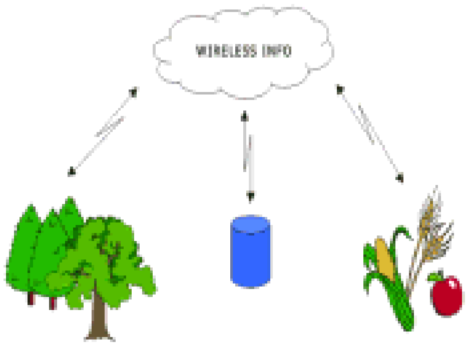


IST-2001-32595 WirelessInfo
D 14.1

Creating info-server

Public

Contractual Date of Delivery to the CEC:	**October 2002**
Actual Date of Delivery to the CEC:	**December 2002**
Author(s)	**S. Nagy, J. Berke, G. Hegedus, M. Csak, J. Busznyak, S. Kafka, S. Holy, S. Nedev, T. Krocan**
Participant(s):	**UV, HSRS, SPACECAD, DUMA**
Workpackage:	**WP14**
Est. person months:	**6,5**
Security:	**Public**
Nature:	**Report**
Version:	**1**
Total number of pages:	**20**

Abstract:
The WP will have focus on the developing of supporting system for **objective methods of landscape changes** evaluation using direct terrain controlling based on utilisation of wireless technologies in combination with another technologies. The combination of remote sensing technologies, 3D modelling, on line surveying and photogrammetry will be used. This deliverables describe implementation of basic system. The implementation of this system is partly build on experiences of another WP, but it connected this results with new tools, mainly supported work with DMT. Also new possibilities of user interfaces are tested. The aim is a quick and effective development, and that the service should be up-to-date and of good quality. An important factor is flexibility, and a simple integration of the new services. Wit was considered that there is a wide variety of wireless devices, which heterogeneity seems to last for a while. These services should be obtainable from as many devices as possible. Based on these reasons did it was chose the MS Visual Studio Net development toolkit, and its accessories. .

Keyword list:
Web Map Service, OGC specification, Minnesota University MapServer for WMS usage, Grass, Visual Studio.net, Mobile Internet Toolkit, Smart Device Extensions

1 EXECUTIVE SUMMARY

Region development and landscape conservation together with countryside development and protection of the environment are connected with the re-allotment of land. Considerable devastation of the environment and also lifestyle in the country during the communist regime is a challenge for Eastern and Central European countries to look for a way out of the current state on which farmers, ecologists, planners, sociologists, and other professions should participate. There are a few basic tasks for every state administration to deal with. The system supported this activities was implemented.
The central aim of the workpackage was to find the most effective way to obtain exact data in mapping applying wireless devices. The task of modifying a map or creating new features in a most convenient and exact way was in question. Another important point was to obtain the most effective dual mode communication with the MapServer. For this, a variety of GIS software, wireless devices, and GPS receivers were analysed and compared from the point of view of special use, considering that geographical data standards are different in our country from those in the others. As a new tools for providing of 3D analysis and remote sensing analysis, direct link between Mapserver and another Open Source solution Grass, was established.
In system is indented to complete two different solutions:

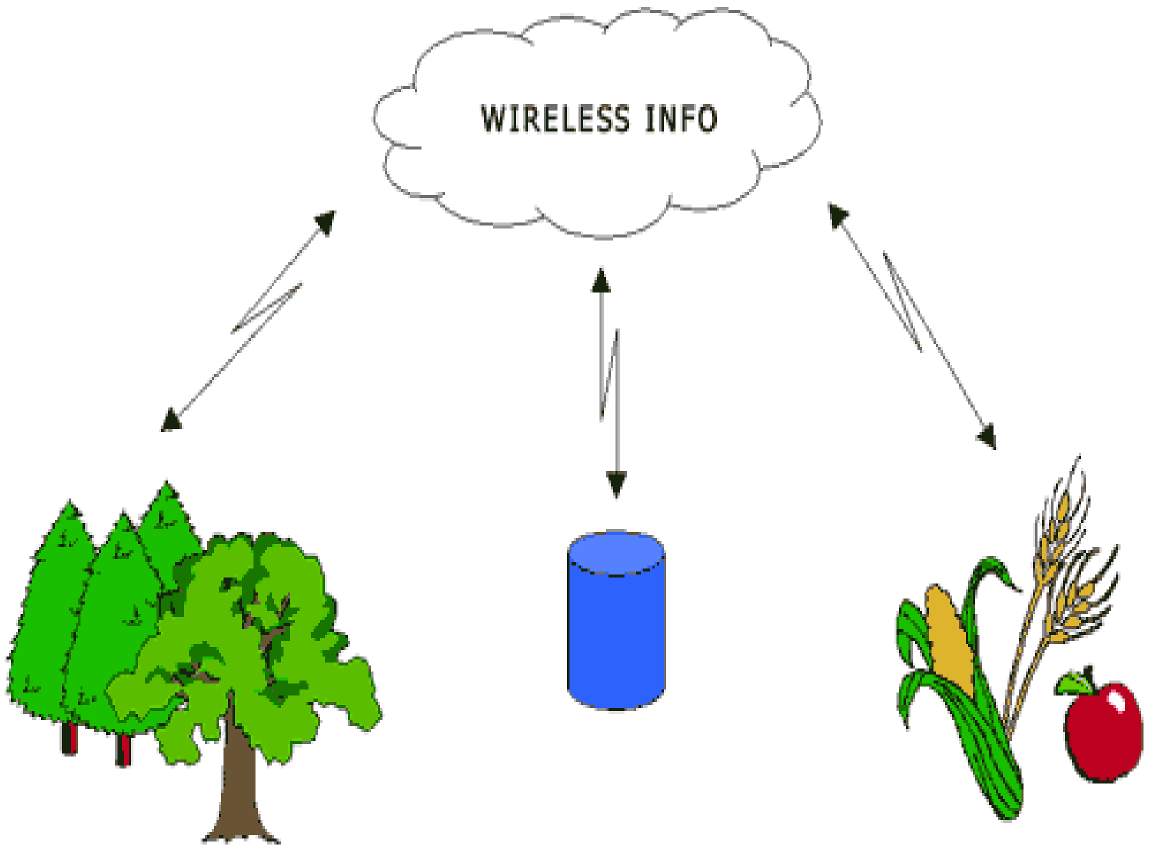
- The usage of on-line map server from a wireless device - With the help of its browser, the wireless device gets in connection with the web server, and a task is selected. The web service quires the position of the client, which is provided by the GPS receiver, attached to it. The data belonging to the coordinates and the given task are sent back to the client (eg. The client sends back the data of an object -or objects- that is/are nearest to the given coordinates). Depending on the type of the task, not only data connected to the coordinates can be provided by the server, but data ordered to the position of the client supplied by the client can also be created and fixed in the central database (eg. Results of measurements obtained in a certain area attached to coordinates are sent to the server.)
- Periodically refreshed usage of MapServer Web-Service from wireless device - As field works are usually time consuming, and the continual Internet connection of the wireless device is costly, a service for certain tasks are being prepared which refreshes the data on the server or the wireless device from time to time depending on whether the client asks for or supplies data (eg. Measurement results of certain points of the field are sent to the server together, or they are downloaded first to the wireless device). In this case, there is already need for client sided programming as well.

