ÇANKAYA UNIVERSITY

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

COMPUTER ENGINEERING

MASTER THESIS

# DEVELOPING A GIS BASED CRIME ANALYSIS SYSTEM

MUSTAFA ÖZÇETİN

FEBRUARY 2013

Title of the Thesis: **Developing a GIS Based Crime Analysis System**

Submitted by : **Mustafa Özçetin**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University

[illegible]

Prof. Dr. Taner Altunok
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

[illegible]

Assist. Prof. Dr. Murat Saran
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

[illegible]

Assoc. Prof. Dr. H. Hakan Maraş
Supervisor

**Examination Date** 08.02.2013

**Examining Committee Members**

Assoc. Prof. Dr. H. Hakan Maraş (Çankaya Univ.) [illegible]

Assoc. Prof. Dr. S. Savaş Durduran (Selçuk Univ.) [illegible]

Assist. Prof. Dr. Abdül Kadir Görür (Çankaya Univ.) [illegible]

## STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

| | |
|---|---|
| Name, Last Name : | Mustafa ÖZÇETİN |
| Signature : | [illegible] |
| Date : | 08.02.2013 |

# ABSTRACT

DEVELOPING A GIS BASED CRIME ANALYSIS SYSTEM

Özçetin, Mustafa

M.S.c., Department of Computer Engineering

Supervisor: Assoc. Prof. Dr. H. Hakan Maraş

February 2013, 74 pages

Geographic Information Systems (GIS) have a lot of applications that are getting more and more importance in our daily life. One of the hot and crucial application areas of GIS is crime mapping. Nowadays, crime rates tend to increase and thus analyzing the crime trends and taking preventive precautions are very important. Often, crimes display *spatial* or *temporal* patterns. For instance, some crime types are committed in some areas with a relatively higher density rate. Some others may occur in particular time ranges in a day. Adding geographic support to crime analysis methods can provide extremely valuable and exclusive benefits which tabular or statistical analyses cannot provide. For example, seeing the spatial distribution of a particular crime type or comparing different types of crimes *on the map* can give critical and important decision making tips for managers. Hence, *adding spatial dimension* to crime analysis techniques help provide great contributions to police departments.

In this study, after doing some researches about crime mapping, a GIS-based crime analysis system, namely *Crime Analyzer*, has been developed from scratch. Crime Analyzer not only provides various spatial and temporal analysis tools but also presents a flexible method to extract data from various sources such as database systems to solve crime mapping problems.

**Keywords:** GIS Based Crime Analysis, Crime Mapping Software

# ÖZ

## CBS TABANLI SUÇ ANALİZİ YAZILIMI GELİŞTİRME

Özçetin, Mustafa

Yükseklisans, Bilgisayar Mühendisliği Anabilim Dalı

Tez Yöneticisi : Doç. Dr. H. Hakan Maraş

Şubat 2013, 74 sayfa

Coğrafi Bilgi Sistemleri (CBS) günlük hayatımızda önemi sürekli olarak artan uygulamalara sahiptir. CBS'nin popüler ve kritik uygulama alanlarından biri de mekansal suç analizidir. Günümüzde suç oranları artış göstermekte ve bu yüzden suç eğilimlerini analiz etmek ve suçu önleyici tedbirler almak büyük önem arz etmektedir. Suçlar çoğu zaman *mekansal* ve *zamansal* modeller göstermektedir. Örneğin, bazı suç türleri bazı lokasyonlarda nispeten yüksek oranlarda işlenebilmektedir. Bazıları ise gün içinde belli saat aralıklarında yüksek oranlarda olabilmektedir. Klasik suç analiz yöntemlerine coğrafi destek eklemek, tablosal veya istatistiksel metotların sunamayacağı son derece önemli ve müstesna faydalar sağlayabilir. Örneğin, belli bir suç türünün mekansal dağılımını görmek veya farklı suç türlerinin lokasyonlarını *harita üzerinde* karşılaştırmak, karar verme pozisyonundaki yöneticilere kritik ve önemli ipuçları verebilir. Bu yüzden, suç analiz yöntemlerine *mekansal boyut katmak* emniyet birimlerindeki  çalışmalara önemli bir destek sağlamaktadır.

Bu çalışmada, suçların haritalanması konusunda gerekli araştırmalar yapıldıktan sonra *Crime Analyzer* adlı bir CBS tabanlı suç analizi yazılımı geliştirilmiştir. Crime Analyzer, suç analizinde karşılaşılan problemleri çözmede muhtelif mekansal ve zamansal analiz araçları sunmanın yanı sıra, veritabanı sistemleri gibi çeşitli veri kaynaklarından veri alma konusunda esnek bir yöntem sağlar.

**Anahtar Kelimeler:** CBS Tabanlı Suç Analizi, Suç Haritalama Yazılımı

## ACKNOWLEDGEMENTS

The author wishes to express his deepest gratitude to his supervisor Assoc. Prof. Dr. H. Hakan Maraş for his guidance, advice, criticism, encouragements and insight throughout the research.

Also, I would like to thank to my wife Emel for her love and morale support.

To my tiny son, Yusuf

## TABLE OF CONTENTS

## LIST OF TABLES

# LIST OF FIGURES

# LIST OF ABBREVIATIONS

| | |
|---|---|
| ADT | Abstract Data Type |
| API | Application Programming Interface |
| CAD | Computer Aided Design (also Computer Aided Drafting) |
| COM | Component Object Model |
| CompStat | Computer Statistics (also Comparative Statistics) |
| DAL | Data Access Layer |
| DB | Database |
| DDE | Dynamic Data Exchange |
| DLL | Dynamic Link Library |
| ERP | Enterprise Resource Planning |
| GIS | Geographic Information System |
| IPC | Inter-Process Communication |
| MDI | Multiple Document Interface |
| MS | Microsoft |
| OLE | Object Linking and Embedding |
| SID | System ID (for an Oracle DB) |
| SQL | Structured Query Language |
| UI | User Interface |
| VS | Visual Studio |

# INTRODUCTION

## The Purpose and Scope of the Study

The purpose of this study is to develop viable and usable spatial crime analysis system from scratch, to perform various analyses on some imaginary crime data and to be able to interpret the analysis results.

Currently there are lots of crime mapping techniques in accordance with the purpose of the analysis. This study aims to implement the widely used and practical analysis methods.

## Roadmap

This study mainly consists of three parts: First, some theory will be provided about GIS, crime analysis and crime mapping. Then, features of the system together with some crime scenarios will be explained. Finally, architecture and implementation of the system will be explored.

## Organization of the Thesis

Chapter 1 makes a general introduction to GIS, crime analysis and crime mapping. The needs for making crime analyses on maps are also discussed.

Chapter 2 mentions the existing crime mapping software on the market. Main features of commonly used three programs are explained.

Chapter 3 explains the features of the system in detail. This includes the capabilities, analysis tools and screenshots of the software developed.

Chapter 4 explores the high-level design and architecture of the system developed. Development environment and the mapping framework on which the system is constructed are specified.

Chapter 5 expresses the development and the technical features of the system. Implementation details with class diagrams are discussed in some detail.

Chapter 6, finally, provides the summary and conclusion of the study.

# CHAPTER I

# GIS AND CRIME ANALYSIS

Before entering the actual topic, namely crime mapping, it will be useful to give some introductory information about GIS. This will make sense because crime mapping is one of the huge application areas of GIS.

## 1.1. INTRODUCTION TO GIS

Although there are various definitions of GIS, one of them may be given as

“GIS is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data. In the simplest terms, GIS is the merging of cartography, statistical analysis, and database technology.

In a general sense, the term describes any information system that integrates, stores, edits, analyzes, shares, and displays geographic information for informing decision making. GIS applications are tools that allow users to create interactive queries (user-created searches), analyze spatial information, edit data in maps, and present the results of all these operations [1].”

According to Maras’ comprehensive definition, "a Geographic Information System is an integration of computer hardware, software, personnel, data and methods, which enables to capture, store, integrate, manipulate, manage, analyze, portray and to graphically render spatially referenced information so as to meet the user requirements [2].”

One can think that GIS is like Computer Aided Design (CAD) but there are important differences between them. CAD programs are basically graphics programs and drawing objects such as lines, cubes etc. are important *themselves*. But GIS objects are actually *representations* of the real world entities and they have *attribute data* that are stored in a database tied with them. Hence these objects are sometimes called *smart*. Another difference is that all geographic data in a GIS has a geographic reference defined with a coordinate system and a projection and thus they all correspond to some location on the Earth.

Basic components of spatial data are points, lines and polygons. These components constitute layers when they serve common characteristics. A layer is “the visual representation of a geographic dataset in any digital map environment. Conceptually, a layer is a slice or stratum of the geographic reality in a particular area, and is more or less equivalent to a legend item on a paper map. On a road map, for example, roads, national parks, political boundaries, and rivers might be considered different layers [3].” Layers are building blocks of maps. Apart from being references to specific data sources, layers can also be associated with cartographic symbols. Layers generally follow a logical z-order in which region layers are placed at the bottom, line layers are top of the region layers and point layers are topmost.

A significant operation in GIS is called *geoprocessing*. Geoprocessing takes some input features, perform some operations with these features and returns the resulting features. Geoprocessing is commonly used in real-world analyses. For example, using geoprocessing tools risk analyses such as flood analysis can be easily made around a river. This involves creating a buffer region around the river and performing a geographic overlay between the buffer and the buildings.

## 1.2. BENEFITS OF GIS

GIS provides concrete benefits to both organizations and individuals in daily life. It not only relates with enterprise or desktop applications in computers but also provides extremely useful applications with navigation devices and mobile phones. Actually any citizen may directly or indirectly get the benefits of GIS even he/she does not involve any of the technological devices like computers or mobile phones. For instance, some municipalities plan bus routes after performing some line optimization analyses. These may include inspecting the population density or residence of old people.

Utilizing a GIS, managers will have a solid and scientific decision support system by performing valuable analyses. Organizations may reduce their expenses with optimized and more efficient planning by applying solutions like shortest path or travelling salesman. Customer services can be improved with analyzing the addresses of nearby customers. Natural resources can be managed more effectively. Setting up a fully integrated infrastructure between organizations, serious industrial accidents like breaking fiber cable lines in excavations can be prevented. Storing tabular and spatial data together in a central system greatly contributes to secure, consistent and updated data. Performing analyses for under-served population locations, public services may be refined. Enterprise Resource Planning (ERP) applications may be greatly supported and enhanced with spatial data.

## 1.3. APPLICATION AREAS OF GIS

GIS has a broad range of application areas. Rather than mentioning all of them, it will be appropriate to give some important ones:

- Urban planning and management
- Land cadastral system and record management
- Emergency management (automatic address locating from an incoming phone call etc.)
- Public infrastructure management (electricity, water, communication lines etc.)
- Public safety (fire prevention and handling etc.)
- Vehicle routing and navigation (shortest or optimal path etc.)
- Network analysis (finding the valves to be closed in case of natural gas leak etc.)
- Military, defense and intelligence
- Election result analysis according to geographic locations
- Environmental risk monitoring
- Agriculture (soil analysis, yield estimation etc.)
- Natural resource research and management (water, petroleum, mining etc.)
- Civil engineering (deciding the underground construction locations, designing highways etc.)
- Demographic analysis for public services
- Transportation and logistics (vehicle tracking etc.)

## 1.4. CRIME ANALYSIS

“Crime analysis is a law enforcement function that involves *systematic* analysis for identifying and analyzing *patterns* and *trends* in crime and disorder. Information on patterns can help law enforcement agencies deploy resources in a more effective manner, and assist detectives in identifying and apprehending suspects. Crime analysis also plays a role in devising solutions to crime problems, and formulating crime prevention strategies. Quantitative social science data analysis methods are part of the crime analysis process; though qualitative methods such as examining police report narratives also play a role [4].”

“Crime analysis can occur at various levels, including tactical, operational, and strategic. Crime analysts study crime reports, arrests reports, and police calls for service to identify emerging patterns, series, and trends as quickly as possible. They analyze these phenomena for all relevant factors, sometimes predict or forecast future occurrences, and issue bulletins, reports, and alerts to their agencies. They then work with their police agencies to develop effective strategies and tactics to address crime and disorder. Other duties of crime analysts may include preparing statistics, data queries, or maps on demand; analyzing beat and shift configurations; preparing information for community or court presentations; answering questions from the public and the press; and providing data and information support for a police department's CompStat (Computer Statistics) process [5].”

