Načítání dat z JAWS probíhá prostřednictvím COM serveru. Důvodem pro výměnu dat tímto způsobem jsou oddělené paměťové prostory aplikací JAWS a JAWS framework, omezený počet funkcí ve skriptovacím jazyce JAWS určených pro přenos dat a naopak dobrá podpora objektů. JAWS obsahuje funkci `SendMessage`, ta však umí přenášet jako parametr pouze čísla typu integer. Sdílení řetězců ukládáním do souboru a opětovným načítáním je nevyhovující kvůli problémům se současným přístupem a nastavením cest.

Aby byl co nejjednodušší a přehledný, slouží COM server výhradně k výměně dat. Neimplementovali jsme do něj žádné dodatečné funkce závislé

na JAWS framework, naopak obsahuje grafické rozhraní pro snazší ladění. Proto je možné jej snadno využít v budoucnu v dalších projektech.

Možnou variantou zjišťování informací z aplikace JAWS bylo také jejich zadávání přímo uživatelem. Tím by se ale značně prodloužila doba vkládání dat do JAWS framework, nevyhnuli bychom se překlepům a celkově by komfort práce s aplikací značně utrpěl. Uživatel by totiž musel znát řadu klávesových zkratek JAWS pro zjišťování třídy okna, práci s manažerem klávesnice, nutili bychom jej psát si vlastní skripty pro určování pozice myši vzhledem k aplikačnímu oknu apod.

Díky COM objektu spočívá ve většině případů použití serveru pouze v přidání názvu skriptu pomocí `include` do *default.jss*.

4.2 Šablona

Šablona vychází ze skriptu pro ozvučení Bílých stránek 2004-2005. Podle hotového skriptu, který obsluhuje všechny typy ovládacích prvků a který je odladěný, je možné získat přehled o částech potřebných, specifických pro aplikaci apod.

Vytváření šablony postupným průchodem hotového skriptu a dopisováním značek se jevilo jako vyhovující postup, jak nezapomenout na žádné části kódu a zároveň vyřadit nepotřebný. Zavedené značky parseru mají řadu výhod:

- začínají středníkem a tak jsou ve skriptovacím jazyce JAWS považovány za komentář
- mají blokovou strukturu - otevírací a uzavírací značku
- začínají vždy na novém řádku a tak strukturují kód do přehledných částí
- uzavírací značka je shodná pro všechny typy bloků
- značka proměnné je v kódu dobře čitelná

Struktura čtyř souborů, obsahující logicky provázaná data (skript, zprávy, rámce, klávesové zkratky), přináší výhodu ve snadném editování potřebné části šablony. Bylo by možné tyto soubory v šabloně sloučit do jednoho, ale poté by přibyly další speciální značky určující konec souboru

a jediný soubor šablony by byl příliš dlouhý a nepřehledný. Navržené řešení využívá možnosti použít stejné značky parseru a tak při generování je nad jedněmi zadanými daty o ovládacích prvcích provedeno parsování čtyř souborů.

Otázka použitých značek je nedílně spjata s volbou parseru, který šablonu bude načítat. Vytváření šablony se ukázalo být velice časově náročné a pozdější převod do některého z možných formátů jako XML by posunul termín vyhotovení projektu za únosnou mez. Tagy XML jsme nepoužívali ihned od začátku - v té době bylo důležité vytvořit si jednoduché značky pro označování bloků kódu a částí skriptu a zaměřit se pouze na správný a korektní návrh funkce šablony.

4.3 Parser

Pro generování skriptu z šablony připadala v úvahu možnost využití některého z nástrojů pro parsování XML (například XSLT), ale zvolili jsme vlastní implementaci parseru. Nástroje pro parsování textu, které jsme hledali k použití, totiž obvykle byly příliš robustní a také jejich zakomponování do programu a nutnost dodatečného stahování a přidávání do aplikace by znamenaly pro uživatele zbytečnou komplikaci.

Celý projekt JAWS framework byl koncipován jako do značné míry otevřený - má poskytovat dostatek prostoru pro pozdější implementaci dalších funkcí a rozšíření. Protože není možné plně postihnout všechny záludnosti aplikací určených pro oskriptování nebo jejich specifické chování, je implementace vlastního parseru dobrým krokem pro pozdější doplnění programu.

Navrhovaný parser vychází čistě z potřeb vytvořené šablony a tak neobsahuje žádné funkce navíc. Základní vkládání bloků a jednoduché cykly stačí ke splnění všech požadavků na parser kladených:

- pouze jeden průchod šablonou
- přímý postup kupředu a nevracení se zpět do souboru
- možnost sekvenčního zpracování více souborů nad stejnými zadanými daty

První dva body jsme volili pro maximální urychlení načítání šablony, poslední požadavek umožňuje rozdělit šablonu do čtyř logicky provázaných souborů.

4.4 Storage

Jelikož mají ovládací prvky řadu kritérií, podle kterých mohou být požadována jejich kompletní data, navrhli jsme úložiště zadaných dat s ohledem na nejčastěji používané parametry - například řazení ovládacích prvků podle jejich typu umožňuje velmi rychlé vyhledávání. Šablony STL jsme využili kvůli algoritmům pro vyhledávání, ochranu přístupu a další již naimplementované vlastnosti.

K tomuto způsobu uchovávání dat jsme přistoupili z důvodu velmi snadné rozšiřitelnosti - je snadné doplnit do třídy `cStorage` další specializované metody pro výběr nebo vyhledávání dat podle libovolných kritérií.