In both cases, the identification of the client and his authority to apply the service are handled according to the given task.
In many applications for the mobile data access, it is necessary to access parallel to more data sources. This data sources are usually distributed among more organisations and they are located among more data servers. It plays a very important role in the case of direct field measuring. Currently this situation is solved by replication of the same data in different institutions. This report describes design of our solutions based on UMN Mapserver. It offers an operative data access anywhere and anytime is necessary to establish a more operative solution. Our solution is built on the utilisation of Web Map Service (WMS - specifically an OGC Web Map Service).
According to intentions of the WirelessInfo project is its technological background based on OGC standards. This approach ensures base conditions for interoperability of used geodata services. Till this date OGC published several standards for sharing geodata over WWW. To the most developed and also adopted by wide community belongs WMS. The WMS is a Web Map Service (specifically an OGC Web Map Service). A WMS is capable of producing maps drawn into a standard image format (PNG, JPEG, etc). based on a standard set of input parameters. The resulting map can contain "transparent" pixels where there is no information and thus several independently drawn maps can be laid on top of each other to produce an overall map.

Creating model

WirelessInfo

IST-1999-21056



2 CONTENT OF REPORT

3 SOLUTION TOOLS

3.1 Mobile Internet Toolkit

Over the past few years, the world has seen an explosion of new wireless devices, such as cell phones, pagers, and personal digital assistants (PDAs), which enable users to browse Web sites at any time from any location. Developing applications for these devices is challenging for the following reasons:

- Different markup languages are necessary, including HTML for PDAs, wireless markup language (WML) for wireless application protocol (WAP) cell phones, and compact HTML (cHTML) for Japanese i-mode phones.
- Devices have different form factors. For example, devices have varying numbers of display lines, horizontal or vertical screen orientation, and color or black and white displays.
- Devices have different network connectivity, ranging from 9.6 KB cellular connections to 11 MB Wireless LANs.
- Devices have different capabilities. Some devices can display images, some can make phone calls, and some can receive notification messages.

The Microsoft Mobile Internet Toolkit addresses these challenges by isolating them from the details of wireless development. Thus, developers can quickly and easily build a single, mobile Web application that delivers appropriate markup for a wide variety of mobile devices.
The Mobile Internet Toolkit contains:

- **Mobile Web Forms Controls** that generate markup language for different devices.
- **Mobile Internet Designer** that works with the Visual Studio .NET integrated design environment (IDE) to provide a drag-and-drop mobile development environment.
- **Browser Capabilities** that are rich enough to extend ASP.NET device capabilities to mobile devices.
- **QuickStart Tutorial** with sample code.
- **Developer Documentation.**
- **Device adapter code samples.**

3.1.1 Mobile Web Forms Controls

The mobile Web Forms controls are ASP.NET server-side controls that provide user interface elements such as list, command, call, calendar, and so on. At execution time, the mobile controls generate the correct markup for the device that makes the request. As a result, you can write a mobile application once and access it from multiple devices.
Because these mobile controls are based on the ASP.NET controls, you can leverage your current desktop development skill set when creating mobile applications. You can also reuse the same business logic and data access code that you use in your desktop application. Mobile and desktop Web Forms can reside in the same Visual Studio .NET project. This makes an application faster to develop and lowers your maintenance cost.