Sociodemographics, along with spatial and temporal information, are all aspects that crime analysts look at to understand what's going on in their jurisdiction [6]. “Crime analysis employs data mining, crime mapping, statistics, research methods, desktop publishing, charting, presentation skills, critical thinking, and a solid understanding of criminal behavior. In this sense, a crime analyst serves as a combination of an information systems specialist, a statistician, a researcher, a criminologist, a journalist, and a planner for a local police department [7].”

## 1.5. CRIME MAPPING

“Crime mapping is used by analysts in law enforcement agencies to map, visualize, and analyze crime incident patterns. It is a *key component* of crime analysis and the CompStat policing strategy. Mapping crime, using Geographic Information Systems (GIS), allows crime analysts to identify crime hot spots, along with other trends and patterns [8].”

With a high-level and informal phrase, crime mapping tries to answer the questions “Which crime happens where, when and why?” Using classical statistical methods only may not be sufficient to make in-depth analyses to understand the crime patterns since crimes often have *spatial dimension*. In other words, some particular crime types may be committed in some particular locations. For example, according to a formal study, robberies generally concentrate on the central locations of settlements [9].

Some studies have demonstrated that offenders *tend* to commit their crimes between their home, workplace and facilities they attend. Hence, even these particular examples show that it is *essential* to inspect the crime data geographically.

# CHAPTER II

## EXISTING STUDIES

Currently there is not so much crime mapping software on the market. ESRI and MapInfo (actually Pitney Bowes Software now) have commercial crime mapping solutions and they are the most commonly used programs. Since ArcGIS Desktop programs (ArcView / ArcEditor / ArcInfo) and MapInfo Professional are general-purpose desktop mapping programs, they do not provide crime mapping tools as built-in. This is reasonable since crime mapping is a specific area and thus they present the crime mapping solution in the form of extensions. ESRI's solution is called *CrimeAnalyst* and MapInfo's is called *Crime Profiler*. These extensions add new tools and capabilities to the existing program that can be used to perform spatial crime analyses.

Also there is a free spatial statistics application, namely *CrimeStat*, developed under a grant from the National Institute of Justice of the USA. Although it is not GIS software, it can read files which are geocoded by other GIS software, like ArcGIS and MapInfo, and can export its results into formats that those programs can read.

## 2.1. CRIMEANALYST FOR ARCGIS

According to the vendor's description, "CrimeAnalyst allows making smarter use of the intelligence at disposal – and this, in turn, can help simultaneously reduce both crime and costs [10]."

"First launched in 2005, CrimeAnalyst has become well established in the public safety community and is used in police forces, crime prevention partnerships and intelligence groups across the UK and around the world. The latest version, CrimeAnalyst 2.1, builds on the proven capabilities of the solution. Responding to the needs of end users, it has been enhanced with a range of new capabilities that make it even easier and quicker to develop National Intelligence Model (NIM) compliant analytical products [10]."

Features of CrimeAnalyst include:

- “Crime hot spot / density maps.
- Temporal crime analysis / aoristic data clocks.
- Spatial distribution and analysis.
- Identifications of repeat victims of crime.
- Identifying connections between related locations in a crime.
- Calculating spatial statistics [11].”

The map in Figure 1 displays the hot spots identifying good places for detection cameras.

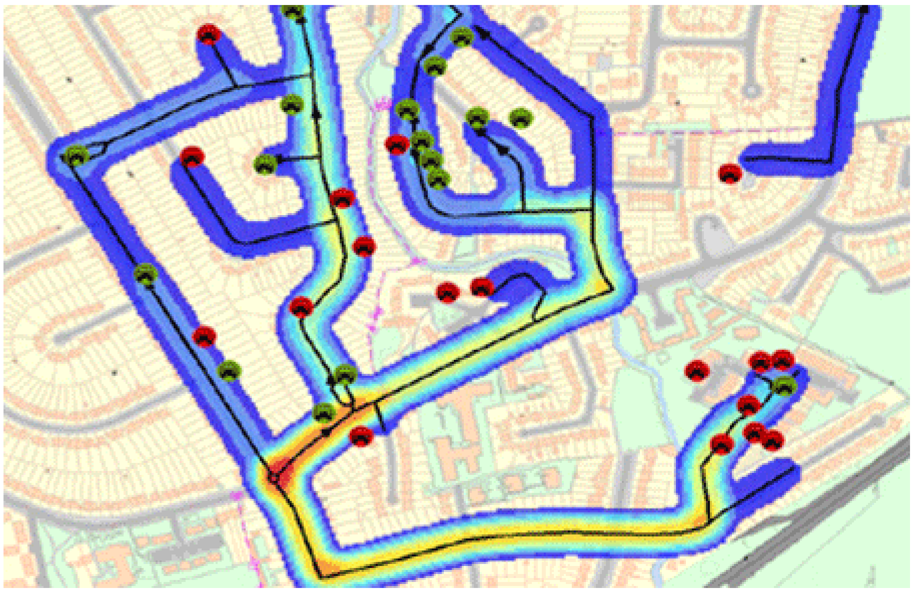


**Figure 1**: A linear hot spot analysis in CrimeAnalyst.

Not all analysis types in CrimeAnalyst involve maps and layers. There are also *temporal analysis* tools like data clocks. Data clock analysis can easily spot the dense crime hours in a weekly period as shown in Figure 2.

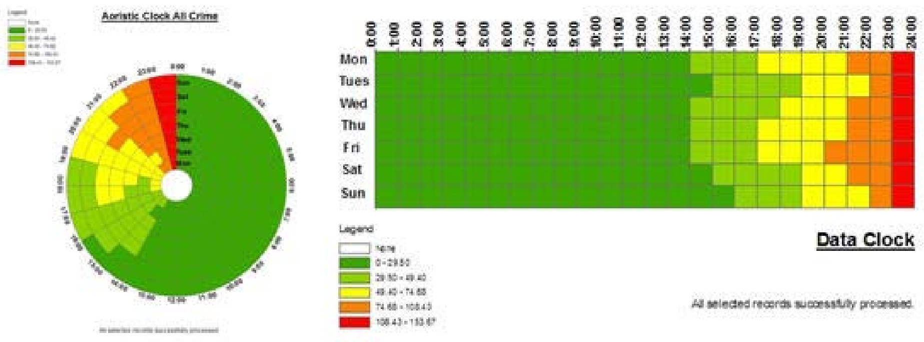


**Figure 2**: A weekly data clock analysis in CrimeAnalyst.

## 2.2. MAPINFO CRIME PROFILER

With the vendor’s description, “MapInfo Crime Profiler is a sophisticated crime mapping and analysis solution from Pitney Bowes Software. Crime Profiler has been designed to help organizations find the best tactics and strategies for dealing with crime and incidents of all types. By providing deeper layers of criminal intelligence it enables law enforcement agencies to manage and more effectively deploy police resources and address key policing challenges within volume crime such as burglary, car theft and antisocial behavior, and serious organized crime such as fraud, drug trafficking and metal theft [12].”

Main capabilities of MapInfo Crime Profiler are:

- Interactive hot spot map creation.
- Various kinds of charts and graphs that are linked with the map data.
- Working with subsets of specific crime data, by series, type, time, locale or other characteristics.
- Proximity analysis (distance or journey based analysis) of crimes to known offenders.
- Mapping functionality that allows working with map data in a wide variety of formats and types including aerial photography, street maps, demographics, segmentation maps and more.
- The ability to define force boundaries such as beats and patrol areas.
- Ability to provide maps, charts, graphs and reports as a printout or in a variety of other formats for sharing with all levels of the organization [13].

Using MapInfo Crime Profiler, different data visualization types can easily be created. For example, Figure 3 shows a data clock, an individual values thematic, a chart and a table of crime data in a single screen.

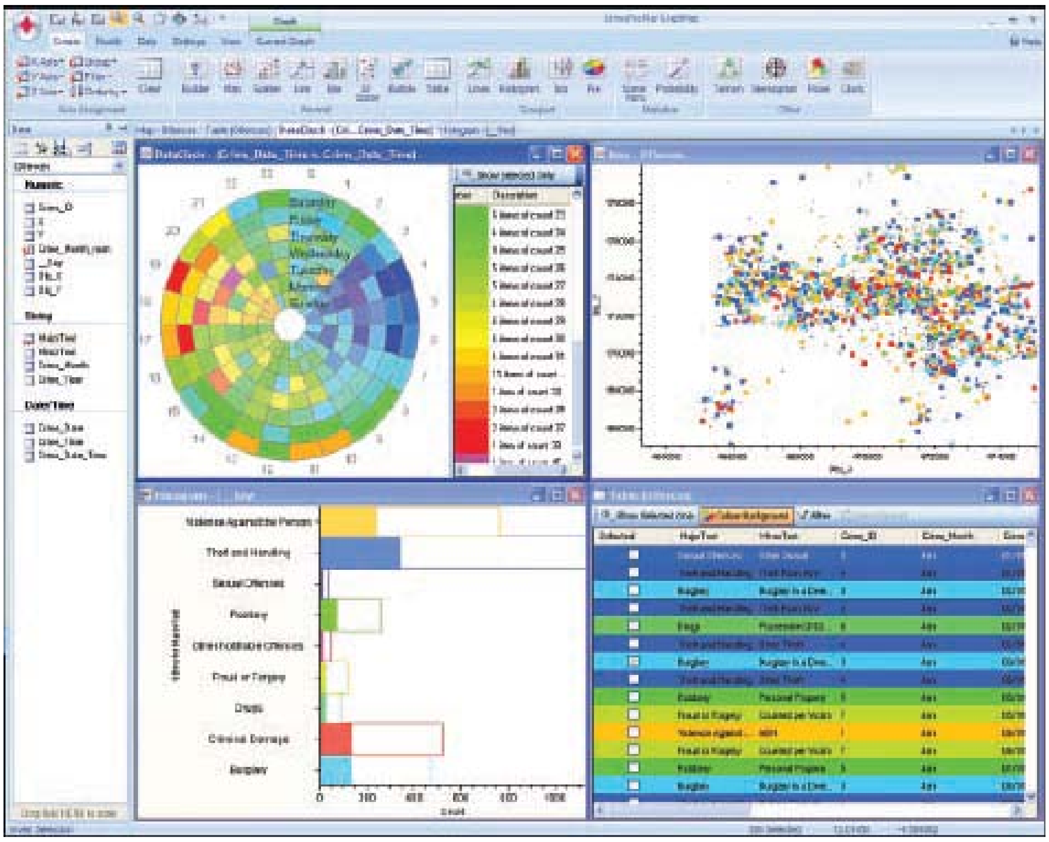


**Figure 3**: MapInfo Crime Profiler overview.

## 2.3. CRIMESTAT

“CrimeStat is a spatial statistics program for the analysis of crime incident locations, developed by Ned Levine & Associates under the direction of Ned Levine, PhD, that was funded by grants from the National Institute of Justice. The purpose is to provide supplemental statistical tools to aid law enforcement agencies and criminal justice researchers in their crime mapping efforts. CrimeStat is being used by many police departments around the country as well as by criminal justice and other researchers [14].”

“The program inputs incident locations (e.g., robbery locations) in 'dbf', 'shp', ASCII or ODBC-compliant formats using either spherical or projected coordinates. It calculates various spatial statistics and writes graphical objects to ArcGIS®, MapInfo®, Surfer for Windows®, and other GIS packages [14].”

# CHAPTER III

## FEATURES OF THE SYSTEM

### 3.1. GENERAL UI FEATURES

The name of the software that has been developed is *Crime Analyzer*. One of the remarkable features of Crime Analyzer is Multiple Document Interface (MDI) support. Actually this is essential for analysis software since the analyst may want to see multiple analysis results on a single screen. Typically, the analyst creates different data visualizations of some crime data and these visualizations in fact may be related to each other. Hence, rather than creating separate analysis outputs one by one and combining them by hand, the analyst can get a single screenshot of all the output or print the screen easily.