4.5 COM server

Analýza typu COM objektu, který bude nutné implementovat, zabrala nejvíce času v celém průběhu práce na projektu JAWS framework. Problematika COM je značně obsáhlá a často není možné obejít některé z pasáží při jejím studování.

4.5.1 Rozhraní

Potřebná rozhraní, ke kterým jsme dospěli v průběhu analýzy požadovaných funkcí COM objektu, jsou `IUnknown` (nezbytné pro funkci COM), `IDispatch` (*automation* rozhraní), typová knihovna, a konečně hlavní rozhraní `IShare`. To slouží k ukládání a načítání dat serveru.

Rozhodli jsme se pouze pro jedno rozhraní pro práci s daty COM serveru - počet metod a vlastností v něm exponovaných není tak velký, aby bylo nutné je dále dělit do menších částí.

4.5.2 Přístup ke komponentě

COM server je typu *out-of-process* server, je *exe* komponentou. Ta na rozdíl od *in-process* komponenty má vlastní paměťový prostor a tak umožňuje přístup klientům z různých programů. Rovněž přeložení do *exe* přináší výhodu v jednoduché registraci a možnosti detailního ladění v grafickém i konzolovém okně serveru (pokud je aktivováno). Samostatně běžící objekt je taky jediným, který je možné registrovat v *ROT* (*running object table*), jejíž využití jsme předpokládali.

Použili jsme standardní marshalování, protože typy přenášených dat jsou pouze základního typu a nepředpokládáme vzdálené připojení ke komponentě na jiném počítači.

Za threadový model jsme zvolili jediný, který je u *out-of-process* komponent přípustný - MTA (*multithreaded appartement*). Instance komponenty totiž běží v procesu realizovaném samotným serverem a nemusí tak koordinovat požadovaný model s nabízeným ([Ka]).

4.5.3 Použité technologie

Server využívá technologie *automation*, protože jej funkce `CreateObject` skriptovacího jazyka JAWS vyžaduje. Usnadňuje to také ladění COM serveru v makrojazycích, které informace z *automation* komponent umí načíst - například Microsoft Visual Basic v prostředí Microsoft Excel. Abychom usnadnili práci při použití serveru v prostředí C++, ve kterém je naprogramováno grafické uživatelské rozhraní, implementovali jsme *automation* pomocí *duálního* rozhraní. Funkce `CreateObject` volaná ze skriptu JAWS tak může využívat rozhraní `IDispatch`, zatímco v C++ je jednodušší použití rozhraní `IUnknown`.

4.5.4 Zamítnutá řešení

Během analýzy chování COM serveru jsme uvažovali oznamovat klientovi nová data pomocí technologie *connection points*. Implementace komponenty *connectable object* se však ukázala být zbytečně složitou a proto jsme se uchýlili k větší zodpovědnosti klienta, který data ze serveru vybírá. Ten si sám zjišťuje, zda se nová data na serveru nacházejí.

Původní myšlenkou navázání spojení mezi JAWS a JAWS framework bylo vytvoření COM serveru aplikací JAWS framework a potom volání funkce `GetObject` ve skriptu JAWS. Tato funkce vyhledává v *ROT* běžící instanci objektu a připojuje se k ní. Popsané řešení se však ukázalo jako nevyhovující a nepružné - je závislé na pořadí spouštěných programů a také realizace je značně složitá. Implementovali jsme vlastní *moniker*, kterým jsme vytvořený objekt zaregistrovali v *ROT*. Nepodařilo se však JAWS „donutit“, aby se k objektu připojil - volání funkce `GetObject` končilo vždy neúspěšně. Proto jsme se rozhodli implementovat server jako *singleton*. Ve skriptu se pak použije funkce `CreateObject`, která buď vytvoří novou instanci serveru, nebo se připojí k již běžící.

4.6 Ukládání dat

Formát ukládaných dat zadaných v grafickém uživatelském rozhraní nemusí být složitý, ani zbytečně univerzální. Proto jsme uvažovali textový soubor se strukturou, která půjde velmi snadno jedním průchodem souboru načíst. Není nutné využívat speciálních tagů, rozhodli jsme se pro jednoduché hlavičky ve stylu *ini* souborů. Zvolili jsme jej pro přehlednost a zejména snadnou implementaci: v hlavičce je uvedeno, co se má načítat a dále pracuje vstupní operátor, který načte všechna následující data až do prázdné řádky. Stejně jednoduše je možné ukládání dat do souboru. Data jsou tvořena dvojicí „klíč=hodnota“ na jedné řádce.

Souboru s uloženými daty jsme přiřadili vlastní příponu *jfc* - JAWS framework configuration. Není uvedena v běžných databázích přípon souborů na internetu a tak nepředpokládáme kolizi s jinými programy, které by příponu využívaly.

Textový formát souboru jsme požadovali pro možnost dodatečné úpravy v textovém editoru. Pro pozdější případné rozšíření JAWS framework se nabízí možnost implementovat jednoduchý editor *jfc* souborů, který umožní bez zásahů do uživatelského grafického rozhraní aplikace měnit a upravovat uložená data.

V průběhu analýzy ukládaných dat jsme narazili na skutečnost, že uživatel může chtít jako hodnotu proměnné uložit i víceřádkový text. To odporuje námi zavedené syntaxi souboru a proto jsme přidali požadavek na změnu dvojce znaků pro zalomení řádky v hodnotě proměnné za dvojici ESCAPE znaků. Ty je možné do souboru vložit v textovém editoru kombinací kláves **Alt**+**27**.