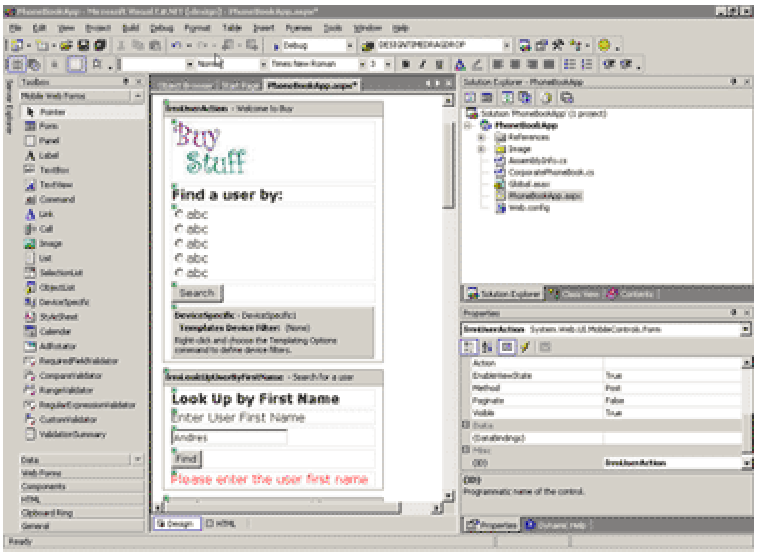
3.1.2 Mobile Internet Designer

The Mobile Internet Designer extends the Visual Studio .NET IDE to create mobile applications. After you install the Mobile Internet Designer, you can create and develop your mobile application in the same way that you would develop a Windows Forms or Web Forms application. The Mobile Internet Designer leverages the traditional Visual Studio design environment so that you can:

- Create a mobile Web project.
- Add a mobile Web Forms page to the project.
- Drag a mobile Web Forms control onto the form.
- Double-click the control to write the logic.

- Rebuild the application.
- Run the application.

The Mobile Internet Designer makes it fast and easy to build and maintain mobile Web applications. In addition, it enables today's desktop developers to quickly learn how to create mobile applications by using Visual Studio .NET.
The following illustration shows a mobile application developed in Visual Studio .NET with the Mobile Internet Designer.



3.1.3 Device Capability Mechanism

Accurate information about the display capabilities of the target device is essential for the successful rendering of mobile controls. At a minimum, mobile controls need the following information about a device:

- Markup language (HTML, WML, cHTML)
- Browser
- Number of display lines
- Cookie support
- Screen size

The Mobile Internet Toolkit adds these mobile device capabilities to the server's machine.config file (desktop ASP.NET applications use this file to maintain device and browser information).

3.1.4 Advanced Features: Extensibility

In addition to adding device characteristics, the Mobile Internet Toolkit includes device adapters for a variety of mobile devices. The device capabilities information in the machine.config file allows device adapters to optimize markup for specific devices. You can create new aggregate controls from existing mobile controls. The Mobile Internet Toolkit supports device adapters for additional devices through the device capabilities mechanism.

3.1.5 Conclusion

The Mobile Internet Toolkit provides the technology and tools to build, deploy, and maintain sophisticated mobile applications quickly. Tight integration with Visual Studio .NET ensures that developers can leverage their existing desktop skill set and produce mobile applications. Additional device support can be added using the Mobile Internet Toolkit's extensibility features.

3.2 Smart Device Extensions for Visual Studio .NET

The most capable personal digital assistants (PDAs) used today are more powerful than many of the desktop computers in use just five years ago. For this reason, many organizations are turning to smart devices as a way to improve operational efficiencies and gain a competitive advantage.
While powerful devices represent one source of opportunity, the ubiquity of the Internet represents another. How does an organization successfully exploit the Internet to integrate these devices with the organization's infrastructure? In addition, how does an organization integrate its operations with the dissimilar systems and platforms that its customers and business partners use?
XML WEB service are the key to this new level of integration. Microsoft .NET is Microsoft's platform for creating and consuming these XML Web services and Web-based applications. The new Microsoft .NET Compact Framework Technology Preview delivers this platform to smart, resource-constrained devices. The Smart Device Extensions (SDE) for Visual Studio .NET simplify the creation of these next-generation applications.
Once installed, SDE becomes directly incorporated into the integrated development environment (IDE). SDE will ship as an integral part of Visual Studio .NET. To date, most developers are unable to employ the same programming tools for mobile and other devices that they use for Microsoft Windows® and the Web. Instead, developers who create applications are typically required to use one set of tools, languages, and programming models for the desktop and another for their device applications. This added demand hinders productivity and requires costly retraining. If developers want to extend their existing desktop applications to the device, they must learn a completely new set of tools, a new set of programming interfaces and practices, and typically a new and specialized language. Visual Studio .NET solves this problem by enabling millions of developers who build desktop and server applications to use the same tools when developing applications for mobile devices. Programmers are thus able to use the same skills, techniques, visual designers, and existing code when creating sophisticated applications for devices or the desktop. Corporations benefit because they are not forced to invest in costly retraining.
SDE for Visual Studio .NET enables mobile applications to be created for the Pocket PC and future devices based on Microsoft Windows® CE .NET operating system. Each of these applications can easily consume XML Web services, which makes integration with other systems dramatically easier.
Once SDE for Visual Studio .NET is installed, developers can immediately start to create mobile applications by using either Microsoft Visual Basic® .NET or Microsoft Visual C#™ .NET. All other languages that support the .NET platform will be supported in future releases.