Figure 4 shows a screenshot of Crime Analyzer after creating 4 different analyses:

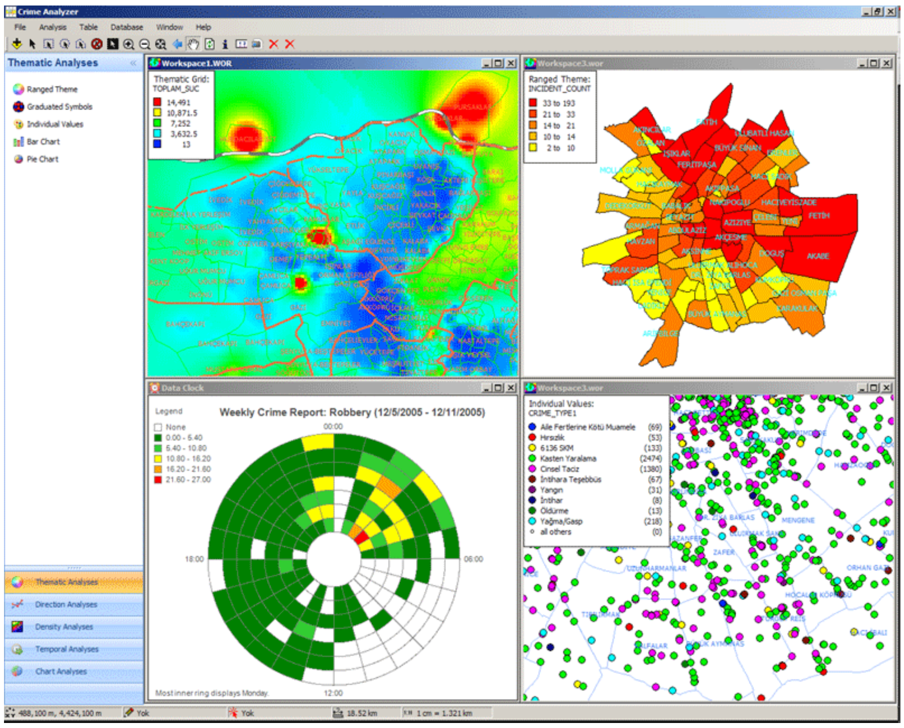


**Figure 4**: Crime Analyzer overview.

The parent, or main, window of Crime Analyzer consists of the following parts:

- Main menu
- Map toolbar
- Navigation bar
- Map statusbar

Figure 5 depicts the main components of Crime Analyzer main window.

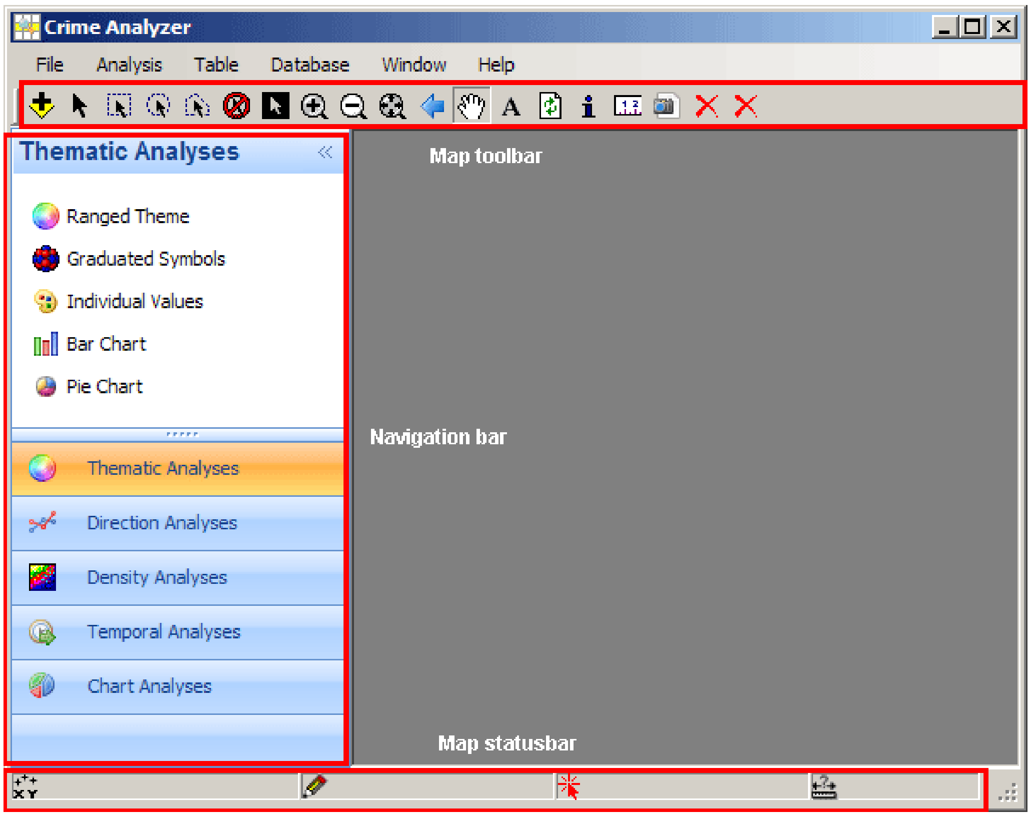


**Figure 5**: Components of Crime Analyzer main window.

Map toolbar consists of buttons to be used for map operations such as zooming, panning and selecting, as shown in Figure 6.

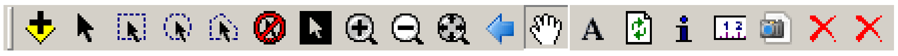


**Figure 6**: Map toolbar buttons.

There are two types of buttons:

**Command buttons**: When clicked, immediately perform an action such as unselecting all objects or displaying the full extent of a layer.

**Tool buttons**: These are similar to command buttons but they require some interaction with the map such as zooming or panning. When the user clicks a zoom in tool, for instance, he/she is supposed to draw a rectangle to be zoomed on the map. After that, the map display is updated to reflect the user's rectangle.

Navigation bar is the panel on the left hosting the links to the main modules of the application. It consists of horizontal light blue bands except for the selected band, which is shown as orange. Each band holds, or groups, related analysis modules in it. For example *Thematic Analyses* is the top band in the navigation bar and it holds the five thematic analysis types Ranged Theme, Graduated Symbols, Individual Values, Bar Chart and Pie Chart.

Map statusbar is the statusbar at the bottom of the main window. Its function is to display the various properties of the active map window such as the current scale. Figure 7 displays the close up of the map statusbar slots:

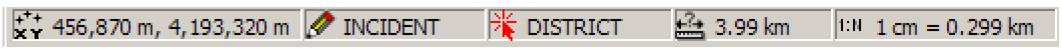


**Figure 7**: Map statusbar.

The first slot displays the current cursor coordinates of the map, which is common to all mapping software. While the user moves the mouse pointer over the map, the exact coordinates of the pointer are instantly displayed here. The second slot specifies the currently edited layer and the third one depicts the currently selected layer. The fourth slot shows the current width of the map and the fifth slot denotes the current map scale.

## 3.2. MAPPING COMPONENT

Crime Analyzer uses MapInfo Professional as the mapping component. Thanks to the *integrated mapping* support, it is possible to fully control and manipulate MapInfo Professional from a client application written with a COM compliant platform such as .NET. In an integrated mapping application, a MapInfo map is re-parented, or displayed, by the client application's user interface (UI), such as in a panel or picture box. Every UI component in Crime Analyzer, except for the map, such as menus, child windows or buttons belongs to Crime Analyzer, which is a .NET application.

## 3.3. PULLING DATA FROM A DATABASE

While performing crime analyses, Crime Analyzer naturally needs some crime data, such as crime count for a particular crime type. This information may reside in a MapInfo table already as a numerical column. This is the simplest case with respects to data pulling to make analyses. But in a corporate environment, the main data sources for the crimes usually exist in relational databases like Microsoft Access or Oracle. The crime analyst (will be called just *analyst* from now on) often needs last-minute information to be able to get up to date analysis results. Therefore, if the analysis software does not provide a mechanism to automatically pull the required data from the corporate database and join it with a MapInfo table, then the analyst will have to accomplish these time consuming operations by hand. Undoubtedly, such software will not be practical and efficient. To prevent this situation, Crime Analyzer lets the analyst to connect to a corporate database, to list tables and columns, to perform flexible queries and to join the result data set with a MapInfo table that will be displayed in the analysis, such as a district or neighborhood table.

Before pulling data from a database, first the analyst has to connect to that database. This can be done with clicking the Database > Connect… menu. Then the **Database Connection** dialog in Figure 8 will pop up:

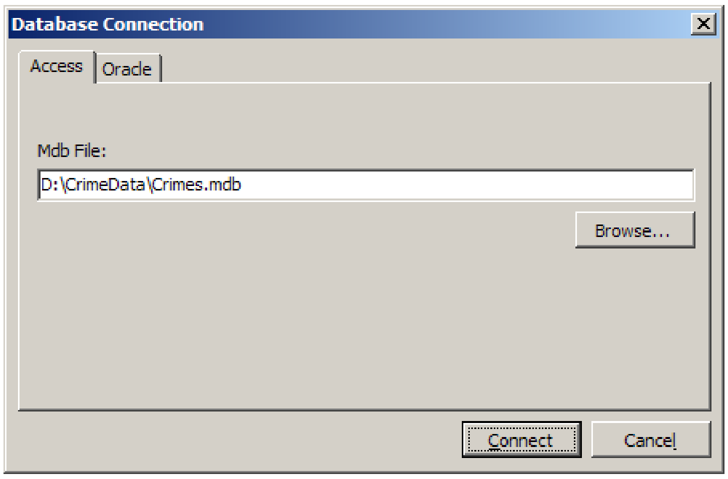
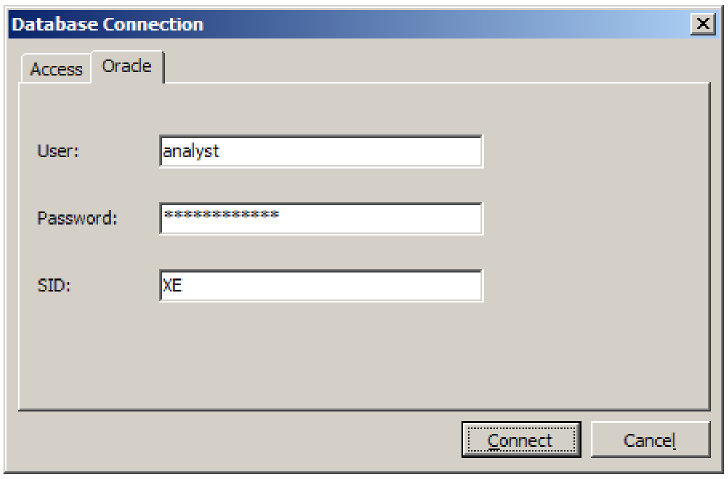


**Figure 8**: Connecting to an Access database.

Currently two database types, namely *Microsoft Access* and *Oracle*, are supported as the corporate data sources. Figure 8 shows how to connect to an Access database. Figure 9 shows how to connect to an Oracle database. It requires entering the user name, password and SID (Oracle System ID) information.



User: analyst

Password: *************

SID: XE

**Figure 9**: Connecting to an Oracle database.

After connecting to a database, the database tables will be available for Crime Analyzer and it will be able to perform automatic joins that MapInfo tables will need.

## 3.4. ANALYSIS MODULES

### 3.4.1. Thematic Analyses

A thematic map is a type of map or chart especially designed to show a particular theme connected with a specific geographic area. These maps can portray physical, social, political, cultural, economic, sociological, agricultural, or any other aspects of a city, state, region, nation, or continent [15]. Creating a thematic map typically involves applying a theme to a table according to the values of a numerical column. For example, in a map, cities with a high population density can be displayed by darker colors.

The column used in generating a thematic map is sometimes referred to as *thematic variable*. In fact, this thematic variable is not essentially a column. It can also be an *expression* of two or more columns such as average of two columns.

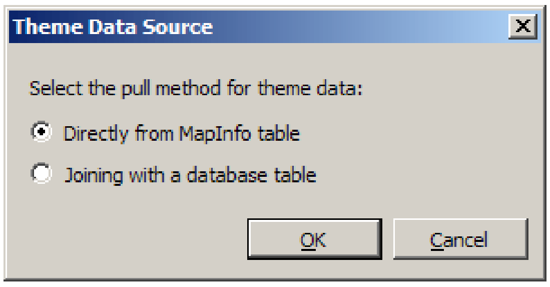
Thematic maps can present valuable and exclusive information since they can give various visualizations of geographic data that tabular data can never provide. Moreover interpreting a thematic map can generally be achieved at a first look. For these reasons, dealing with thematic maps is important in crime mapping.

#### 3.4.1.1. Ranged Theme

One of the most commonly used thematic types is ranged thematic. Ranged thematic typically *shades* or *colors* a predefined regional layer, based on the numerical values of a column. This regional layer may be a country, city or district layer. If the regional layer has one or more numerical columns such as population or crime counts in its attribute table, then it is possible to create a thematic layer based on these column values. For instance, districts with higher crime rates may appear darker in the thematic map. Ranged thematic is also called *choropleth* mapping.

Ranged thematic is appropriate for crime mapping and thus it is one of the thematic types that Crime Analyzer implements.