4.7 Optimalizace

Pro generátor skriptů JAWS framework jsme kromě již zmíněných postupů pro urychlení vyhledávání dat v úložišti žádné další optimalizace nenavrhovali. Pro počet ovládacích prvků v rozsahu podobném Bílým stránkám je rychlost generování skriptů ze šablony přiměřená. Uvažované optimalizace by mohly spočívat v ukládání nejčastěji používaných dat v cyklech se stejnými parametry do „cache“.

Kapitola 5

Skript generovaný aplikací JAWS framework

Výstup JAWS framework jsme testovali na aplikaci Bílé stránky 2004-2005. Zadávání ovládacích prvků Bílých stránek do JAWS framework a následné generování skriptu bylo dobrou cestou pro nalezení a odstranění chyb v JAWS framework i šabloně skriptů.

V této kapitole představujeme vygenerovaný skript aplikace a porovnáváme jej se skriptem původním, „ručně“ psaným, z kapitoly 3. Aby byl zachován maximální komfort práce s Bílými stránkami, bylo nezbytné do generovaného souboru skriptů dopsat funkce a skripty zajišťující ozvučení prvků specifických pro tuto aplikaci. Přidané části jsou popsány v 5.3.

Všechny soubory vygenerovaného skriptu naleznete na přiloženém CD v adresáři `Skripty pro Bílé stránky\Generované\`. Soubory dodělávaného skriptu se nalézají ve složce `Skripty pro Bílé stránky\Generované finální\`.

5.1 Struktura skriptu

Šablona generátoru skriptů vychází z původního skriptu pro Bílé stránky, doznala však četných změn a vylepšení.

5.1.1 Identifikátor ovládacích prvků

Identifikátorem ovládacích prvků ve skriptu není již celočíselná hodnota, nýbrž řetězec složený ze čtyř částí, oddělených tečkou. Jednotlivé části vyja-

dřují typ, název, umístění a pořadové číslo prvku. Tato změna má kladný vliv na čitelnost výsledného kódu, neboť je ihned z identifikátoru patrno, o který ovládací prvek se jedná. Zjištění typu nebo umístění prvku je realizováno funkcí `StringSegment`, která z identifikátoru snadno vybere požadovanou část.

Dalším přínosem a důvodem k zavedení tohoto označení je snadný způsob generování názvu proměnných a dalších identifikátorů k jednotlivým ovládacím prvkům. Popis principu práce generátoru je možné nalézt v programátorské dokumentaci JAWS framework.

5.1.2 Názvy proměnných

V šabloně jsme sjednotili formát názvů proměnných a identifikátorů, které spolu souvisí. Ukážeme jej na příkladu editačního pole *Organizace* na okně *Telefonní stanice organizací*:

"iid.Edb.Organizace.Org." – identifikátor ovládacího prvku. `Edb` je typ, názvem je `Organizace` a `Org` je název okna.

msgEdbOrganizaceOrg – název zprávy s textem „Organizace“ – popis ovládacího prvku

msgHlpEdbOrganizaceOrgL – název zprávy s dlouhou nápovědou

msgHlpEdbOrganizaceOrgS – název zprávy s krátkou nápovědou

"frmEdbOrganizaceOrg" – název rámce ohraničující editační pole

Zavedením jednotného označení identifikátorů, zpráv a rámců jsme výrazně zlepšili čitelnost kódu skriptu.

5.1.3 Start aplikace

JAWS framework není určen pouze pro „oskriptování“ aplikace Bílé stránky a tak dříve použitý způsob ke zjištění startu aplikace bylo nutné nahradit jiným, univerzálnějším.

Nově je start ozvučené aplikace indikován ve skriptu nulou v globální proměnné `iAppPoprve` – standardní hodnotou po startu aplikace. Po zobrazení úvodního textu nebo vykonání jiných úkolů po startu naprogramovaných ve

skriptu je hodnota změněna na `1` a do `hAppWindow` je uložen handle hlavního okna aplikace.

Použitím události `WindowDestroyedEvent` je handle každého zavíraného okna aplikace porovnán s handle hlavního okna a pokud souhlasí, je ještě před ukončením běhu skriptu proměnná `iAppPoprve` nastavena zpět na nulu. Tím jsme obešli uchovávání hodnoty globální proměnné i po ukončení běhu skriptu.

5.2 Novinky v ovládání aplikace

Zásadní a důležitou změnou v emulovaném ovládání aplikace oproti původnímu skriptu je nový způsob práce s editačním polem. Po vstupu do tohoto ovládacího prvku je skriptem volána nová funkce `NacistTextEdb`. Ta načte obsah editačního pole z aplikace stejným způsobem jako v původním skriptu, ale tento řetězec zobrazí pomocí funkce `InputBox` v dialogu aplikace JAWS. Jedná se o jednoduchý dialog s víceřádkovým editačním polem, které je ozvučeno, a i editace textu probíhá zcela podle běžných zvyklostí. Nevidomý uživatel vůbec nepozná, že je otevřen nový dialog – JAWS oznámí text dialogu stejný jako je popisek editačního pole a typ je stejný (viz obrázek 5.1).