3.2.1 Visual Studio .NET: A Unified Development Environment

Visual Studio .NET also enables mobile-application developers to use the same unified development environment for developing mobile applications that they currently use to build desktop and server applications. Since the base templates for Pocket PC and Windows CE .NET-based applications are included in an additional "Smart Device Application" project type, developers can begin developing device applications using the same skills and practices they already use for Windows and the Web. These familiar Visual Studio .NET capabilities include:

- **Integrated development environment (IDE)**
  Increasingly, application development does not simply focus on one tier of the solution. Rather, a developer builds a solution that integrates multiple tiers: network servers, Web servers, desktops, and mobile laptops and devices. By bringing all of these development tiers into one integrated environment, Visual Studio .NET makes it easy to build connected and mobile applications.

  In the Visual Studio .NET environment, all languages share the same visual designers—which drastically reduces the time needed for training. The same forms designer, component designer, compiler, debugger, intelligent code editor, server exploration tools, and database-design tools can all be used. This new synergy between desktop and device development results in shorter development cycles and lower total cost of development.

  XML Web services
  Like desktop applications, mobile Web applications now can easily consume XML Web services. This enables mobile applications to integrate with business logic and databases on other systems and on dissimilar platforms by using open Internet protocols.

- **Windows Forms**
  Developers have used the Visual Studio Windows Forms designer to create sophisticated user interfaces for desktop applications. In Visual Studio .NET, they can use that same designer to create applications for devices. A rich subset of the controls available for desktop applications is available for the development of applications for the Pocket PC or Windows CE .NET operating systems. As with desktop application development, new controls can be created by inheriting them from existing ones. Controls can also be bound to data sources.

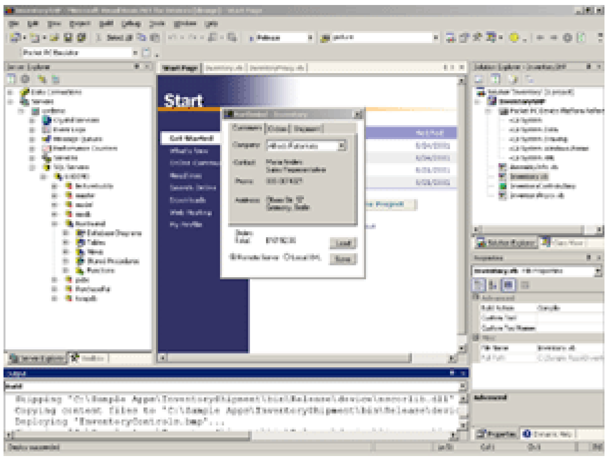
- **ADO.NET**
  The data access mechanism used to create desktop applications is also available to developers who build Pocket PC applications. Based entirely on XML, Microsoft ADO.NET enables device applications to work with local or remote data as well as integrate and share data with other platforms through XML Web services. Ideally suited for mobile devices, an ADO.NET dataset can be persisted locally and taken offline for use in a disconnected environment.

3.2.2 More Visual Studio .NET Features

In addition to offering the same unified design environment for desktop and device applications, Visual Studio .NET provides many unique and compelling features for the developer of device applications. These include:

- **Device emulation**
  Developers need not have access to a specific device before creating or testing their applications. A powerful, high fidelity device emulator is integrated within the Visual Studio .NET design environment. This emulator accurately represents the physical device by executing the exact target operating system on the developer's desktop, which runs actual operating system images in a virtual machine. Emulation operating systems for Pocket PC devices are provided. In fact, as new custom devices based on Windows CE .NET emerge, their emulators can be easily be plugged into Visual Studio .NET to speed application development for those devices.

- **Automatic deployment**
  When developers need to test their applications, they can choose to run the applications in the emulator mentioned above (using the Pocket PC emulator), or they can download the application to the connected device. When deploying to a device, Visual Studio .NET will automatically determine which files are required on the specific device and will begin the download. Once downloaded, the application is executed, and debugging can begin.



```
Skipping 'C:\Sample Apps\InventoryShipment\bin\Release\device\mscorlib.dll'
Copying content files to 'C:\Sample Apps\InventoryShipment\bin\Release\devi
Deploying 'InventoryControls.bmp'...
```

Figure 1. An application running in the emulator

Figure 2. The same application running on a device

- **Remote debugging**
  Once applications are automatically deployed to the device they can be remotely debugged from within Visual Studio .NET using the integrated debugger. The process for debugging applications running in the desktop emulator or on the connected device is identical to the process for debugging desktop applications. Developers can set breakpoints, step through code, watch and trace variables, view local variables, monitor the call stack, and more.

3.2.3 The .NET Compact Framework

Sitting underneath all mobile and device applications created with Visual Studio .NET is the .NET Compact Framework, a rich subset of the .NET Framework found on the desktop and the server. The .NET Compact Framework is geared toward developers who build applications that target devices. The .NET Compact Framework provides all of the benefits of the .NET Framework, but it is designed specifically for small, resource-constrained devices. The .NET Compact Framework greatly simplifies the process of creating and deploying applications to mobile devices while allowing the developer to take full advantage of the capabilities unique to each device.
Applications that run on top of the .NET Compact Framework are often referred to as "managed" applications. These applications are able to use a range of run-time services—including a common language runtime, memory management, and a rich set of base classes that handle security, data access, XML Web services, and more. Using Microsoft eMbedded Visual C++ can develop native or „unmanaged“ applications®.

