Dynamic Management of User Interface of Business Application System

Stjepan Vidačić
Neven Vrček
University of Zagreb
Faculty of Organization and Informatics Varaždin
stjepan.vidacic@foi.hr, neven.vrcek@foi.hr

Abstract. *Today there is a discrepancy between the available resolutions of modern screens and the ability of standard windows applications to automatically adapt their user forms to those resolutions (resizing forms). Another problem is the justified aspiration and need of individual users of client/server applications to dynamically and permanently adapt user forms of the application to their personal creativity and needs, in accordance with the available options of the PC's screen resolution.*

The mentioned problems are reflected as a flaw in a number of the existing business windows applications developed in various development systems, and they may result in pronounced dissatisfaction of users.

This paper presents a model of a solution to the mentioned problem. It is based on the addition of a separate object control (class) to each form of the client/server windows application. That class enables dynamic and permanent adaptation of every form and all of its objects to the active resolution of the PC screen in the business network, and also the recording in the data base of the form coordinates is adapted on the PC for each user of a form of the client/server application. In fact, it is a system of dynamic generation of the personal user interface of the application.

As a result, an entirely new and extremely useful option has been created for the users of standard client/server windows applications..

The model uses a modified, generally licensed object class (http://www.activewebsoftwares.com, WebIntel@Webintel.net, VFP Form Resizer Source Code V-3) that is developed for the MS Visual FoxPro 7.0 - 9.0 development system.

The implementation of the mentioned model in the MS Visual FoxPro 9.0. development system in this particular type of client/server windows applications provides a high-quality and new visual dimension, and also significantly contributes to users' satisfaction.

In view of the above, this paper is aimed at promoting the concept of the need to develop a generally closed AktiveX Controls of the class that will allow dynamic management of the user interface that may be integrated into a variety of OS Windows development systems intended for the development of client/server business applications, which would solve the above described problem.

Keywords. client/server windows application, personal user application interface, screen resolution, resizing forms class, AktiveX Controls

1 Introduction

The idea for this paper resulted emerged from author's extensive experience in the development of standard client/server business applications in the MS Visual FoxPro 6.0 - 9.0 development system over the past dozen years, the implementation of those applications in the information systems of various companies, as well as from continuous dealing with requests of the users to be provided with automatic and dynamic adaptation of the application form to the active screen resolution on the PCs in a business network.

One of the paradigms of the current level of development of the information systems and business applications is in the ever-growing possibilities of the information technology and in the existence of a large number of development systems, numerous applications developed to support those, or similar information systems, and great differences in the capabilities of those applications from the users' point of view.

The possibility of automatic adaptation of the user forms of application to the user's PC active screen resolution is just one of those pronounced differences.

This particular problem and the concept of its solution are not new. They have been present since the very beginnings of the development of the object development systems in the OS Windows environment, and the information industry has been investing great efforts in creating a basis for its solution [2, 3].

Of course, the problem does exist and it affects only the standard windows applications, whereas in the systems for development of web applications it has already been solved and it no longer exists as such.

However, the fact is that standard windows applications are still predominant within the business information systems that support business processes, and that we can still see on modern PC big screens a business application whose forms occupy only its smaller part and thus provide much less amount of information than possible, or required.

As a result, the user is frustrated, because he invested substantial funds into a new and modern screen, whose features he cannot use in his standard business application which is, from his point of view, unacceptable.

The search for possible solutions to the mentioned problem as far as the Microsoft Visual FoxPro 9.0 (VFP9.0) development system worked out by this author is concerned, resulted in obtaining the license for the developed system of ClassLibrary Resizer.vcx ('VFP Form Resizer Source Code V-3' [1]). After enlarging the object classes contained in the ClassLibrary Resizer.vcx and its implementation in the VFP9.0 development system, a new system has been created that ensures dynamic management of user forms of client/server VFP9.0 applications whose amazing features are the subject of this paper.

The system is implemented in the client/server application known as *TRENIS* [7] that was developed on the basis of the *VISTEL* [5] application, and by converting the static forms of the application into dynamic forms on the local user level. To the great satisfaction of users, it allows full exploitation of graphic possibilities of the screen in all forms of application, from the screen resolution 1024 X 768 onwards.

2 Dynamic system of user client/server interface of business application

In terms of dynamic adaptation to the screen resolution, the user interface of the application is scrutinized in this paper primarily from the point of view of the user in terms of ergonomics.