The first step in creating a ranged thematic is to specify the data source. Crime Analyzer asks the thematic data source with the **Theme Data Source** dialog shown in Figure 10:


Theme Data Source

Select the pull method for theme data:

[✓ Directly from MapInfo table]

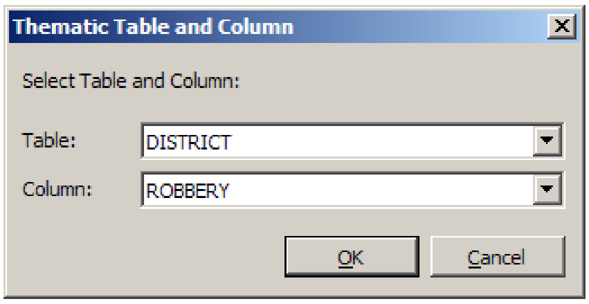
Joining with a database table

**Figure 10**: Selecting a theme data source.

It will be suitable to inspect these two cases separately.

### 3.4.1.1.1. Ranged Theme Directly From a MapInfo Table

The first option in Figure 10 is appropriate if the region table to shade already has the numerical column and the column is filled with the values. This way is relatively easy to create a thematic map. After selecting this option and then clicking the OK button, the **Thematic Table and Column** dialog in Figure 11 is shown to specify the region table and the thematic column.


Thematic Table and Column

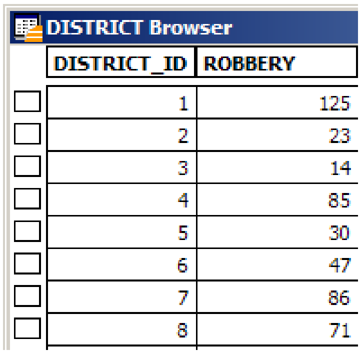
Select Table and Column:

Table: DISTRICT

Column: ROBBERY

**Figure 11**: Selecting the region table to shade and thematic column.

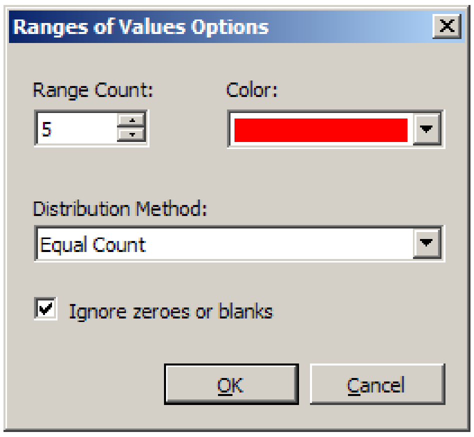
The selected column in Figure 11, namely ROBBERY, is a numerical column storing the robbery count in each district. Therefore, the thematic map to be displayed will be based on the number of robberies. The attributes of the DISTRICT table in MapInfo may look like in Figure 12:



| DISTRICT_ID | ROBBERY |
|---|---|
| 1 | 125 |
| 2 | 23 |
| 3 | 14 |
| 4 | 85 |
| 5 | 30 |
| 6 | 47 |
| 7 | 86 |
| 8 | 71 |

**Figure 12**: Attribute data of the District table and column.

The last step creating the thematic map directly from a MapInfo table is to select the thematic options via **Ranges of Values Options** dialog in Figure 13.



Range Count: 5
Color: 🟥
Distribution Method: Equal Count
✓ Ignore zeroes or blanks

**Figure 13**: Specifying the ranged thematic options.

In the **Ranges of Values Options** dialog, range count, color theme and distribution method can be set. Also there is an option to specify whether the records having zero or blank value will be ignored or not. Distribution methods will be explained in the section 4.4.1.1.3.

Clicking the OK button will finish the thematic map and display it on the current map window as in Figure 14.

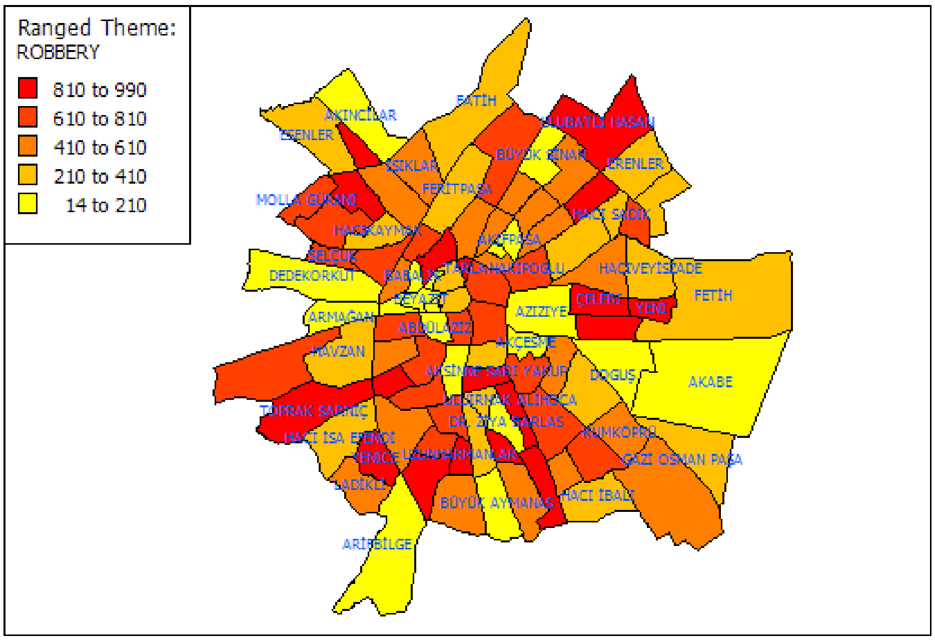
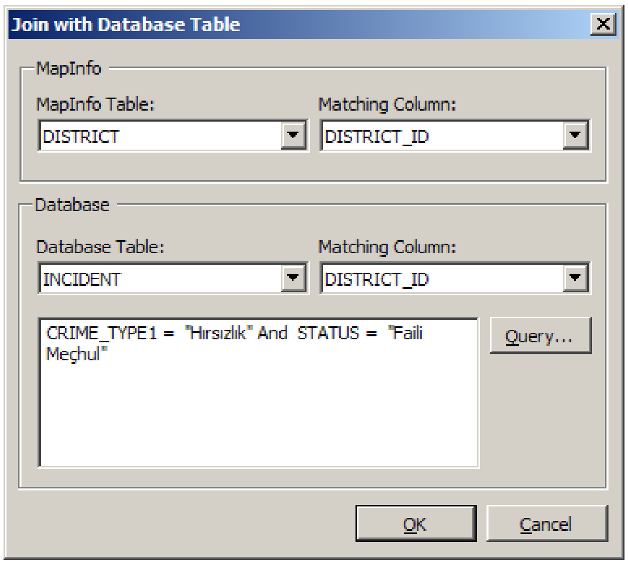


**Figure 14**: Ranged thematic with a MapInfo table as the data source.

## 3.4.1.1.2. Ranged Theme by Joining with a Database Table

The second option in Figure 10 allows the analyst to pull data automatically from a corporate database to use as the thematic data. This is a powerful and flexible method to be able to use the last minute information to generate up to date analysis results. This is performed by first populating a dataset from the database according to the query criteria specified by the analyst, and then joining this dataset with the MapInfo table to shade. To achieve this, **Join with Database Table** dialog is used as in Figure 15.


Join with Database Table

MapInfo

MapInfo Table: DISTRICT

Matching Column: DISTRICT_ID

Database

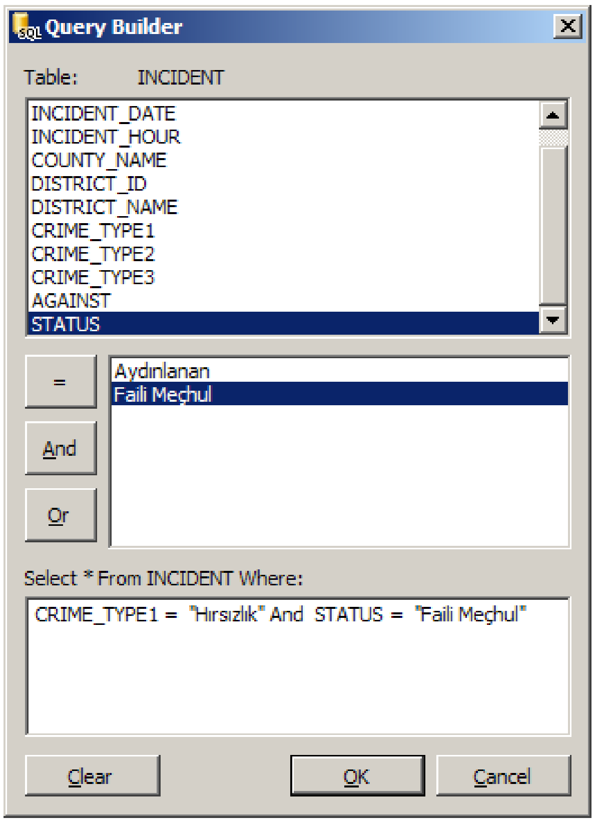
Database Table: INCIDENT

Matching Column: DISTRICT_ID

CRIME_TYPE1 = "Hırsızlık" And STATUS = "Faili Meçhul"

**Figure 15**: Join with Database Table dialog.

The join operation requires two matching columns from the MapInfo table and the database table. Using the text area at the bottom, query expressions can be specified against the database table. This is an extremely useful feature for the analyst since he/she can filter the data with any filters like crime type, status or time interval. Although it is possible to write the query by hand, a more suitable method is to use the *Query Builder* dialog. For that, the analyst uses the Query… button in Figure 15. When clicked, it opens up the **Query Builder** dialog in Figure 16:



CRIME_TYPE1 = "Hırsızlık" And STATUS = "Faili Meçhul"

**Figure 16**: Query Builder dialog.

Query Builder dialog lists the columns of the table which is selected in the Database Table field in the **Join with Database Table** dialog in Figure 15. To generate the query within the text area at the bottom, the analyst first selects the column to filter. Selecting a column automatically lists its unique values in the list at the middle. Double clicking the selected column appends it in the query text area. To insert an operator such as equals (=), the analyst clicks the = button on the middle left. Finally, double clicking an item in the values list will append the item to the query text area. In this way, the analyst can generate flexible filters against the database table. Clicking the OK button in the Query Builder dialog closes the dialog and transfers the query text to the text area in **Join with Database Table** in Figure 15.

The next step to generate the thematic map with joining with a database table is the same as the step in 4.4.1.1.1. That is, it involves setting the thematic options. The result is displayed in Figure 17:

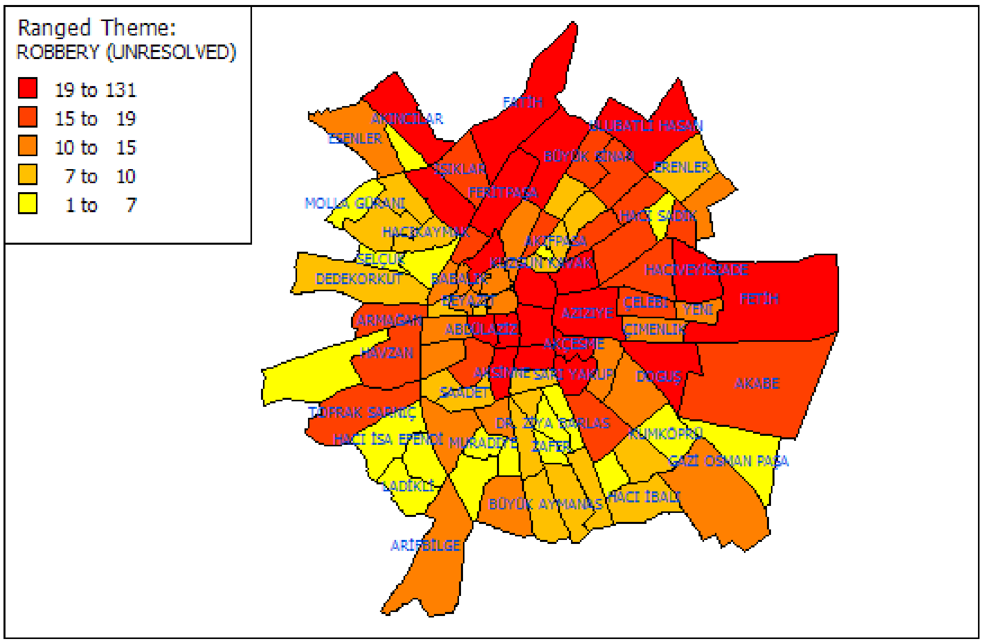


**Figure 17**: Ranged thematic with a database table as the data source.

Unlike the thematic map in the section 4.4.1.1.1., this time the thematic map displays the distribution of the crimes that are robbery *and* are unresolved. Moreover, Query Builder allows the analyst to create much more complex filter expressions.