- **CPU independence**
  Applications that target the .NET Compact Framework run independently of the CPU resident on the device. The advantage of this is that applications created for the Pocket PC will run on all Pocket PC devices without needing to be recompiled.

High-Performance compiler and JIT technology
The .NET Compact Framework uses the same high-performance Microsoft Visual C# .NET and Visual Basic .NET compilers as the desktop, ensuring maximum compatibility across servers, desktops, and devices. In addition, the heart of the .NET Compact Framework is a just-in-time (JIT) compiler, which compiles mobile applications on demand, enabling them to often run at near native-code speeds.

4 GPS POSITIONING IN MAPPING

4.1 Description of the Task

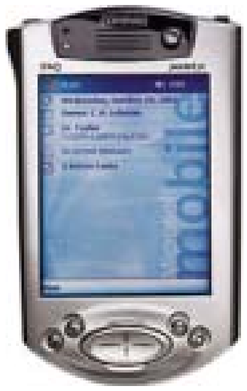
The objective of the tasks are:

- Making the coordinates of the maps of the area in question (ESRI shapefile) more exact with the help of a GPS device.
- Modification of the map, creating attributions.
- Selecting the suitable GIS software, wireless equipment, and a GPS receiver for this task and the dual mode communication with the MapServer.

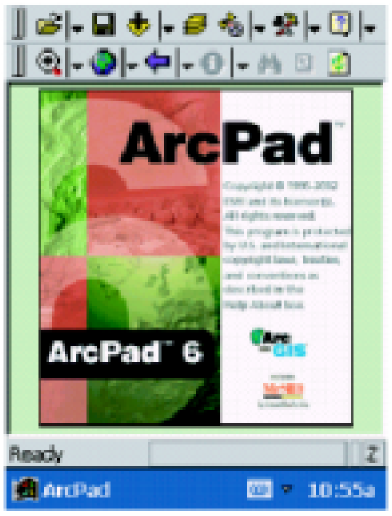
A task within the project was to make our shapefiles more exact with the help of Global Position System (GPS). For this reason, iPAQ Pocket PC H3870 Garmin GPS receiver was used with ESRI ArcPad 6 software.

iPAQ Pocket PC H3870 - Specifications

Operating System	Microsoft Pocket PC 2002
Processor	206 MHz Intel StrongARM 32-bit RISC Processor
Display Type	Color reflective thin film transistor (TFT) LCD, 64K colors
Touch Screen	Yes
Resolution	240 x 320
Pixel Pitch	.24 mm
Viewable Image Size	2.26 x 3.02 inches
RAM	64 MB
ROM	32 MB
Input Method	Handwriting recognition, soft keyboard, voice record, inking
Communications Port	Interface with USB / Serial connectivity that connects via serial or USB cable
Card Slot	SD Memory Slot, Optional expansion packs
Wireless Connectivity	Bluetooth™, Infrared port (115 Kbps)
Speaker & Microphone	Yes
Audio Out Jack	Yes (3.5 mm Stereo)
Battery	1400 mAh Lithium Polymer
Dimensions	5.3" x 3.3" x .62"
Weight	6.5 oz. including battery
Warranty	1-year limited



4.2 ArcPad 6



Usage of ArcPad 6
ESRI ArcPad software is a mobile mapping and geographic information system (GIS) technology which provides database access, mapping, GIS, and GPS integration to users out in the field via handheld and mobile devices.
Important features for our project

ArcPad supports vector and raster data in a multi-layered environment. Data can be downloaded to ArcPad with the help of TCP/IP connection. ArcPad allows you to create and edit spatial data using input from either the mouse pointer, pen, or GPS.
The measurement within the project was taken by Garmin GPS receiver which we got access to for one month to try. We used the shapefiles made of the given area. Coordinates supplied by GPS made the data more accurate. The projection of the shapefiles was very problematic, as in Hungary a special coordination format is being used (EOV). We had problems with the direct use, and the conversion of the data was also of difficulty. There is no software available at the moment, which could provide compatible data, and, if they are converted, there are also problems with the accuracy. These are the reasons why we used GCS_WGS_1984 projections in our trial surveys. The accuracy of our GPS receiver is about 10 metres, which is too much. In the future, NAVMAN GPS is going to be used to improve quality of results (guaranteed accuracy ~5m).