That means that user's standpoint is defined through his expectation that the forms of the existing applications relative to the entering of and browsing through data, should be adapted in a standard way to the graphic possibilities and the available space on the screen, so as to enable maximum amount of information.

Unfortunately, that standard could be met only in the classical textual format of DOS applications and it was met in the systems of web applications, whereas in the standard windows applications this problem calls for different solutions.

If we define the user interface form as an elementary and open dynamic system with feedback, then we can present its structure as an object diagram shown in Fig. 1

.

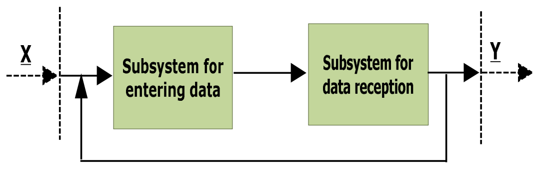

Fig. 1. Dynamic system of application's user interface

The input vector X in Fig. 1 refers to the data that users can enter into the base via data entry forms, while output vector Y refers to the data and information that users can read on the active forms, and also to the data and information that are created as a result of data processing and activation of certain functions on the active form of the application.

In that sense the application form may be considered as a system on the first level, which consists of two subsystems. The subsystem for data entering encompasses field and functions of the form intended for data entry, while the subsystem for data reception contains options of the form to receive, process and browse the database.

The feedback between the two mentioned subsystems of the application active forms is very important, as it proves the adaptability and power of the form as a complex object, which is often burdened with a large number of fields, control functions and textual descriptions of the fields and functions.

The more complex the application form is, and the more objects of various types it contains (Table 1) – the problem of its size, its objects and fonts of textual information are more pronounced and, naturally, it calls for dynamic adaptation of the form to the active screen resolution and expansion of the form over the entire screen.

Table 1. Form application objects that are dynamically adapted to the screen resolution

Type of object
'Commandbutton', 'Optionbutton', 'Pageframe', 'Container', 'Listbox', 'Grid', 'Textbox', 'Label', 'Editbox', 'Checkbox', 'Combobox', 'Spinner', 'Line', 'Shape', 'Image', 'Olecontrol', 'Oleboundcontrol', 'Page', 'Commandgroup', 'Optiongroup'

3 ClassLibrary Resizer.vcx system for dynamic adaptation of Visual FoxPro forms to the active screen resolution

The original ClassLibrary Resizer.vcx for dynamic adaptation of Visual FoxPro forms [1], was created for the standard desktop windows applications, installed on the local computer, and in its original structure it can not be implemented in the system of client/server applications. The structure of the Resizer.vcx system with a list of functions is shown in a UML diagram in Fig. 2.