As one can easily comprehend from the legend, districts with a higher crime count are displayed with darker colors. This way, it is easy to spot the districts having high robbery crimes.

### 3.4.1.1.3. Distribution Methods

When creating ranged thematic maps, there is a **Distribution Method** field in the **Ranges of Values Options** dialog in Figure 13. By this field it is possible to use different distribution methods, namely equal count, equal ranges and natural break. Using different methods may result in slight differences in shading.

**Equal Count** "has the same number of records in each range. To group 100 records into 4 ranges using Equal Count, the ranges are computed so that approximately 25 records fall into each range, depending on the rounding factor which is set. When using Equal Count (or any other range method), it's important to watch out for any extreme data values that might affect your thematic map (in statistics, these values are referred to as *outliers*) [16]."

**Equal Ranges "**divides records across ranges of equal size. For example, if a column has data values ranging from 1 to 100, when creating a thematic map with four equal size ranges, the ranges 1–25, 25–50, 50–75, and 75–100 are produced [16]."

**Natural Break** "creates ranges according to an algorithm that uses the average of each range to distribute the data more evenly across the ranges. It distributes the values so that the average of each range is as close as possible to each of the range values in that range. This ensures that the ranges are well-represented by their averages, and that data values within each of the ranges are fairly close together. MapInfo bases its Natural Break on the Jenks-Caspall Algorithm [16] [17]."

#### 3.4.1.1.4. Point and Linear Ranged Thematics

In addition to the ranged thematic maps with regional layers, it is also possible to create point and line based ranged thematic maps. For point layers, a thematic map showing points with different colors or varying sizes can be generated. Likewise, for line layers a thematic map displaying lines with different colors or varying widths can be created. The latter has already been implemented in section 4.4.3.3 (Linear Hotspot).

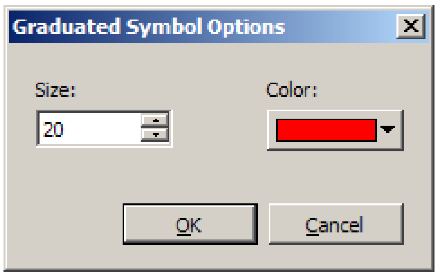
### 3.4.1.2. Graduated Symbols

Another type of a thematic map that Crime Analyzer implements is graduated symbols. With graduated symbols, data typically is depicted as proportionally sized symbols according to the values of a numerical column. Hence graduated symbols are sometimes called *proportional symbols*.

A good example for graduated symbols may be a map displaying the population density of cities. This thematic map may display the city centers with high population by big red circles, while displaying the ones with low population by small red circles.

At a first glance, one can think that graduated symbols can be used only for point layers. But this is not the case. They can be used for layers including any type of objects such as lines or regions.

The steps to produce a graduated symbols map in Crime Analyzer are the same except for the thematic options dialog. In other words, the analyst first selects the thematic data source, and then continues. Regardless of the theme data source which is selected, the last step is to set the graduated symbol options, which is done by means of the **Graduated Symbol Options** dialog in Figure 18.


Graduated Symbol Options

Size: 20

Color: 🟥

**Figure 18**: Graduated symbol options.

There are two options to set in this dialog: Symbol size and symbol color. Symbol style will be a filled circle. Clicking the OK button will create and display the graduated symbols map as in Figure 19.

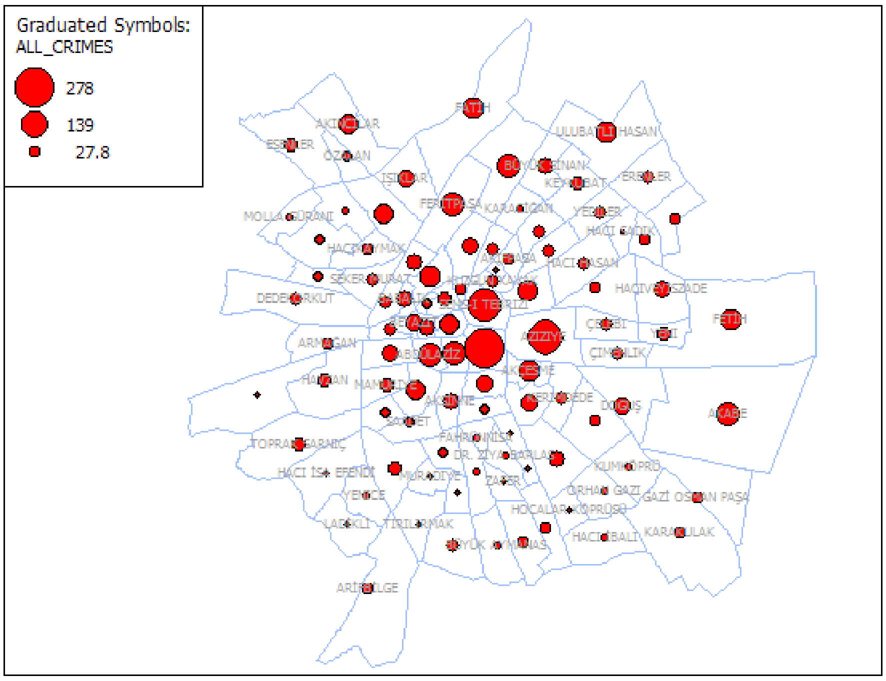


**Figure 19**: Graduated symbols showing the crimes.

The map displays clearly that crimes are mostly committed at the central locations of the city.

### 3.4.1.3. Individual Values

Individual values are the third thematic type in Crime Analyzer. In an individual values thematic map, the unique values in the thematic column are determined and each object on the map is displayed with particular colors or symbols based on these unique values. For instance, showing different crime types with different colors would be a good example of individual values theme. Both numerical and nominal values can be used as individual values. Figure 20 depicts four different crimes.

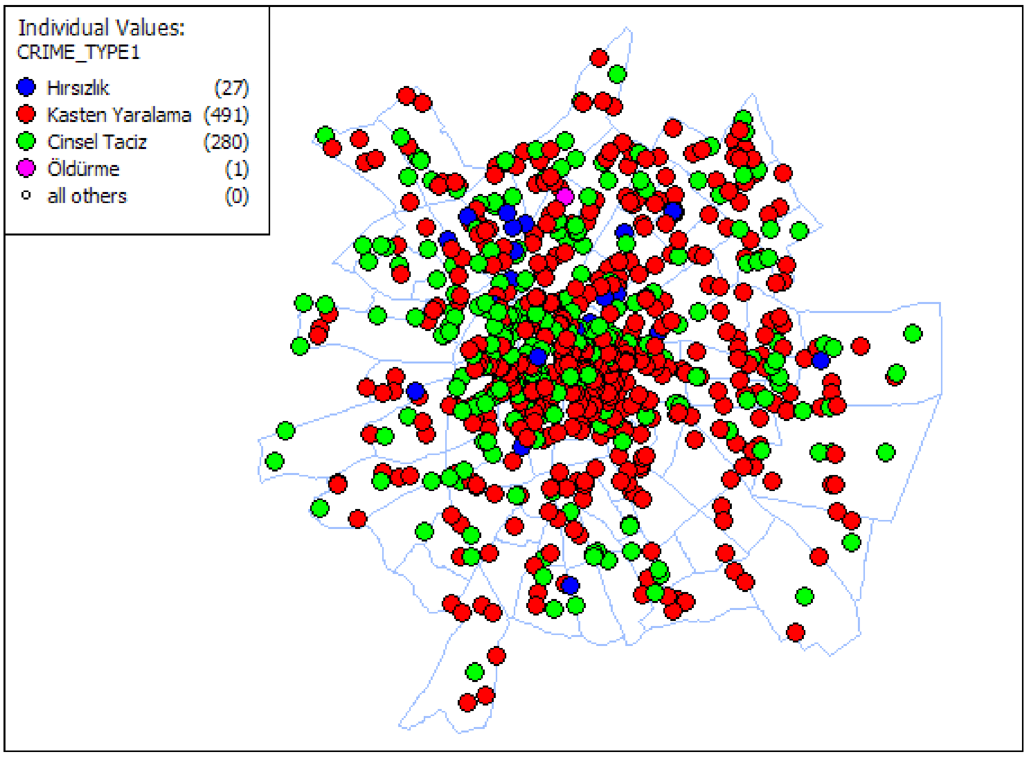


**Figure 20**: Individual values displaying different types of crimes.

Using the thematic map in Figure 20, it is possible to determine the theft locations since there are some *clusters* with blue points, which spot the theft locations. Also the map says that malicious injuries and sexual offences are widely spread across the city.

### 3.4.1.4. Bar Chart

The previous thematic types, namely ranged thematic, graduated symbols and individual values all have *single* thematic variable. Hence they are also called *one-variable thematic maps*. Sometimes it may be useful to be able inspect more than one thematic variable per map object at a time. Bar chart thematic maps allow the analyst to achieve this. When a bar chart thematic is created, for each map object bars with the number of thematic variables are created at the centroid of the map objects. This not only allows the analyst to compare the heights of the bars in a single object but also allows inspecting the same variable among all the charts belonging to other map objects. Therefore, bar charts are called *multi-variable thematic maps*.

The symbols of the bars may be in different styles such as multi-bar chart or stacked bar chart. Crime Analyzer uses stacked bar charts as default.

To produce a bar chart thematic map in Crime Analyzer, the analyst should specify the MapInfo table and the thematic variables, or thematic columns, in the **Thematic Table and Columns** dialog in Figure 21.

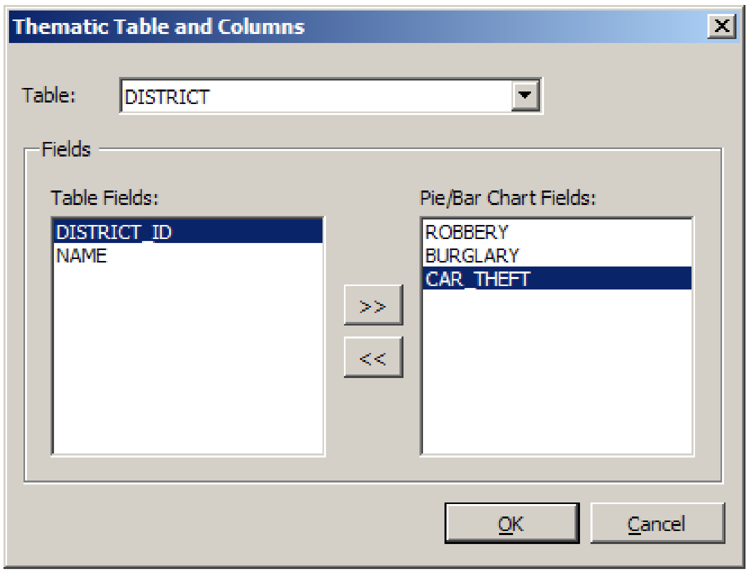

Thematic Table and Columns

Table: DISTRICT

Fields

Table Fields: DISTRICT_ID, NAME

Pie/Bar Chart Fields: ROBBERY, BURGLARY, CAR_THEFT

**Figure 21**: Setting the table and thematic columns of a bar chart.

In the **Thematic Table and Columns** dialog, selecting the MapInfo table from the Table field lists the columns of the table in the Table Fields list. Then the analyst should decide and select the thematic columns and then transfer them to the right, that is, Pie/Bar Chart Fields list. Clicking the OK button will create the thematic map in Figure 22.