Titulek dialogu je určen konstantou `cTitulekEditacnihoDialogu` a má hodnotu řetězce složeného ze třech mezer. Podle tohoto titulku se jiných funkcích skriptu určuje, zdali není otevřen dialog pro editaci textu, protože pro další práci s aplikací je třeba jej uzavřít. Na dialogová tlačítka *OK* a *Storno* uživatel nemá přístup.

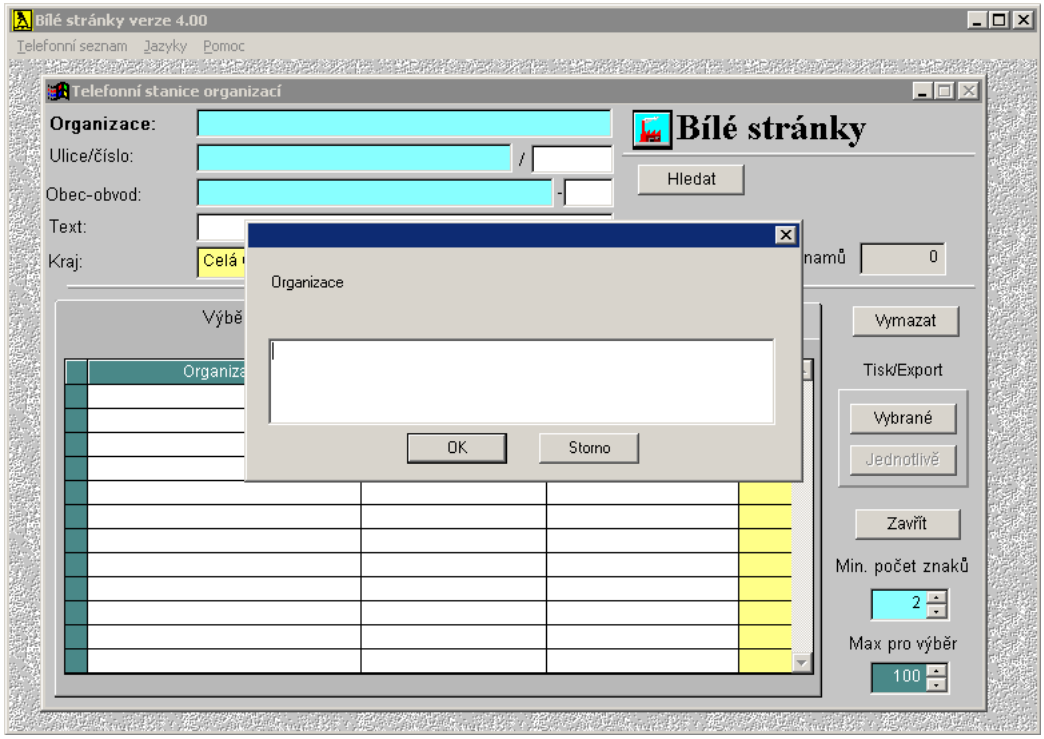
Stiskne-li uživatel klávesu **Enter**, uzavře skript dialog a text uživatelem vložený vypíše do odpovídajícího editačního pole v aplikaci. Poté skript přejde na další ovládací prvek. Po stisku kláves **Escape** dojde opět k uzavření dialogu, ale zadávaný text se ignoruje a v aplikaci zůstává v editačním poli původní hodnota. Následuje opět přechod na další ovládací prvek.

Klávesám **Tab** a **Shift+Tab** je ve skriptu v události `KeyPressedEvent` přiřazen kód pro odsouhlasení editačního dialogu, pokud je v době stisku klávesy zobrazen. Podobný kód, avšak pro zamítnutí dialogu, je ve všech skriptech `KlikNa...` nebo `VstupDo...`.

Celou funkci `NacistTextEdb` můžete vidět na příkladě 5.2.1

Vylepšená editace textu značně zpříjemnila práci s aplikací. Uživatel již nemusí „trpělivě čekat“ mezi jednotlivými stisky kláves při zadávání textu, ale může psát jak je zvyklý. Použité řešení si vynutilo zákaz přepínání mezi

Obrázek 5.1: Zadávání textu do editačního pole *Organizace*



Příklad 5.2.1 Funkce `NacistTextEdb` ze souboru *VFP6R.JSS*

```
void Function NacistTextEdb ()
var string retezec
    ;načtení obsahu editačního pole
  let retezec = VratObsahEdb ()
  if (InputBox(VratPopis (sAktID),   ;popis editačního pole
               cTitulekEditacnihoDialogu, ;titulek dialogu
               retezec)) then        ;editovaný text
      ;editační dialog ukončen OK
    KlikNaOvl (sAktID)     ;návrat do editboxu
    TypeKey ("CTRL+A")     ;označení textu
    if retezec != "" then  ;neprázdný editační dialog
      TypeString (retezec) ;výpis načtené hodnoty
    else
        ;prázdno - vymazání původního textu
      TypeKey("Delete")
    endif
  endif
EndFunction
```

aplikacemi klávesami **Alt**+**Tab** a **Alt**+**Shift**+**Tab** v průběhu otevřeného editačního dialogu – po návratu do aplikace Bílé stránky se totiž dialog nepřenesl do popředí a uživatel nemohl dále aplikaci používat.

5.3 Dopisované části skriptu

Abychom zachovali stejnou míru funkčnosti a pohodlí práce s Bílými stránkami jako v původním skriptu, museli jsme některé části skriptu dopsat. Jedná se obvykle o kód pro ozvučení nestandardních ovládacích prvků nebo informací oznamovaných aplikací Bílé stránky grafickým textem.