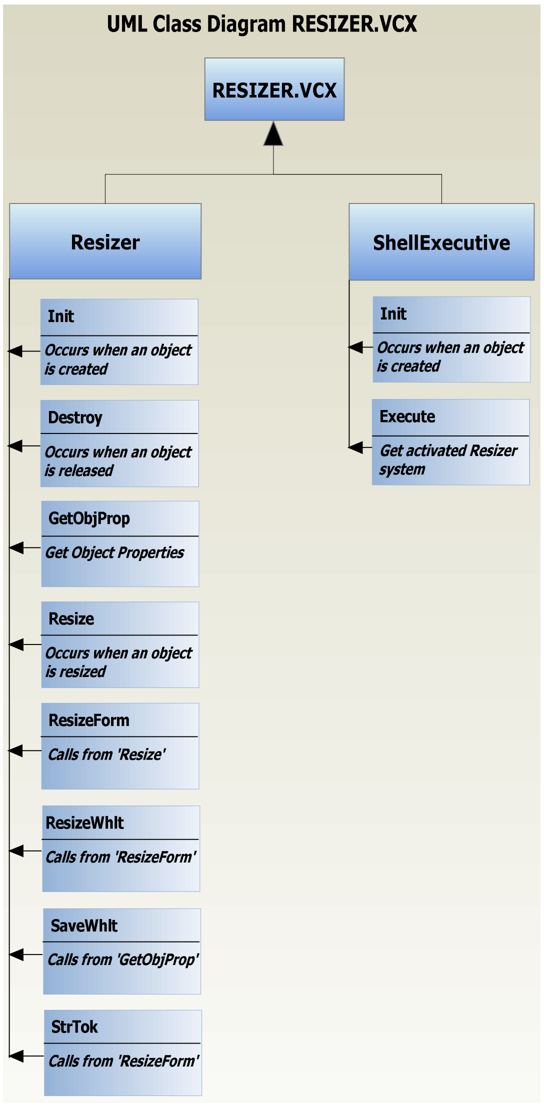

Fig. 2. UML diagram of ClassLibrary Resizer.vcx

The implementation of the standard ClassLibrary Resizer.vcx [1] within the framework of the client/server application starts from the assumption that the local dynamic management is an option, as well as occasional or permanent adaptation of selected application forms on the local user level.

In the sense of the above, the relational database model of the application needs to be enlarged with an additional relation of 'UserForms' whose attributes are described in Table 2.

Table 2. Description of attributes of 'UserForms' Relation

Attribute	**Description of attribute**	**Status of attribute Org - standard, New - new**
UserName	Application user mark	New
Cname	Name of application form	Org
Nwidth	Form width in pixels	Org
Nheight	Form height in pixels	Org
Nleft	X form coordinate in pixels	Org
Ntop	Y form coordinate in pixels	Org
UserNameCom	Name of application user	New

The standard, and by the author of this paper modified Visual FoxPro code of some functions of the 'Resizer' class given in Fig. 2, is shown in Table 3.

Table 3. Modified Visual FoxPro code of some functions of 'Resizer' class [1]

Function	**Visual FoxPro Code**
	PARAMETER lRestore
	private aResize
	IF parameter()!=0
	this.lRestore=m.lRestore
	ENDIF
	*
	PRIVATE cSelect
	if this.nOrigH=0 or this.nOrigW=0
	this.nOrigH=thisform.Height
	this.nOrigW=thisform.Width
	ENDIF
	cSelect=alias()
	*
	this.nProp=0
	this.getobjprop('thisform.objects')
	this.save
	*
Init	IF this.lRestore
(Rectified function)	IF !used('UserForms')
	SELECT 0
	USE UserForms ALIAS UserForms SHARED
	ELSE
	SELECT UserForms
	ENDIF
	CURSORSETPROP("Buffering",5,"UserFor
	ms")
	SET ORDER TO index1
	SEEK xUserForms + thisform.Name
	IF FOUND() .and. xResizeUserForms =
	.T.
	thisform.width = UserForms.nWidth
	thisform.height = UserForms.nHeight
	thisform.top = UserForms.nTop
	thisform.left =
	UserForms.nLeftthis.resize
	ENDIF
	ENDIF
	*
	private cSelect
	IF this.lrestore
	cSelect=alias()
	IF !USED('UserForms')
	SELECT 0
	USE UserForms ALIAS UserForms SHARED
	ELSE
	SELECT UserForms
	ENDIF
Destroy	CURSORSETPROP("Buffering",5,"UserFor
(Rectified function)	ms")
	SET ORDER TO INDEX1
	SEEK xUserForms + thisform.name
	IF xResizeUserForms = .T.
	IF !found()
	APPEND BLANK
	RLOCK()
	REPLACE UserForms.UserName WITH
	xUserForms,;
	UserForms.cName WITH
	thisform.name,;
	UserForms.Korisnik WITH
	xyzUserMod
	ENDIF
	RLOCK()
	REPLACE UserForms.nWidth with
	thisform.width,;
	UserForms.nHeight with
	thisform.height,;
	UserForms.nLeft with
	thisform.left,;
	UserForms.nTop with
	thisform.top
	unlock
	TABLEUPDATE(.t.)
	ENDIF
	IF !empty(m.cSelect)
	IF ALLTRIM(m.cSelect) != "USERFORMS"
	select (m.cSelect)
	ENDIF
	ENDIF
	ENDIF
	parameter cObject
	private i,cObjectI
	if !empty(cObject)
	i=1
	cObjectI=cObject+'(i)'
	do while type(cObjectI)='O'
	with &cObjectI
	do case
	case .baseclass$'Commandbutton
	Optionbutton'
	this.savewhlt()