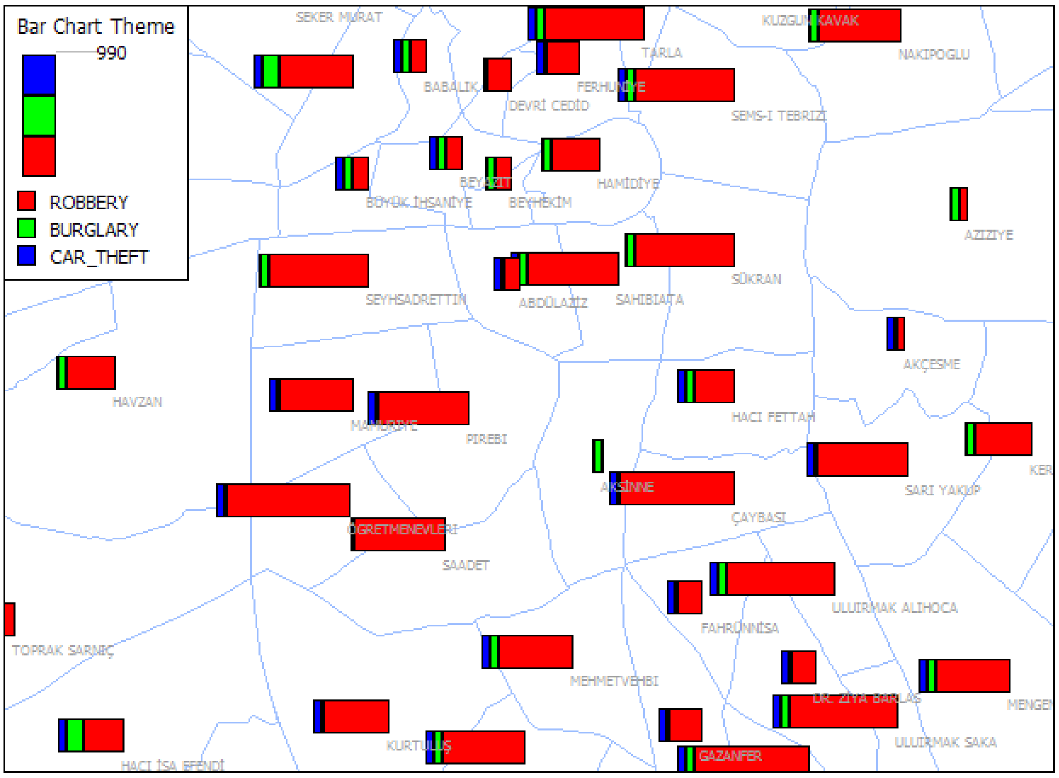


**Figure 22**: Bar chart thematic map showing the robbery, burglary and car thefts.

As the map suggests, the bar chart theme has the capability to display 3 thematic variables, namely *robbery*, *burglary* and *car theft*, at the same time.

### 3.4.1.5. Pie Chart

Pie chart thematic is much like the bar chart thematic. It is a *multi-variable thematic map*, too. Instead of displaying bar charts, it shows pie charts at the map objects' centroid, allowing making comparisons between the slices of the pie. In other words, pie charts present a way to compare the pieces of a whole.

Creating a pie chart thematic in Crime Analyzer is also the same as creating bar charts. Hence the full steps will not be repeated here. It involves selecting a table and thematic columns. A typical pie chart thematic map may appear as in Figure 23.

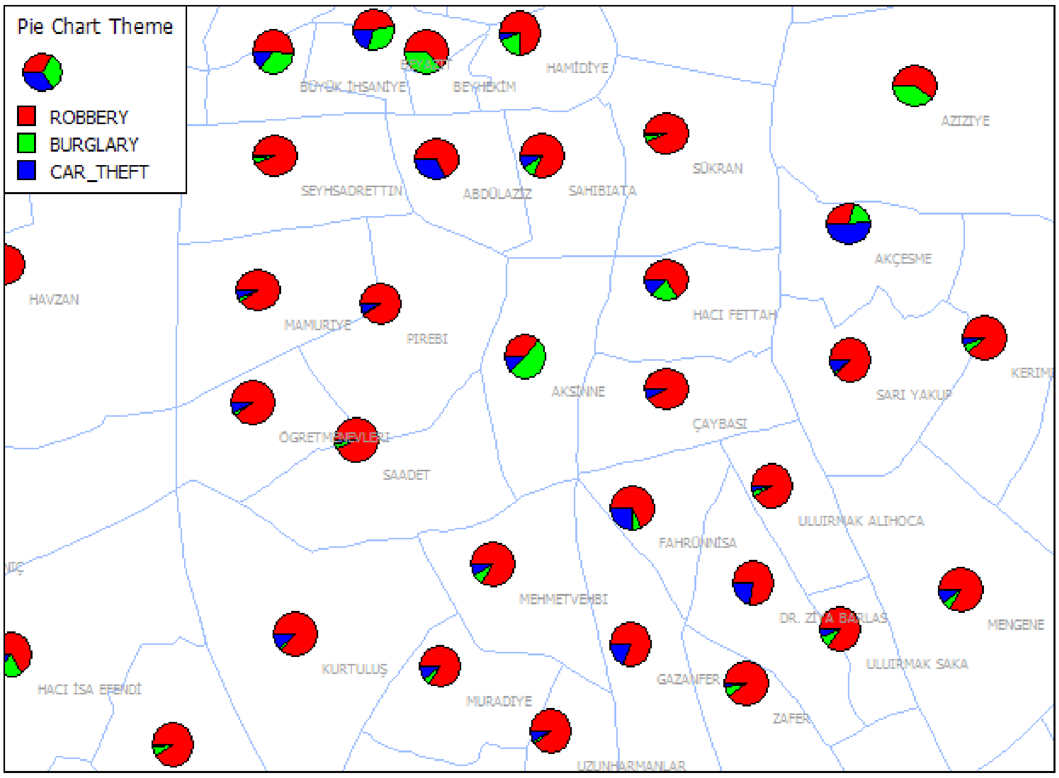


**Figure 23**: Pie chart thematic map showing the robbery, burglary and car thefts.

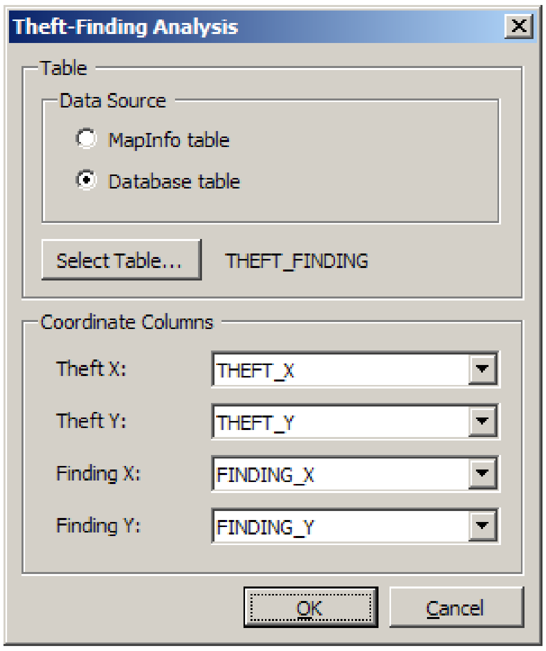
As the map suggests again, the pie chart theme has the capability to display 3 thematic variables, namely *robbery*, *burglary* and *car theft*, at the same time.

### 3.4.2. Direction Analyses

#### 3.4.2.1. Car Theft-Finding

Since car theft is one of the common crimes types, there is a dedicated module in Crime Analyzer for that crime type. It gets the required coordinate values from a data source and then draws arrows from theft locations to finding locations. In this way, it can be possible to determine the probable locations of criminal organizations that steal and break cars into pieces.

Using the **Theft-Finding Analysis** dialog in Figure 24, the analyst can display the theft and finding locations on the map.


Theft-Finding Analysis

Table

Data Source

MapInfo table [✓ Database table]

Select Table... THEFT_FINDING

Coordinate Columns

Theft X: THEFT_X

Theft Y: THEFT_Y

Finding X: FINDING_X

Finding Y: FINDING_Y

OK Cancel

**Figure 24**: Theft-Finding Analysis dialog.

In the **Theft-Finding Analysis** dialog, the analyst first selects the data source and the table including the coordinates. When the table is selected, the columns of the table are listed in the four fields in the Coordinate Columns group. Selecting the right columns and then pressing the OK button displays the analysis result as in Figure 25:

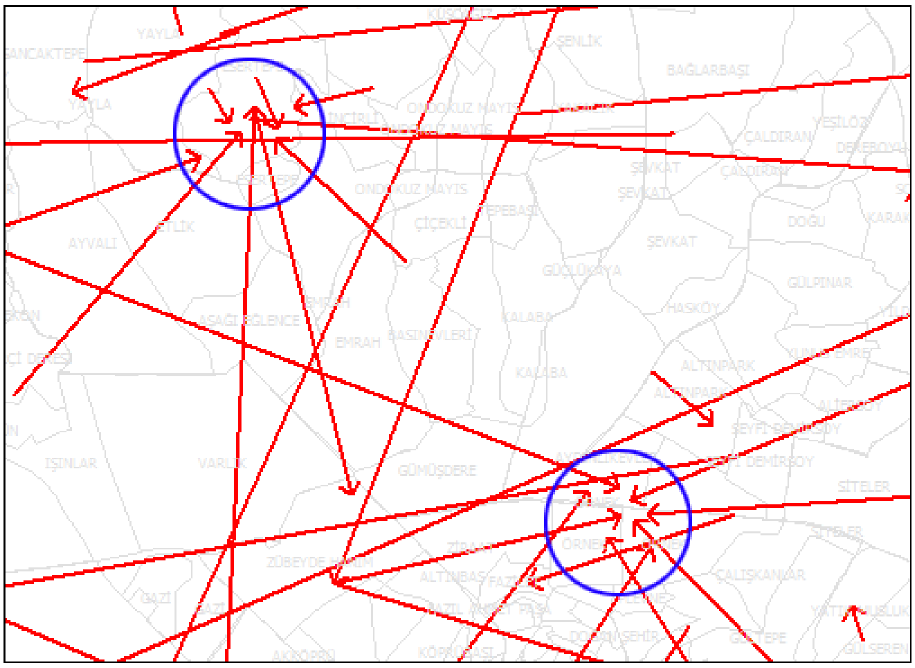
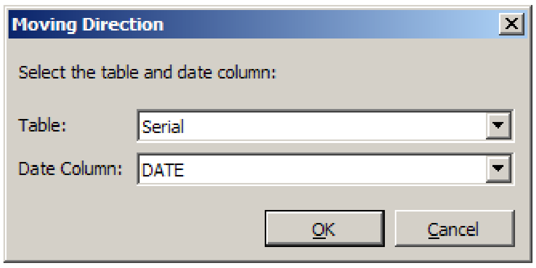


**Figure 25**: Car finding locations.

The analysis result in Figure 25 can suggest a strong probability that there are at least two crime organizations within the blue circles since there are apparent car *collection locations* there.

### 3.4.2.2. Moving Direction

Sometimes it may be important to be able to see spatial *shifts* in serial or organized crimes. To display these shifts Crime Analyzer has a module. It expects a point MapInfo table and a date column in this table as shown in Figure 26.


Moving Direction

Select the table and date column:

Table: Serial

Date Column: DATE

**Figure 26**: Selecting the crime table and date column.

Then Crime Analyzer orders the crime points by date and draws lines between the points by joining them. The result is a map in Figure 27.

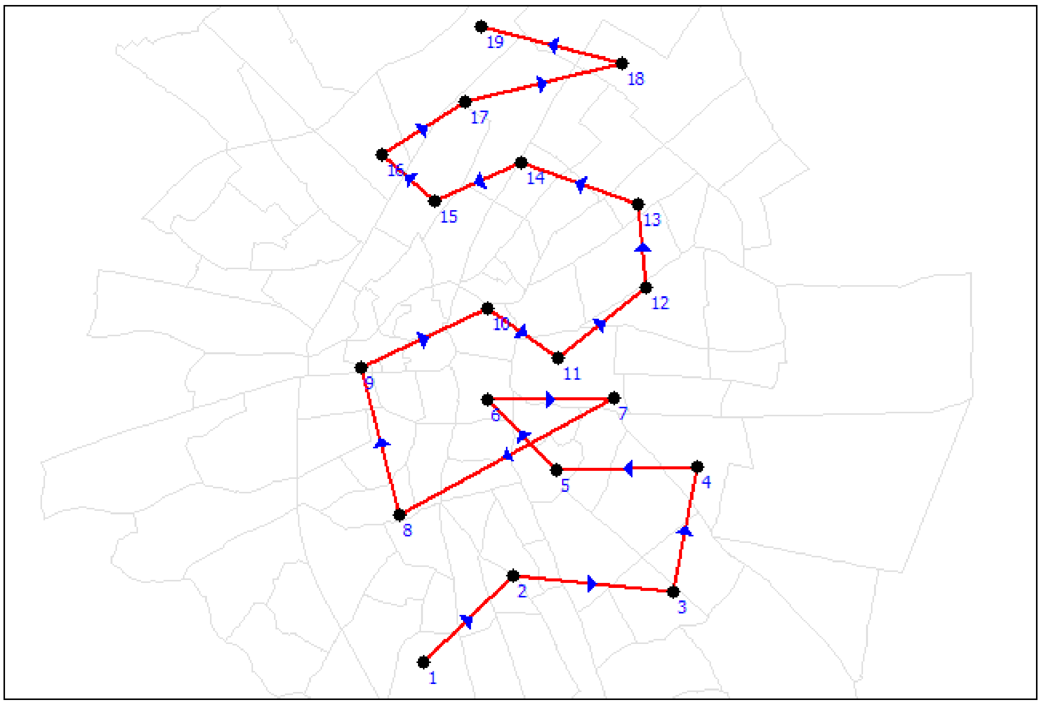


**Figure 27**: Determining the moving direction of the offender.

As Figure 27 offers, it is not difficult to comprehend that the serial crimes are *shifting to the north* over time. This way, it can be possible to narrow and estimate the potential new crime area.