Vkládané funkce a skripty do generovaného skriptu je možné nalézt na konci skriptovacího souboru. Podrobný seznam dopisovaného kódu zjistíte nejlépe porovnáním obsahů obou souborů *VFP6R.JSS* – generovaného a výsledného s doplněnými změnami.

5.3.1 Tabulka

JAWS framework neumožňuje oskriptovat ovládací prvek typu tabulka, protože je poměrně problematický na ovládání a určení pozice. Může obsahovat

řadu omezení, která není možné souhrnně postihnout, nebo některá i předvídat. Navíc aplikace Bílé stránky obsahuje v tabulce pro zobrazení vybraných záznamů v prvním sloupci zaškrtávací políčka.

Do generovaného skriptu jsme proto přidali funkce a skripty pro práci s tabulkou, včetně klávesových zkratek v souboru *.jkm*. Tyto funkce a skripty bylo možné z původního skriptu přímo zkopírovat, upravili je pouze do souladu s nově zavedeným formátem názvů identifikátorů.

Do již existujících funkcí pro jednoduchý pohyb kurzorovými klávesami (`SayPriorLine`, `SayNextLine`, `JAWSHome`, `JAWSEnd` a další) jsme přidali větev podmíněného příkazu, která je aktivní případě typu tabulka. Pro přístup do tabulky jsme přidali kód i do skriptů `Tab` a `ShiftTab`.

5.3.2 Stav vyhledávání a počet vybraných záznamů

Pro oznámení o průběhu a výsledku vyhledávání jsme přidali do vygenerované ale prázdné funkce `ButtonAkcePoStisku` kód pro akci po stisku tlačítek *Hledat* na obou hlavních oknech aplikace – volání přidaných funkcí `CekaniNaVysledek` a `OznamitVysledek`.

5.3.3 Focus aplikace

Protože JAWS framework generuje prázdnou funkci `UklidFocus`, doplnili jsme do ní obdobný kód jako v případě původního skriptu.

5.4 Omezení skriptu

Přestože se spouštěcí soubor aplikace jmenuje *eltes.exe*, název souboru skriptů pro Bílé stránky je *VFP6R.jss*. Programům vytvořených v prostředí Microsoft Visual Fox Pro přiřadí JAWS jméno skriptu podle dynamické knihovny Fox Pro s názvem *vfp6r.dll*. Zde může dojít ke konfliktu: pokud by JAWS obsahoval soubor skriptů stejného názvu (tedy například pro jinou aplikaci naprogramovanou také v prostření Fox Pro), je nutné pro ozvučení Bílých stránek přepsat vygenerovanými skripty soubory existující.

5.5 Zhodnocení

Zadání všech údajů o ovládacích prvcích a vygenerování skriptu pro Bílé stránky pomocí JAWS framework je otázkou několika desítek minut. Přizpůsobení vygenerovaného skriptu konkrétní aplikaci je úkolem programátora a trváme-li na kvalitním ozvučení aplikace, není bohužel možné tento krok obejít.

Nový skript je oproti původnímu stabilnější při zadávání textu do editačních polí. Celkově si zachovává nezměněnou míru funkčnosti, což bylo jedním z požadavků na výsledný skript.

Kapitola 6

Shrnutí

V této práci jsme představili program JAWS framework pro ozvučování nestandardních aplikací, který umožňuje generování skriptů z dat zadaných jednoduchým způsobem vidícím uživatelem.

Šablonu, ze které se skripty generují, jsme vytvořili na základě skušeností s ozvučením vybrané nestandardní aplikace. Pro ní jsme napsali složitý skript, který kompletně emuluje uživatelské rozhraní.

JAWS framework jsme testovali na téže nestandardní aplikaci. Nové ozvučení pomocí JAWS framework doznalo významných vylepšení a zvýšení komfortu práce s aplikací.

Použité obraty při skriptování mohou být programátorům inspirací a vodítkem k řešení problémů při ozvučování nestandardních aplikací. Stejně tak i samotná implementace JAWS framework obsahuje zajímavá řešení a zejména poskytuje dostatek prostoru pro další rozšíření.

Literatura

[Ka] Kačmář D.: *Programujeme v COM a COM+*. Computer Press, Praha, 2000, pp. 230.

[BS] *Basics of Scripting Manual*.
`http://www.freedomscientific.com/fs_support/documentation/JAWS_Scripting_Docs/Basics_of_Scripting.exe`

[Ed] Eddon G., Eddon H.: *Inside distributed COM*. Microsoft Press, Redmond, 1998.

[FS] Freedom Scientific Developer Network: *Functions and Scripting Reference Manuals*.
`ftp://ftp.freedomscientific.com/users/hj/private/WebFiles/training/ScriptFunction/FSDN.exe`

[Go] Gould Kenneth A.: *Everything You Always Wanted to Know About Writing JAWS Scripts, But Didn't Know Whom to Ask*, 2000.