GetObj Prop
 	
(Standard Function)
	this.aProp(this.nProp,5)=.fontsize
	case .baseclass='Pageframe'
	this.savewhlt()
	
	this.getobjprop(cObject+'('+allt(str
	(i))+')'+'.pages')
	case .baseclass='Container'
	this.savewhlt()
	
	this.getobjprop(cObject+'('+allt(str
	(i))+')'+'.objects')
	case .baseclass$'Listbox'
	this.savewhlt()
	
	this.aProp(this.nProp,6)=.columnwidt
	hs
	case .baseclass$'Grid Textbox Label
	Editbox Spinner Checkbox Combobox'
	this.savewhlt()
	
	this.aProp(this.nProp,5)=.fontsize
	case .baseclass$'Line Shape Image
	Olecontrol Oleboundcontrol '
	this.savewhlt()
	case .baseclass='Page'
	
	this.aProp(this.nProp,5)=.fontsize
	
	this.getobjprop(cObject+'('+allt(str
	(i))+')'+'.objects')
	case .baseclass$'Commandgroup
	Optiongroup'
	this.savewhlt()
	this.getobjprop(cObject+'('+allt(str
	(i))+')'+'.buttons')
	endcase
	endwith
	i=i+1
	enddo
	endif

Resize
 	
(Standard Function)
	PRIVATE I
	IF this.nOldWidth!=0 and
	this.nOldHeight!=0
	IF thisform.Width!=this.nOldWidth or
	thisform.Height!=this.nOldHeight
	this.nDifW=thisform.Width/
	this.nOrigW
	
	this.nDifH=thisform.Height/
	this.nOrigH
	IF abs(this.nDifW)>.5 or
	abs(this.nDifH)>.5
	this.nProp=0
	this.resizeform('thisform.objects')
	this.save
	thisform.refresh
	ENDIF
	ENDIF
	ENDIF

Resize Form (Rectified function)	PARAMETER cObject PRIVATE i,cObjectI,nCol,cCOlWidths,nColWidth i=1 cObjectI=cObject+'(i)' DO while type(cObjectI)='O' WITH &cObjectI DO case CASE .baseclass$'Commandbutton Optionbutton' this.resizewhlt() CASE .baseclass='Pageframe' this.resizewhlt() this.resizeform(cObject+'('+allt(str (i))+')'+'.pages') CASE .baseclass='Container' this.resizewhlt() this.resizeform(cObject+'('+allt(str (i))+')'+'.objects') CASE .baseclass$'Listbox' this.resizewhlt() cOColWidth=this.aProp(this.nProp,6) IF !empty(cOColWidth) nCol=1 cColWidths='' cCol=this.strtok(m.cOColWidth,m.nCol ,',') DO while !empty(cCol) nColWidth=max(int(val(m.cCol)*this.n DifW),0) cColWidths=m.cColWidths+allt(str(m.n ColWidth))+',' nCol=m.nCol+1 cCol=this.strtok(m.cOColWidth,m.nCol ,',') ENDDO .columnwidths=cColWidths ENDIF CASE .baseclass$'Grid Textbox Label Editbox Listbox Checkbox Combobox Spinner' this.resizewhlt() CASE .baseclass$'Line Shape Image Olecontrol Oleboundcontrol ' this.resizewhlt() CASE .baseclass='Page' IF this.lResizeFont nFontSize=max(ceiling(this.aProp(thi s.nProp,5)*this.nDifW),7) .fontsize=m.nFontSize ENDIF this.resizeform(cObject+'('+allt(str (i))+')'+'.objects') CASE .baseclass$'Commandgroup Optiongroup' this.resizewhlt() this.resizeform(cObject+'('+allt(str (i))+')'+'.buttons') ENDCASE ENDWITH i=i+1 ENDDO

The 'Resizer' class shown in Fig. 2 functions in the following algorithm:

1) Expanding the form all over the screen in active resolution, or manual positioning of the form by the user to the desired location on the screen;
2) Memorizing of the active form coordinates of the individual user in the relational fields of 'UserForms': Nwidth, Nheight, Nleft, Ntop;
3) After each subsequent activation of a certain application form on the part of the individual user, and based on previously memorized form coordinates, the 'Resizer' class dynamically adapts all fields, fonts and controls mentioned in Table 1 in line with the active screen resolution.