4.3 Navman GPS

GPS 3000 for iPAQ H3800 series handheld

Navman GPS3000 Technical Specification

Antenna Type	Quadrifiler helix (multi-directional)
Satellite Measure Used	12-channel parallel, automatic selection
Memory Card	Type I & II Compact Flash cards
Time to First Fix (TTFF)	
Hot Start	18 Seconds (typical)
Warm Start	48 Seconds (typical)
Cold Start	120 Seconds (typical)
Re-acquisition Time	25 Seconds (typical)
Accuracy of Position Fix	Horizontal: 5.0 metres, typical (95% probable)
Interface	iPAQ Option Pack Interface
Update Rate	Initial, every second. Typically every 2 seconds once fix established
Output Message	NMEA 0183 Ver: 2.20, GPGGA, GPGSA, GPRMC, and GPGSV.
Power	3.3V DC from the iPAQ's internal Lithium Polymer rechargeable battery
External Power	12.0V DC in-car adaptor
iPAQ Battery Life	1.5 - 2.2 hours when typically running with full backlight and GPS enabled

System Specification

PC Requirements

iPAQ Requirements	IBM Compatible 486 or higher
	iPAQ H3600, H3700 or H3800 series
	Dual Speed CD Rom or higher
	800Kb for application software
	MS Windows 95, 98, ME, 2000 or XP
	Typically 12Mb free for map data
	For H3600 series units: Pocket PC ROM build 1.77 or higher
	For H3700 & H3800 series units: Pocket PC 2002, all ROM builds

The Complete Solution

Windshield Mounting Bracket

Vehicle Power Cable

SmartPath City GPS Software
SmartPath Trip GPS Software

Windshield Mounting Bracket

4.4 Evaluation of the Navigation Systems

4.4.1 Evaluation of the Navigation Systems that could have been used in project

	Navigation with GPS	iPAQ compatible	Works with NMEA-0183	Price	Info
WinPilot Pro	y	y	y	$ 600	NAVMAN compatible With GPS and maps
TomTom Navigator	y	y	y	$ 430	With GPS and maps
PowerLOC Destinator	y	y	y	$ 400	With GPS and maps
TravRoute CoPilot	y	y	y	$ 350	With GPS and maps
Pocket Streets MapPoint 2002	y	y	y	$ 250	with North American Maps
AeroMap 1.23 Hungarian	y	y	y	$ 100	NAVMAN compatible With maps
Terrasync szoftver a Trimble	y	y	y	$ 500	Not NAVMAN compatible With GPS
Fugawi	y	y	y	$ 500	With GPS and maps

Out of these, the following demo versions were tried in practice: ArcPad, HGIS,

	Navigation with GPS	NAVMAN compatible	iPAQ compatible	Editing map	Works with NMEA-0183
ESRI ArcPad	y	y	y	y	y
StarPal HGIS	y	y	y	y	y

Both ArcPad and HGIS software are suitable for our aim, the tasks can be completed using either. They are both available for roughly the same price (~500$). ArcPad has more extension to Hungarian mapping characteristics, and there is customer support available, so it is easier to use in our country.
Out of the GIS/GPS software we have tried or examined, we found ESRI ArcPad the most suitable for our aims. The iPAQ H3780 hardware with Navman GPS receiver, and the accuracy of location definition with the help of map positioning algorithmisation are suitable. The dual mode communication with the MapServer needs further own software development.

5 SOLUTION OUTLINES

According to intentions of the WirelessInfo project is its technological background based on OGC standards. This approach ensures base conditions for interoperability of used geodata services. Till this date OGC published several standards for sharing geodata over WWW. To the most developed and also adopted by wide community belongs WMS.
WMS is based on transfer of georeferenced maps i.e. images or eventually drawings, which are generated on request. In some cases exist demand on transfer not maps but geofeatures. This possibility belongs to communication among geodata sources. Such type of the service is covered by another OGC specification WFS – Web Feature Service. This specification defines formal request over WWW, which returns OGC simple features in GML format. GML is an XML dialect for geographic features.
Self WFS isn't still in version 1.0, but GML is well defined. Specification GML ver. 2.1.1 is on address http://www.opengis.net/gml/02-009/GML2-11.pdf. Because of very few implementations of WFS, for purpose of Wireless info we try to develop simple web feature broker, which cove just necessary subset of WFS capabilities.

5.1 Review of WMS implementation

To the date 30.4.2002 is registred 67 implementations from 24 institutions. Majority of institutions adopt version 1.0.0 of the WMS. The Minnesota Mapserver is probably most developed open source project in Web GIS area. Project starts as an open alternative to ESRI IMS server. Initially was based on shapefile handling, but nowadays incorporating OGC standards for spatial data. Generally UMN MapServer is very advanced in adoption of the OGC standards. WMS are implemented in last versions and also cascading is supported.

5.2 Testing overview

In this stage of testing we are focused on establishing proposed structure on Intranet environment. We simulate network of WMS enabled datasources with WMS server. Such as is UMN Mapserver. This solution we consider such optimal according to several reasons:

- Mobile device gateway is already developed and running inside of project
- Advanced implementation of OGS standards
- Open solution with agile community of the developers

For network nodes we suppose to test all WMS solutions which can be used by involved data providers. In the fist stage we test above-mentioned open or free solutions.