[JS] JAWS for Windows Script Files-Blind Programming site.
`http://www.blindprogramming.com/jfw.htm`

Příloha A

Specifikace JAWS framework

A.1 Zadání

Cílem projektu JAWS Framework je napsat program, který automatizuje psaní skriptů pro nestandardní aplikace. Bude se skládat z několika částí: generátoru skriptů s uživatelským rozhraním, COM objektu sloužícím pro přenášení dat mezi JAWS a generátorem, a doplňujících skriptů a funkcí ve skriptovacím souboru *default.jss* v aplikaci JAWS. Generátor skriptů bude ovládán vidícím uživatelem, jenž aplikaci ozvučuje, a za pomoci myši a vhodných klávesových zkratek bude přes řídící skript předávat generátoru informace o jednotlivých ovládacích prvcích. Po ukončení vkládání informací vygeneruje program do několika vzájemně logicky provázaných souborů zdrojový kód skriptu. Toto vytváření spočívá v předem připravené, univerzální kostře skriptu, která se hodí pro jakoukoli aplikaci, a následném vkládání kusů zdrojového kódu, které odpovídají realizacím ovládacích prvků.

Manipulace s ovládacími prvky bude do skriptu převedena tak, aby byla co nejuniverzálnější. Bude se tedy předpokládat co nejvíce nepřátelská aplikace, jejíž ovládací prvky nemají vlastní typ, třídu, ani identifikátor. Součástí projektu JAWS framework je i ozvučení aplikace *Bílé stránky 2004-2005*, které podobné chování vykazuje. Zkušenosti získané při „ručním“ psaní skriptu se pak promítnou do způsobu generování skriptu.

JAWS framework bude maximálně nezávislý na programu JAWS: předpokládá se pouze přidání skriptů pro ovládání COM objektu do hlavního souboru skriptů *default.jss*. Hlavní motivací je co nejméně zasahovat do programu JAWS a jeho nastavení.

Pro správnou funkci generátoru a spolehlivou spolupráci mezi řídícím

skriptem a COM objektem je třeba navrhnout:

- základní strukturu a chování JAWS framework
- podporované typy ovládacích prvků, informace potřebné pro jejich generování
- soubory, které skript vygeneruje
- datové struktury pro uchovávání uživatelem vložených informací o rozmístění ovládacích prvků

A.2 Základní ovládání JAWS framework

Program bude mít uživatelské rozhraní, ve kterém uživatel nastaví základní informace o ozvučované aplikaci (název, nápověda), pomocná nastavení a seznam typů ovládacích prvků, které chce ozvučit. Informace z programu JAWS potřebné pro určení polohy ovládacího prvku, nebo nadpisy oken a jejich třídy, získá uživatel stisknutím klávesové zkratky navolené u podpůrných skriptů JAWS framework.

Na základě uživatelem vybraného typu ovládacího prvku nabídne generátor editační pole pro zapsání typicky názvu, stručného popisu a podrobného popisu ovládacího prvku. Zároveň uživateli oznámí, jakým způsobem se určí poloha ovládacího prvku v aplikaci, tedy jaké klávesové zkratky při jaké poloze kurzoru myši vzhledem k ovládacímu prvku má uživatel stisknout, aby generátor poznal, kde se ozvučovaný prvek v aplikaci nachází. Rozlišovacím znakem pro aktivaci ovládacího prvku se stane text v záhlaví každého z vlastních oken aplikace, popřípadě třída okna nebo text v okně. Každý ovládací prvek tedy musí mít určeno okno aplikace, na kterém leží.

Pro ovládací prvek bude uživatel moci zvolit klávesovou zkratku pro přímý přístup. U editačního pole se tak po stisku klávesové zkratky přesune PC kurzor do editační oblasti apod. Pokud toto chování ovládací prvek neumožňuje, například při stejné akci by bylo tlačítko stisknuto, může na to být uživatel upozorněn v nápovědě k typu vybíraného ovládacího prvku i s popisem, jak je tato situace ošetřena.

Po navolení všech těchto údajů se vyčká na stisknutí posloupnosti klávesových zkratek pro určení polohy ovládacího prvku uživatelem, oznámí se úspěšné vložení nového ovládacího prvku do paměti generátoru skriptů a nabídne se ozvučení dalšího.

Na každý ovládací prvek v ozvučované aplikaci bude možné přistoupit klávesami **Tab** a **Shift+Tab**. Pořadí přepínání ovládacích prvků v aplikaci bude určeno sekvenčním číslováním v průběhu ozvučování jednotlivých ovládacích prvků. Uživatel bude moci změnit toto číslování a tím měnit pořadí, v jakém budou ovládací prvky v ozvučované aplikaci přístupné. Pro každý prvek bude možné určit následující a předchozí ovládací prvek.

Po určení pořadí ovládacích prvků nechá uživatel vygenerovat skript pro ozvučovanou aplikaci. Ten bude vytvářen generátorem ze šablony, umístěné v pracovním adresáři programu, do výstupního adresáře. Zvolí-li si uživatel kompilaci skriptu po skončení generování, budou výsledné skripty překopírovány do adresáře skriptů programu JAWS a zkompilovány. Výsledek kompilace uvidí uživatel v konzolovém okně. Určení adresářů potřebné při kompilaci provede uživatel při prvním generování skriptů s následnou kompilací. Tyto hodnoty se při ukončení programu uloží do registru systému Windows a při startu se opět nahrají.

V generátoru skriptů bude možné uložit a opět nahrát rozdělanou práci.

A.3 Typy ovládacích prvků

V této části specifikace jsou vyjmenovány všechny podporované typy ovládacích prvků zároveň s jejich možnostmi a potřebným nastavením.