Activation/deactivation of classes 'Resizer' and 'ShellExecutive' on the selected form of the client/server application is obtained by using the 'drag and drop' method within the

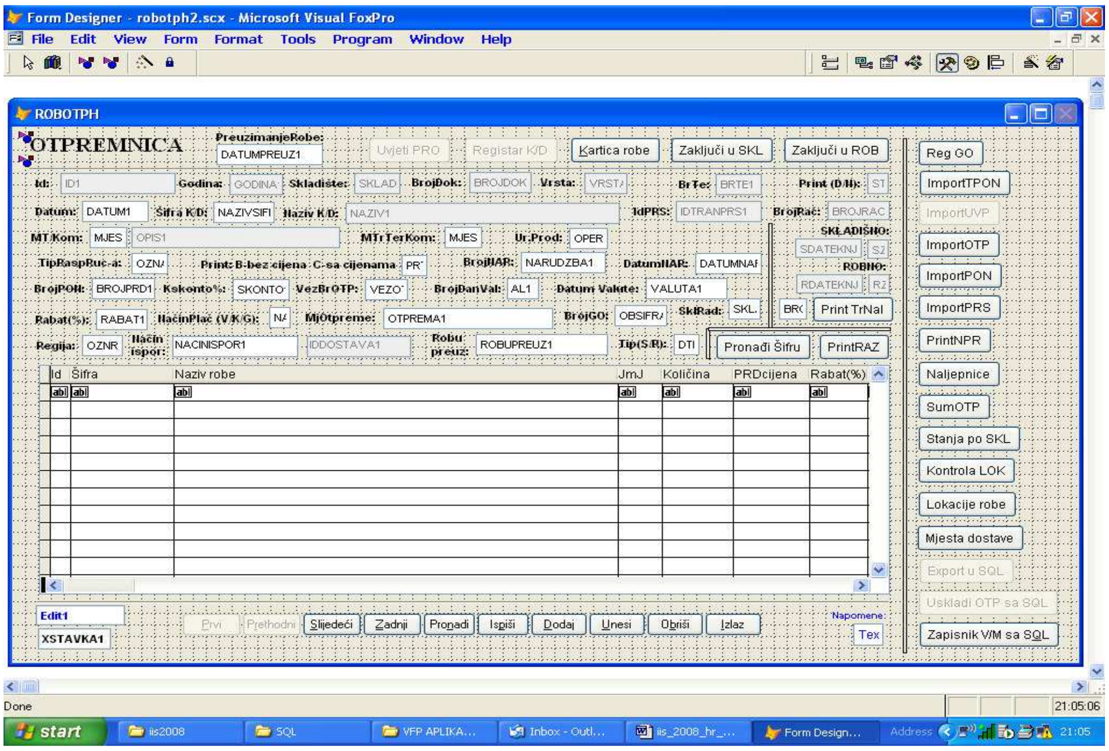

Fig. 3. Example of a form with activated 'Resizer' and 'ShellExecute' classes

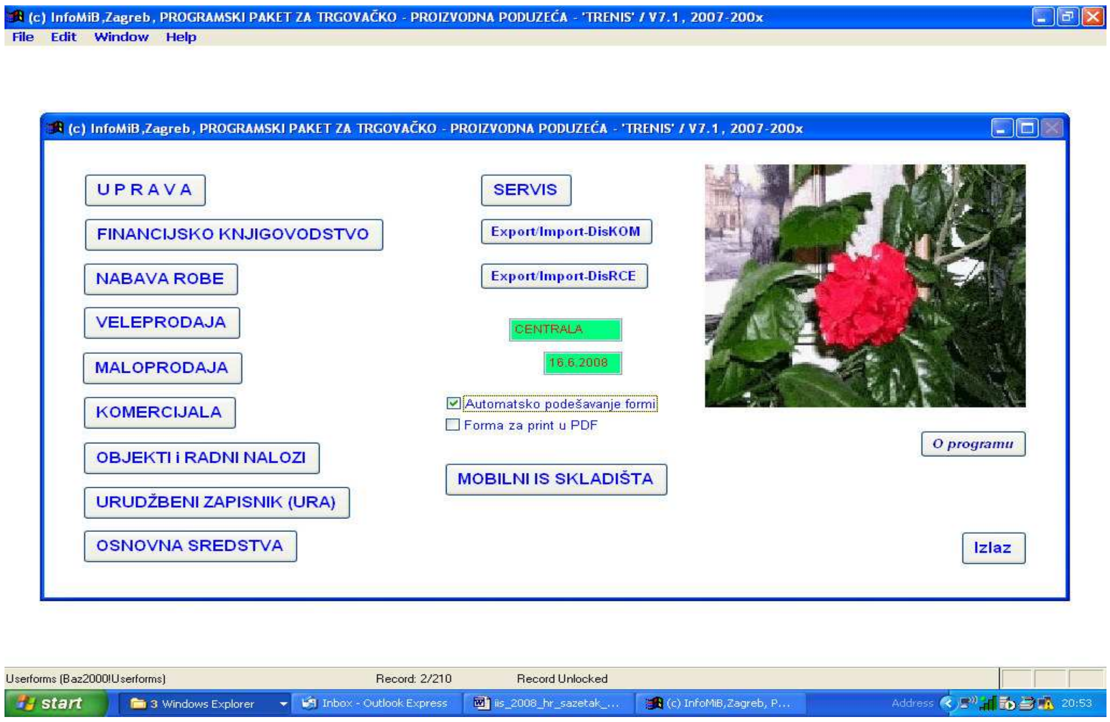

Fig. 4. Initial form of *TRENIS* application

framework of the project of the development system of the application (Fig. 3), which then requires compiling and re-installing of the application.

Fig. 3 shows the classes set up in the upper left corner of the form. They are activated by applying the 'drag' method from the standard 'ToolBars' system.

4 Some aspects of the implementation of the dynamic system of managing the user forms of the VFP client/server application

A paradox in implementing the option of dynamic adaptation of the client/server application form to the screen resolution is in that there are always users who refuse to accept new and advanced options and insist on the classical way of management of the application forms.

This implies that, apart from the option of the dynamic local customization of client/server application forms by each individual user within the business network, there must exist an option of off/on switching of the basic 'Resizer' class on the local level taking into account the needs of individual application users.

In the *TRENIS* client/server application that particular option is built in the initial form of the modular application system and is marked as 'Automatic adaptation of form', (Fig 4).

Installation of the 'Resizer' class into Visual FoxPro application of *TRENIS* has made the user interface system of that application dynamic and independent of the screen resolution, which is has been met with great satisfaction of the users. As such, it is one of the most relevant advantages in comparison with the similar business windows applications currently applied.

5 Conclusion

As has been pointed out earlier in this paper, the problem of dynamic adaptation of the forms of user interface windows applications to the active screen resolution goes back to the very beginnings of the development of object development systems in the OS Windows environment. Failure to find an efficient solution to that problem led to a discrepancy between the graphic options of modern screens and the user interface of the application forms and, consequently, to serious discontent of the users. Among other things, this too contributed to the surprise and shock of users when the process of abandonment of DOS applications and transition to windows applications began.