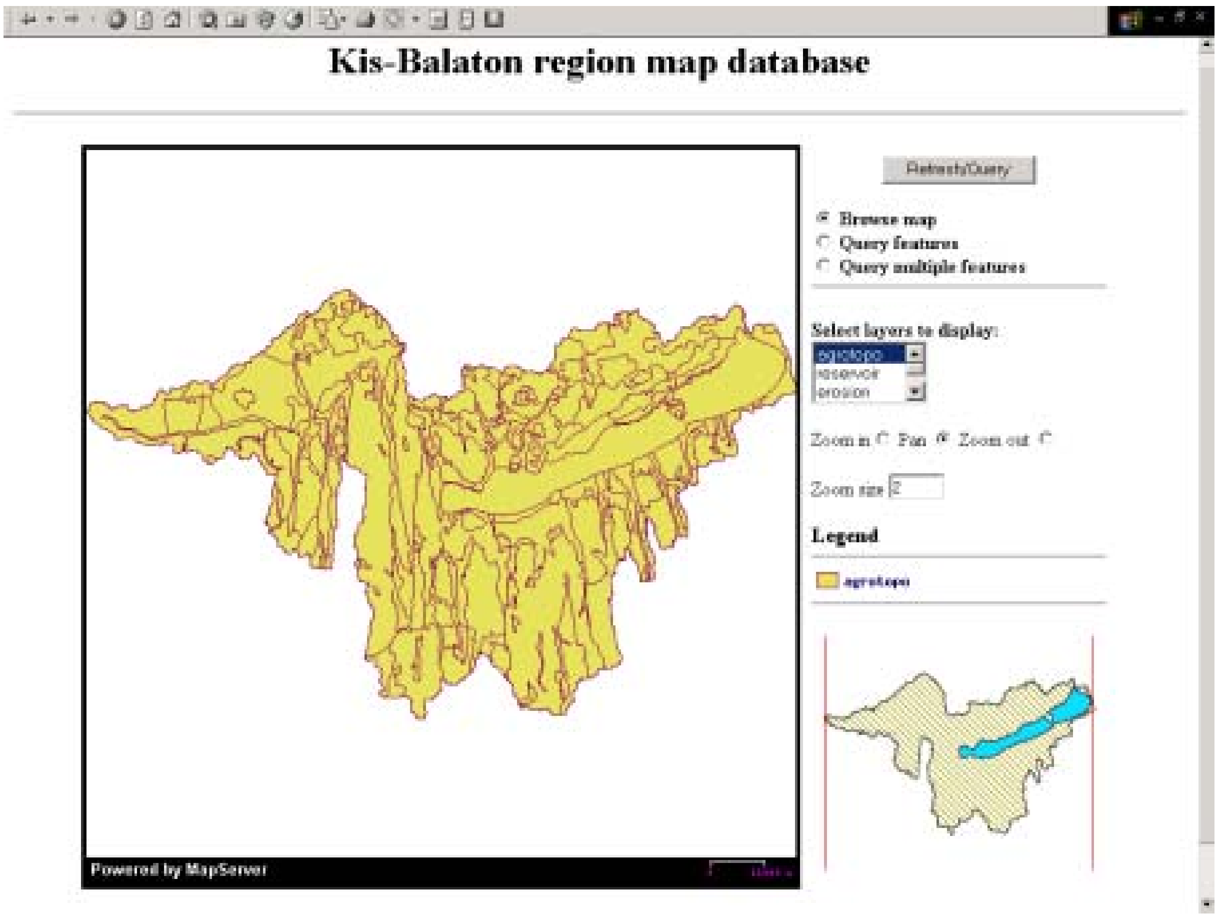
Testing will continue with regular datasets and involvement of commercial GIS data sources. In following stage we plan to test Intergraph Geomedia WebMap Srever and Geoconcept (claim to support WMS, but isn't on OGC list)

Access to server:

http://193.224.81.33/balaton/

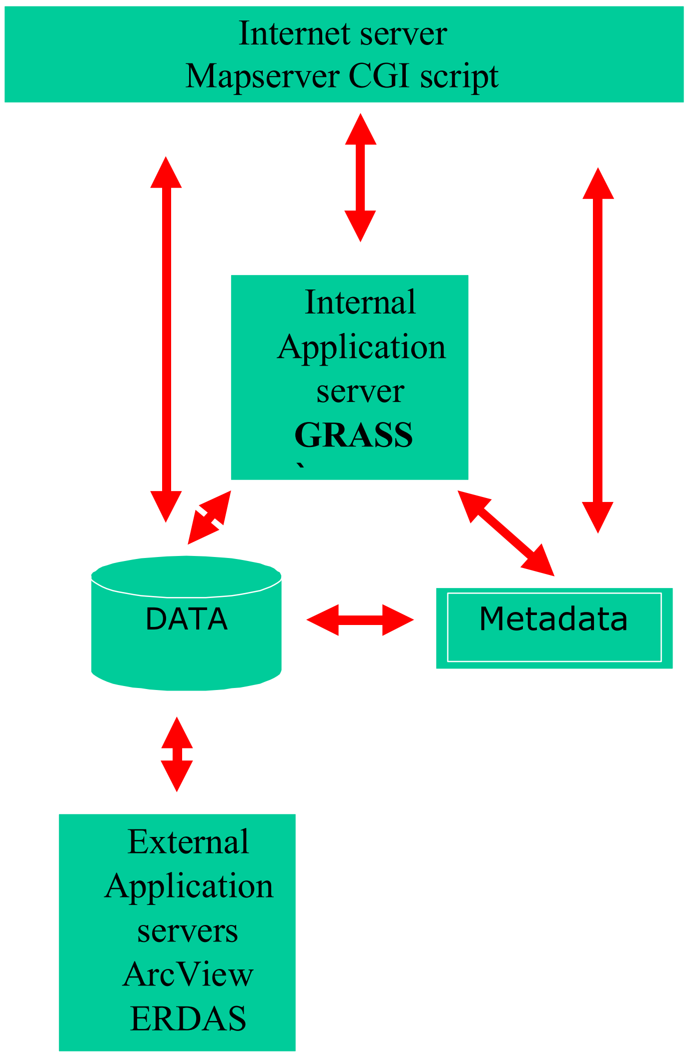
On the appearing main page, it can choose between English and Hungarian versions. You can either quire for data or find maps.
At present, the services of the page can be obtained after an identifying process to make sure that only authorized persons can get access to the information. Safety partitioning of certain parts of the service according to levels of authority is possible.