Tlačítko (Button)

Textové údaje:

- název – text na tlačítku
- stručný popis – stručný účel tlačítka
- podrobný popis – podrobné informace o použití tlačítka

Klávesová zkratka:

- stisknutí tlačítka

Tab pořadí:

- protože není možné jednoznačně v každé aplikaci umístit focus na tlačítko, oznámí skript, že je tlačítko vybráno, jeho název a typ, ale vidící uživatel nebude pozorovat žádnou změnu v přesunu PC kurzoru

Určení polohy ovládacího prvku:

- kurzorem myši na střed tlačítka + klávesovou zkratku

Editační pole (EditBox)

Textové údaje:

- název – text typicky uváděný před editačním polem – čeho se EditBox týká
- stručný popis, podrobný popis

Klávesová zkratka:

- kurzor se přesune do editačního pole, oznámí se název, typ a obsah.

Tab pořadí:

- stejné chování jako klávesová zkratka

Určení polohy ovládacího prvku:

- kurzorem myši na levý horní roh editačního pole + klávesovou zkratku
- kurzorem myši na pravý dolní roh editačního pole + klávesovou zkratku

Číselník (Counter)

Textové údaje:

- název, stručný popis, podrobný popis

Klávesová zkratka:

- kurzor se přesune do číselníku, oznámí se název, typ a obsah. Číselník je určen pouze pro čtení

Tab pořadí:

- stejné chování jako klávesová zkratka

Určení polohy ovládacího prvku:

- kurzorem myši na levý horní roh číselníku + klávesovou zkratku
- kurzorem myši na pravý dolní roh číselníku + klávesovou zkratku
- kurzorem myši na střed tlačítka „nahoru“ + klávesovou zkratku
- kurzorem myši na střed tlačítka „dolů“ + klávesovou zkratku

Rozbalovací seznam (ListBox)

Textové údaje:

- název, stručný popis, podrobný popis

Klávesová zkratka:

- kurzor se přesune do seznamu, oznámí se název, typ a obsah

Tab pořadí:

- stejné chování jako klávesová zkratka

Určení polohy ovládacího prvku:

- kurzorem myši na levý horní roh ListBoxu + klávesovou zkratku
- kurzorem myši na pravý dolní roh ListBoxu + klávesovou zkratku
- kurzorem myši na střed tlačítka „rozbalení seznamu“ + klávesovou zkratku

Textové pole (Memo)

Textové pole slouží pro oznámení textu uživateli a je určeno pouze pro čtení. Jeho chování bude stejné jako editační pole, avšak případný text zadaný uživatelem bude ignorován.

Přepínač (RadioButton)

Pro správnou funkci přepínače je třeba definovat i ostatní přepínače, se kterými je provázán – je ve skupině. Toho bude dosaženo stejným základem v názvu ovládacího prvku, uživatel bude určovat pořadí přepínače, který v rámci skupiny ozvučuje.

Textové údaje:

- název – název přepínače; obsahuje-li skupina přepínačů název, měl by název skupiny předcházet názvu přepínače
- stručný popis, podrobný popis

Klávesová zkratka:

- ano - pro každý přepínač zvlášť

Tab pořadí:

- oznámí název skupiny přepínačů (pokud je uveden), typ, aktivní přepínač (název, počet z celku)

Určení polohy ovládacího prvku:

- kurzorem myši na střed přepínacího kolečka + klávesovou zkratku

Sada karet (TabControl)

Pro správnou funkci karet je třeba definovat i ostatní karty ve skupině. Toho bude dosaženo stejným základem v názvu ovládacího prvku, uživatel bude určovat pořadí karty, kterou v rámci skupiny ozvučuje.

Textové údaje:

- název – nápis na kartě; obsahuje-li skupina karet název, měl by název skupiny předcházet názvu karty
- stručný popis, podrobný popis

Klávesová zkratka:

- pro každou kartu zvlášť

Tab pořadí:

- oznámí se aktivní karta a typ

Určení polohy ovládacího prvku:

- kurzorem myši na levý horní roh nadpisu karty + klávesová zkratka
- kurzorem myši na pravý dolní roh nadpisu karty + klávesová zkratka
- takto pro každou kartu

A.4 Výstup generátoru skriptů

Všechny vygenerované soubory budou mít název, který JAWS přiřazuje skriptu aplikace. Přípony souborů jsou následující:

jss – vygenerovaný soubor skriptů. Obsahuje pomocné proměnné pro každý ovládací prvek (například obsah editačních pole, pozici kurzoru v něm apod.), předělané skripty pro základní práci s ovládacími prvky (`NextCharacter`, `NextLine`) a skripty pro další práci s aplikací.

jsm – popisy ovládacích prvků a nápověda. Nápověda k ovládacím prvkům bude generována ze stručného a podrobného popisu při zadávání uživatelem. Obecná nápověda pro použití ovládacího prvku bude připojena za nápovědu uživatelskou.

jkm – klávesové zkratky programu. Zde jsou uvedeny všechny klávesové zkratky, které si uživatel navolil během vkládání ovládacích prvků do generátoru, včetně těch, které obsluhují pomocné skripty, například klávesová zkratka **End** pro skript `MyJawsEnd` realizující chování klávesy **End** na jednotlivých ovládacích prvcích apod.

jff – rámce ozvučované aplikace. Ke každému ovládacímu prvku bude náležet jeden nebo více rámců, kterými bude určena jeho poloha a umístění dalších podpůrných částí ovládacího prvku, důležitých pro jeho správnou činnost (například tlačítko „rozbalení seznamu“ u rozbalovacího seznamu). Podmínkou pro aktivaci rámce bude text titulku okna shodný s oknem, na němž má ovládací prvek příslušný rámci ležet.