### 3.4.3. Density Analyses

#### 3.4.3.1. Dot Density

In dot density maps, dots are used to represent the data value related with a region. Dot density maps are especially useful for displaying data where a single dot represents a large number of a quantity such as population or manholes. The total number of dots is reduced to a fraction of the actual point count. For example, each 100 points on a table may be represented as 1 dot. This produces a much cleaner map.

Dot density maps are also useful for crime mapping since they can clearly spot the regions with a high crime density with raw crime data. To demonstrate this, one can consider the raw crime data in Figure 28.

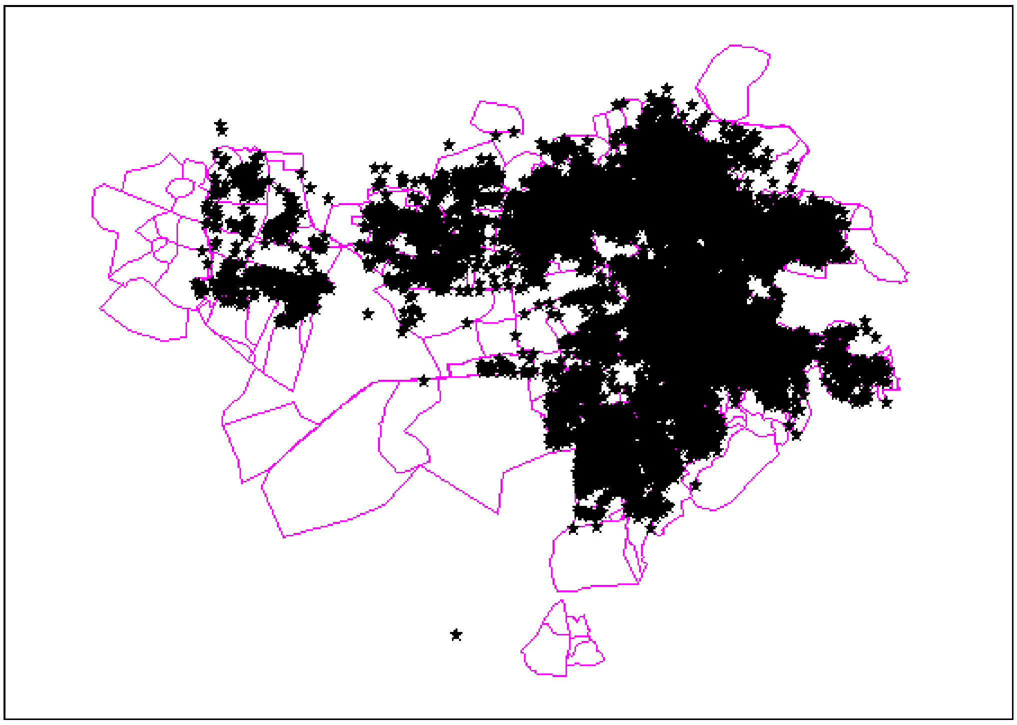
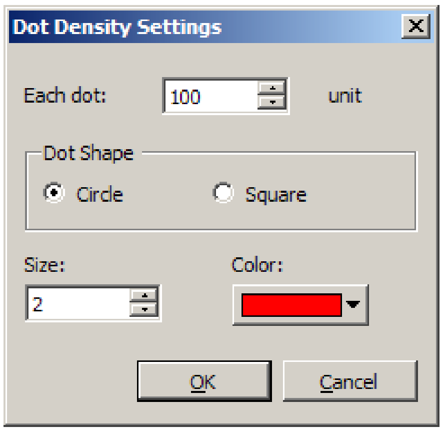


**Figure 28**: Displaying raw crime data.

As the map in Figure 28 suggests clearly, displaying raw crime data alone does not help the analyst too much because of the crowded points. Right at this point, dot density maps come into play.

Creating dot density maps in Crime Analyzer is the same as creating a ranged thematic. The only difference is setting the options in the **Dot Density Settings** dialog in Figure 29.


Dot Density Settings

Each dot: 100 unit

Dot Shape

[✓ Circle] Square

Size: 2

Color: 🟥

**Figure 29**: Dot density settings.

In the **Dot Density Settings** dialog, one can specify the unit count that each dot will represent, the dot shape, the dot size and the color. Hitting the OK button will generate the dot density map in Figure 30.

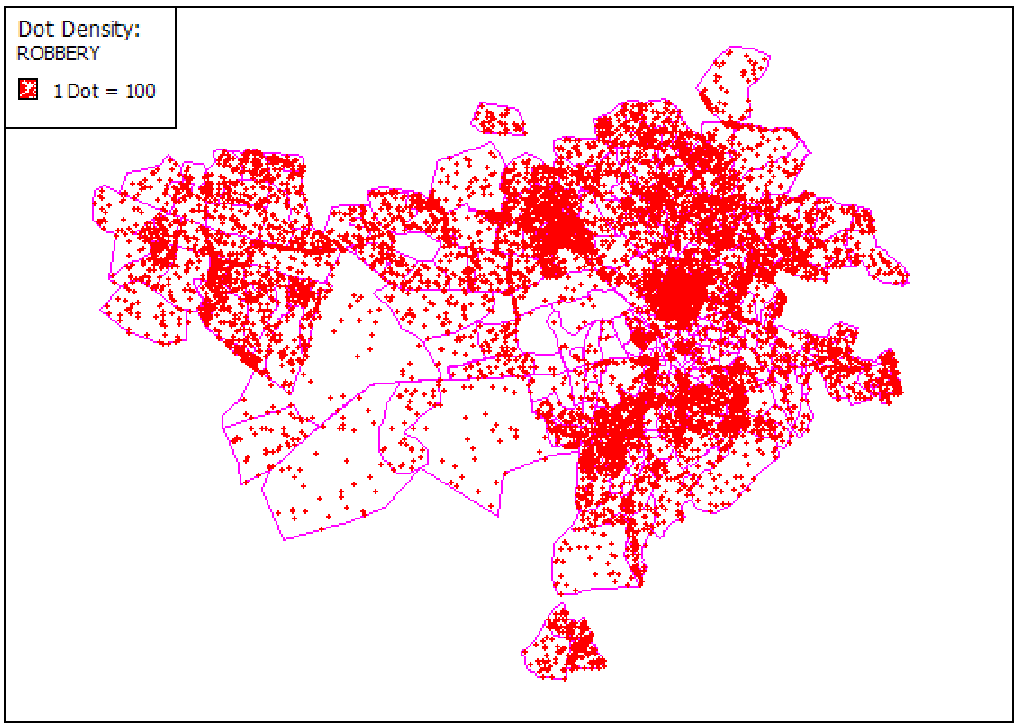


**Figure 30**: Representing each 100 crimes with 1 dot in a dot density map.

As the map suggests, dot density maps are particularly useful in showing the raw data in a wide area such as a city. It is easy to determine the regions of robbery with a high density at a first look.

### 3.4.3.2. Thematic Grid

Thematic grid maps are different from the other thematic types in that the others are based on vector layers. Thematic grids, on the other hand, are created using an interpolation point data from a source table. To do this, first a grid file is created from the interpolated data and then this file is shown as a raster layer in the map. The result is typically a raster image with continuous color transition.

Good examples of thematic grids are displaying rainfall amounts or climate changes in weather forecasts. Thematic grids are also suitable to *hot spot* the peak crime locations on the map. So they are implemented in Crime Analyzer.

Steps to generate thematic grids in Crime Analyzer are the same as to generate a ranged thematic. The thematic values can be pulled from a database table or simply a numerical MapInfo table column can be used as thematic variable. A thematic grid may appear as in Figure 31.

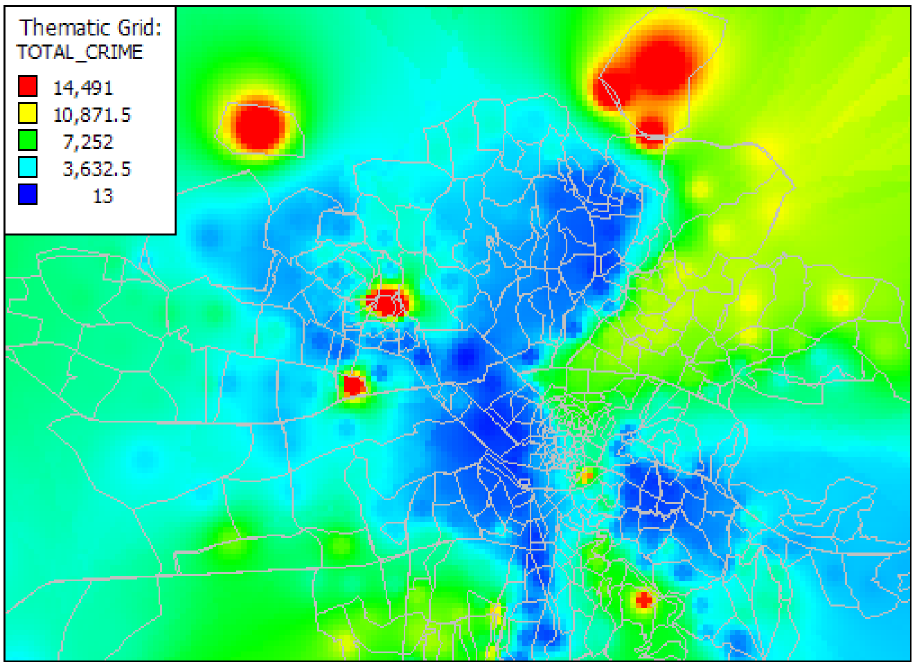


**Figure 31**: Thematic grid depicting hot spots.

As can be comprehended from the legend in Figure 31, the *red spots* highlight the *peak crime areas* in the city. The areas with cold colors like blue and green show the minimal crime locations. Another point is that, the result is not a vector layer, but a continuous raster image as mentioned above.

### 3.4.3.3. Linear Hotspot

Linear hot spot is a slightly different thematic type since rather than points or regions, they involve *line* objects. In other words, it "measures the risk distribution of crime along a linear network by calculating the rate of crimes per section of road [18]." The crimes can be, for instance, car thefts or incidents occurring in bus trips along a street. Considering and inspecting the possible relation between crimes and linear networks can give an exclusive point of view that other thematic types cannot provide.

Linear hot spots are actually a kind of ranged thematic since the objects are colored according to a column such as incident count of a street.

The steps to generate a linear hotspot map is exactly the same as to generate a ranged thematic and involves specifying first the data source and thematic column and then the ranged thematic options. A typical linear hot spot map can appear as in Figure 32.

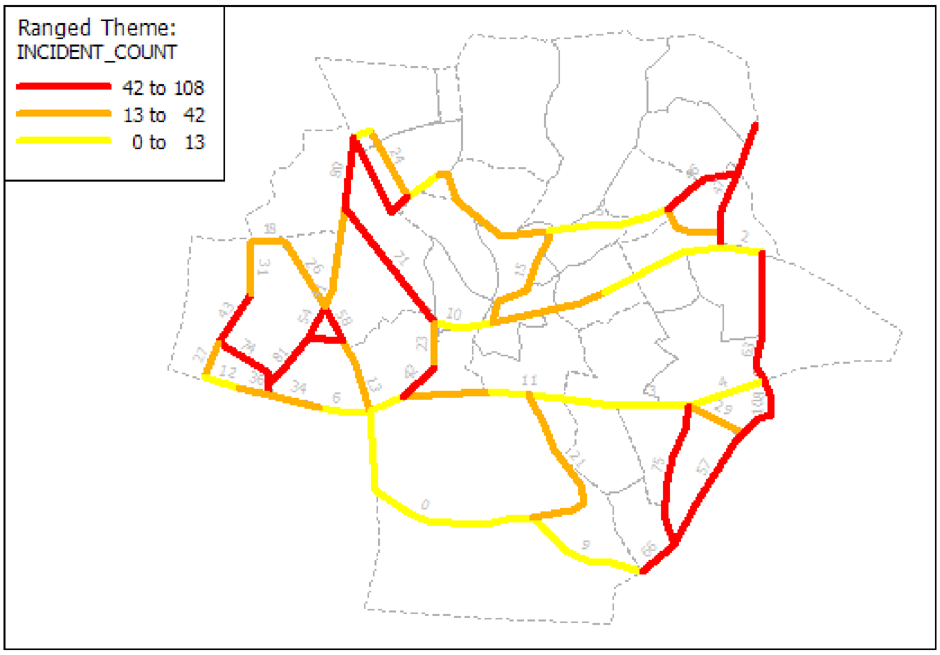


**Figure 32**: Linear hot spots along streets.

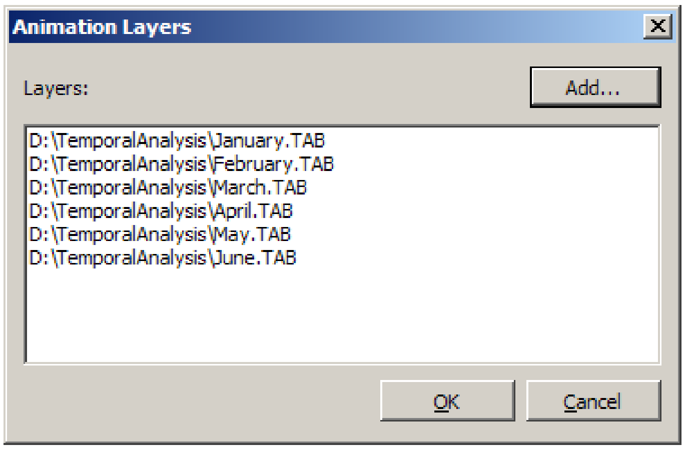
As the legend in Figure 32 points out, the streets with red represent the top crime lines. If the crime here is assumed to be car theft, then it can be inferred that the streets with red lines certainly *requires more security measures* by means of additional police forces or cameras.