In the following illustration, the completed surface can be seen.



6 GRASS CONNECTION

The Premathmod architecture will be based on previous experience from WirelessInfo project, but this solution will be modified. The expected architecture will be according following scheme.


Internet server
Mapserver CGI script
Internal Application server GRASS
DATA
Metadata
External Application servers ArcView ERDAS

The new tools in architecture will be:

- Internal Application Server
- External Application server
- Resources

6.1.1 Internal application server

Internal application server will offer new possibilities of data processing trough Internet on line. Grass systems and special statistic libraries will be implemented. The Internal Application Server is new solution in Premathmod system, which offers full analytical possibilities and decision support (DSS). The analytical functions will be accessible trough CGI script and results of analysis will be displayed trough Mapserver application.

6.1.2 External application servers
External applications servers are of line application, which are not connected trough Internet with external users. These technologies were already implemented and are used inside of service companies. Service companies prepares currently all data analysis in Premathmod system and all outputs, use external applications. For data analysis and preparation of application maps is currently use ArcView. Operator analyse incoming data prepare applications maps and this results are stored in Mapserver system. In future is expected, that most from this operations will be provided trough Internet by internal application server. Only specialised tasks as photogrammetric operations will be provided in off line system using external applications..

6.2 Grass
GRASS (Geographic Resources Analysis Support System) is a public domain raster based GIS, vector GIS, image processing system, and graphics production system. GRASS contains over 40 programs to render images on monitor and paper; over 60 raster manipulation programs; over 30 vector manipulation programs; nearly 30 multi-spectral image processing manipulation programs; 16 data management programs; and 6 point file management programs.

7 ANNEX

7.1 *Setting of Minnesota University MapServer for WMS usage*

7.1.1 Base conditions

Necessary condition for establish WMS server based on UMN MapServer is to have installed Web server with CGI support. In case of our testing we use Web server IIS 5.0 on MS Windows 2000.

In the directory /cgi-bin/ (according to document root of the Web server) is need to put UMN MapSever engine in appropriate executable binary form – in our case mapserv.exe. MSU binaries is possible to download from

http://mapserver.gis.umn.edu/dload.html

From several types of the UMN Mapserver distributions is one with WMS support needed. We can ensure that we have appropriate one if we run mapserv.exe with –v switch. We obtain something like this:

```
MapServer version 3.5 OUTPUT=PNG OUTPUT=JPEG OUTPUT=WBMP SUPPORTS=PROJ
SUPPORTS=TTF SUPPORTS=WMS_SERVER INPUT=EPPL7 INPUT=JPEG INPUT=OGR INPUT=GDAL
INPUT=SHAPEFILE
```

7.2 *Configuration of the UMN MapServer*

Whole configuration of the WMS service is based on appropriate modification of the MAP files – i.e. files which defines map object. Example of suitable *.map file following. Important parts are equipped with comments (listing is in ARIAL italic, commented parts are bold):

```
#
# Start of map file
#
NAME BALATON
SIZE 600 600
EXTENT 439705.5625 110751.101563 585105.5625 198951.109375
UNITS METERS
SHAPEPATH "C:\Inetpub\WWWRoot\Balaton\data"
```

Setting of the path to shapefiles is necessary define in form of local computer directory system, relative paths are not recommended.

```
#
# Start of web interface definition
#
WEB
  HEADER first_h.htm
  TEMPLATE first.htm
  FOOTER first_f.htm
  MINSCALE 1000
  MAXSCALE 1550000
  IMAGEPATH "C:\inetpub\wwwroot\tmp\"
  IMAGEURL "C:\inetpub\wwwroot\tmp\"
```

This is setting of the directory for storage of generated image files

```
END  #WEB
#
# Start of legend
#
LEGEND
KEYSIZE 18 14
  LABEL
    TYPE BITMAP
    SIZE MEDIUM
    COLOR 2 0 89
  END
  STATUS ON
END   #LEGEND
#
# Start of scalebar
#
SCALEBAR
  IMAGECOLOR 255 255 0
  LABEL
    COLOR 255 0 255
    SIZE tiny
  END
  STYLE 1
  SIZE 80 4
  COLOR 255 255 255
  UNITS METERS
  INTERVALS 1
  TRANSPARENT TRUE
  STATUS TRUE
END   #SCALEBAR

#
# Start of layer definitions
#
LAYER
  NAME agrotopo
  TYPE polygon
  STATUS on
  DATA agrotopo
  CLASS
    NAME 'agrotopo'
    TEMPLATE "agrotopo.htm"
    COLOR 225 225 90
    OUTLINECOLOR 128 0 90
  END
  HEADER "agrotopo_h.htm"
  FOOTER "agrotopo_f.htm"
  TOLERANCE 3
END  # LAYER
END  # MAPFILE
```