Given that standard desktop and client/server business applications are still prevailing in the business systems, the described problem is highly pronounced, and it requires a solution considering the pressurizing of discontent application users.

One of the possible solutions to the problem of client/server application developed within the Visual FoxPro 9.0 development system has been presented in this paper. It actually upgrades one of the general commercial solutions [1]. The upgrading of the ClassLibrary Resizer.vcx system and its successful implementation in the project Visual FoxPro of the *TRENIS* application substantiate the quality of that solution.

Given the relevant experience with the ClassLibrary Resizer.vcx, the authors of this paper deem that the future work on the problem of dynamic adaptation of the user forms of business applications should be focused on the development of the universal and closed ActiveX Control class that can be integrated in all existing systems intended for the development of standard client/server business applications, which would solve the described problem.

References

[1] http://www.activewebsoftwares.com, WebIntel@Webintel.net, VFP Form Resizer Source Code V-3.

[2] http://www.sharewareconnection.com/titles/resizer-activex-control.htm

[3] http://www.programurl.com/software/activex-control.htm

[4] Microsoft Visual FoxPro 9.0 SP2, MSDN – Help.

[5] Vidačić, S.: **Some Models of the "VISTEL" Program Used to Support the Management of the Business Processes of a Trading Company**, Proceedings of the 14th International Conference of information and Intelligent Systems - IIS'2003, september 24-26, Varaždin, 2003, str. 263-272.

[6] Vidačić, S.: **The use of Information System in The management of Business Rules**, Proceedings of the 16h International Conference of Information and Intelligent Systems - IIS'2005, september 21-23, Varaždin, 2005, str. 145-151.

[7] Vidačić, S., Brumec, S.: **Project Software package TRENIS – MOBTRENIS**, ordered by Ellabo d.o.o., Zagreb,