A.5 Uchovávání dat

Všechny datové struktury budou realizovány šablonami kontejnerů STL. Využije se tak již připravených algoritmů pro vyhledávání, třídění, ochranu rozsahu přístupu do kontejnerů apod.

Základními prvky pro uchovávání dat budou tabulky „proměnných“ a „parametrů“. Jedná se o řetězce ve formátu „klíč=hodnota“. Proměnná je místo v šabloně generovaného skriptu, které se nahradí textovou hodnotou příslušné proměnné. Parametr slouží pro ohraničení kusů kódů šablony a umožňuje tak vypouštět místa, která nejsou pro ozvučované prvky relevantní.

Úložiště dat bude obsahovat globální informace o aplikaci (název okna aplikace, jméno souboru skriptu apod.), data všech oken aplikace a data ovládacích prvků, jež na oknech leží. Data oken budou uložena v asociativních seznamech s klíčem názvu okna, každé okno bude obsahovat asociativní seznam ovládacích prvků s klíčem identifikátoru ovládacího prvku. Každý ovládací prvek bude jednoznačně určen čtveřicí typ, název, podmínka, helptext – typ bude zkratka názvu typu ovládacího prvku, podmínka je identifikátor okna, na kterém prvek leží a helptext je pořadovým číslem pro ovládací prvky s více částmi (sada karet, skupina přepínačů).

Vygenerovaný soubor **.jss* se bude skládat z několika typů skriptů a funkcí: části, které jsou naprosto nezávislé na ozvučených ovládacích prvcích a jsou nutné pro správný chod aplikace (`FocusChangedEvent`, `AutoStartEvent`); části obsluhující ovládací prvky (do těch se přidá pouze část *když je ovládací prvek typu XXX, potom něco*; a části, které popisují jednotlivé konkrétní ovládací prvky (pořadí výběru prvků, pomocné proměnné prvků, popisy prvků v **.jsm* souboru apod.)

Šablona skriptu, ze které bude výsledný skript generován, bude tvořena několika vzájemně logicky provázanými soubory (**.jss, *.jsm, *.jff, *.jkm*). V nich budou značkami, které budou odlišné od syntaxe skriptovacího jazyka JAWS, vyznačena místa, na která se má vkládat text relevantní pro ovládací prvky.

Generátor skriptu se bude skládat z parseru, který bude procházet soubory šablony skriptu a podle značek v textu vyhledávat požadované hodnoty proměnných nebo pravdivost parametrů. Šablony budou podporovat několik typů bloků – blok ohraničený hodnotou parametru, cykly přes prvky splňující hodnotu parametru a komentáře.

Pro uložení a načtení rozdělané práce v programu JAWS framework budou sloužit soubory strukturou podobné *ini* souborům. Ukládání bude realizováno zapsáním hlavních informacích o ozvučovaném programu (název, nápověda) a následným průchodem všech oken a výpisem všech doposud vložených ovládacích prvků.

A.6 Operační systém a verze software

Microsoft Windows XP Home Edition SP2, JAWS 5.0, Bílé stránky 2004-2005

A.7 Programovací jazyk a vývojové prostředí

Psaní skriptů bude probíhat ve skriptovacím jazyce JAWS (JAMAL), programování COM objektu v jazyce C++. Rozhraní JAWS framework bude implementováno v Microsoft Visual Studiu 2005 pomocí Win32 API.

Příloha B

Obsah CD-ROM

K této práci je přiložen CD-ROM s elektronickou formou bakalářské práce, zdrojovými kódy aplikace, aplikací samotnou a skripty vytvořenými v průběhu práce na projektu JAWS framework.

Obsah přiloženého CD:

Bakalářská práce.pdf Tato práce ve formátu PDF

Dokumentace\Uživatelská dokumentace.pdf Uživatelská dokumentace ve formátu PDF. Obsahuje návod k instalaci JAWS framework a rychlého průvodce aplikací.

Dokumentace\Programátorská dokumentace.pdf Programátorská dokumentace ve formátu PDF.

Instalace\JAWSframework Zkompilované soubory aplikace JAWS framework. Postup instalace a použití programu naleznete v uživatelské dokumentaci.

Instalace\JFskript.jsb Soubor skriptů pro spolupráci JAWS framework s aplikací JAWS.

Zdrojové soubory\JawsFramework.sln Soubor Visual Studio solution aplikace JAWS framework. Obsahuje projekty JawsFramework a JFcomponent.

Zdrojové soubory\JawsFramework Zdrojové soubory aplikace JAWS framework.

Zdrojové soubory\JFcomponent Zdrojové soubory COM serveru JFcomponent.

Zdrojové soubory\JFskript Zdrojové soubory podpůrného skriptu JFskript pro aplikaci JAWS framework .

Skripty pro Bílé stránky\Původní skripty Soubory programovaných skriptů pro ozvučení Bílých stránek.

Skripty pro Bílé stránky\Generované Soubory skriptů generované aplikací JAWS framework.

Skripty pro Bílé stránky\Generované\VFP.jfc Uložená data pro oskriptování Bílých stránek aplikací JAWS framework.

Skripty pro Bílé stránky\Generované finální Soubory skriptů generované aplikací JAWS framework s dopisovanými částmi specifickými pro ovládání Bílých stránek.