### 3.4.4. Temporal Analyses

Besides the spatial analysis tools like thematic or direction analyses, Crime Analyzer also presents non-spatial tools. Temporal analyses let the analyst to investigate and compare particular crimes at particular time intervals. They are superior for spotting the date or hour intervals including peak crime rates.

#### 3.4.4.1. Temporal Distribution

Analyzing only some snapshots of time may not be sufficient for a crime analyst to determine the crime patterns or future trends. Sometimes it is necessary to see the crime distribution or change trend by inspecting the crime data in different time points. For example, displaying the locations of a particular crime for each six months may spot the changes in the distribution of the crime with time. Temporal distribution first requires that the analyst select the tables with the **Animation Layers** dialog, as in Figure 33.



```
D:\TemporalAnalysis\January.TAB
D:\TemporalAnalysis\February.TAB
D:\TemporalAnalysis\March.TAB
D:\TemporalAnalysis\April.TAB
D:\TemporalAnalysis\May.TAB
D:\TemporalAnalysis\June.TAB
```

**Figure 33**: Specifying the tables for temporal distribution.

Hitting the OK button in the **Animation Layers** dialog will open the **Incident Animation** form in Figure 34.

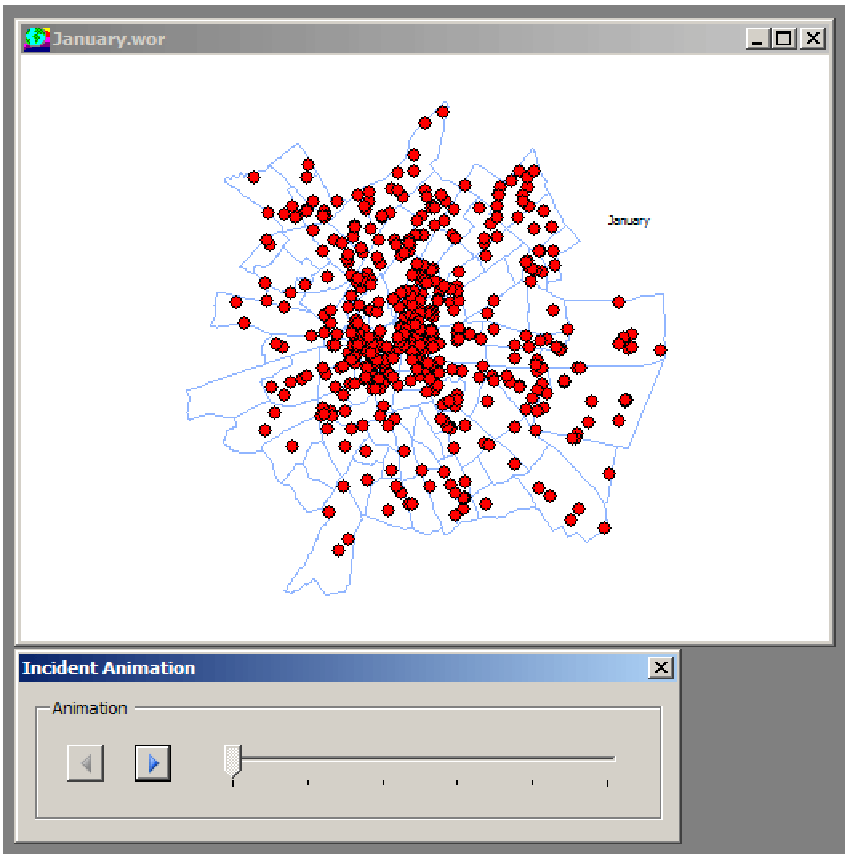


**Figure 34**: Incident animation displaying successive months' crimes.

**Incident Animation** form provides *next* and *previous* buttons that provide the functionality to display the next or previous month's crimes on the map. While moving with the months, the position of the track bar on the right is updated accordingly. This way, the analyst can easily see the *crime distribution with time* and can compare the crimes belonging different months.

### 3.4.4.2. Creating Animation

It is possible to create animations from a collection of map images. This will make sharing the temporal analysis results much easier. Another benefit is that the animated gif images can be displayed in a web page, for example.

Here it will be suitable to note that Crime Analyzer has the capability to save a map as an image. To do this, **Export Map Image** button on the map toolbar with icon can be used. When clicked, this button pops up the **Image Export Options** dialog in Figure 35.

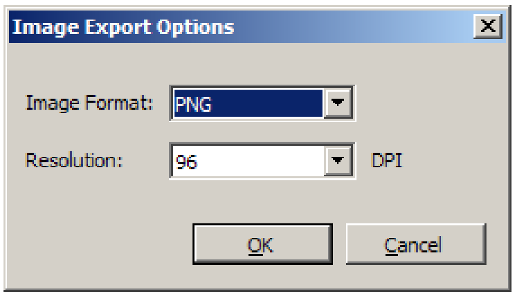


Image Export Options

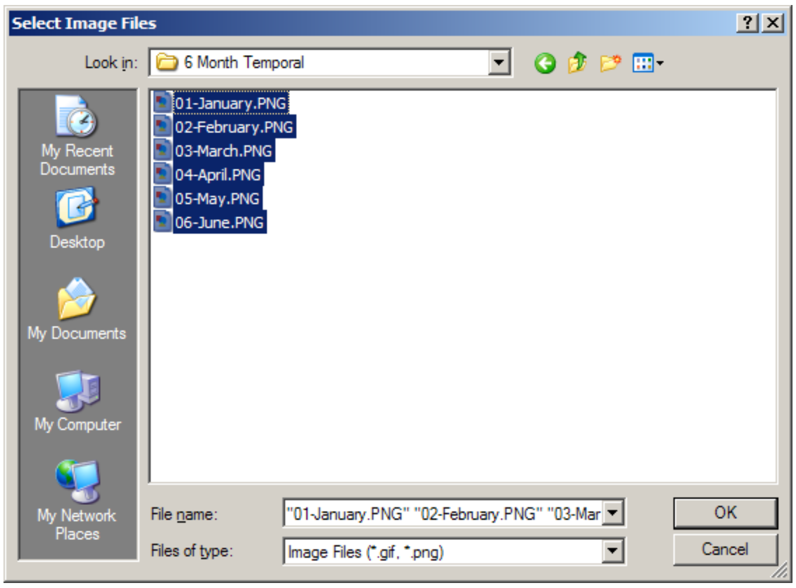
Image Format: PNG

Resolution: 96 DPI

**Figure 35**: Image Export Options dialog.

Clicking the OK button in the **Image Export Options** dialog saves the current map window as image with the selected options.

To create an animation file, it is sufficient to browse and select the input images as shown in Figure 36.


Select Image Files

Look in: 6 Month Temporal

01-January.PNG
02-February.PNG
03-March.PNG
04-April.PNG
05-May.PNG
06-June.PNG

File name: "01-January.PNG" "02-February.PNG" "03-Mar

Files of type: Image Files (*.gif, *.png)

**Figure 36**: Selecting the files to create animation.

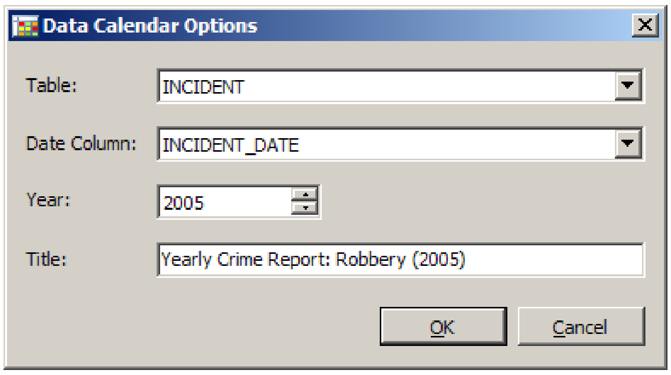
Hitting the OK button in Figure 36 creates and opens the animated gif file. While implementing this module, an open source library from the CodeProject site has been used [19].

### 3.4.4.3. Data Calendar

As well as presenting various map tools, Crime Analyzer provides non-spatial analysis types like data calendar and data clock. The reason is that temporal analyses are also important in crime analysis; *relating the time domain with the spatial domain* can give exclusive and well directed clues for crime trends and patterns. For example, the analyst can perform a thematic analysis on the map, and then see the same crime records in a data calendar or data clock to determine the peak dates and hours of the crimes.

In some resources data calendar is called “data clock” but it will be more appropriate to call it “data calendar” since its output is much like a yearly *calendar* as shown in the Figure 38.

Generating a data calendar requires connecting to the database, specifying the source incident table, the date column in the table, the report year and optionally the report title as in the **Data Calendar Options** dialog in Figure37.



| | |
|---|---|
| Table: | INCIDENT |
| Date Column: | INCIDENT_DATE |
| Year: | 2005 |
| Title: | Yearly Crime Report: Robbery (2005) |

**Figure 37**: Data Calendar Options dialog.

Hitting the OK button in the **Data Calendar Options** dialog will display the data calendar with specified options as in Figure 38.

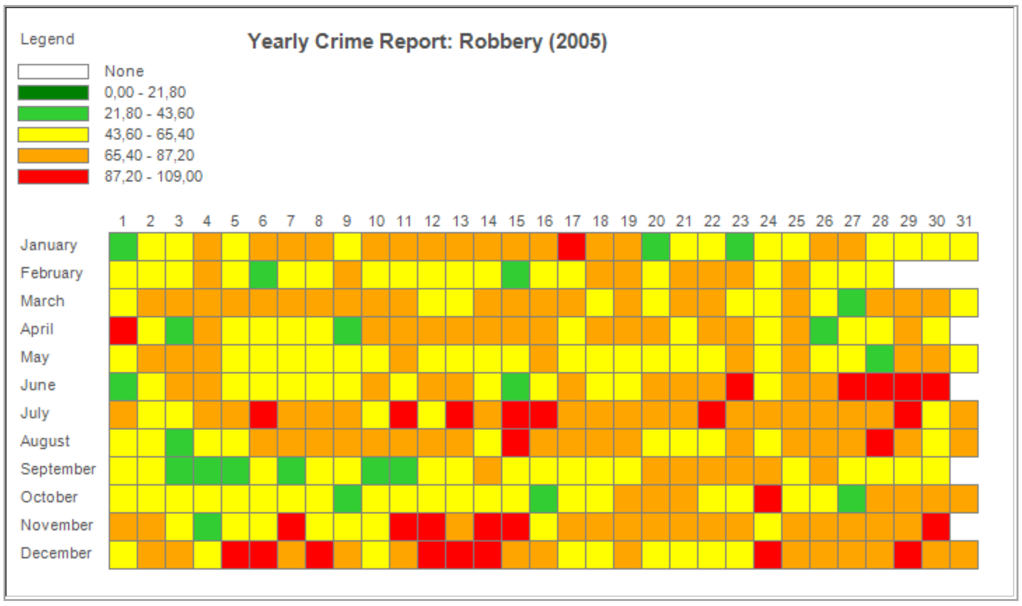


**Figure 38**: Data calendar.

Data calendar is a powerful data visualization tool that can summarize the crime distribution for the whole year in a single picture.

Determining the peak time intervals of particular crimes is crucial in crime analysis. Generating and inspecting a data calendar can greatly contribute this purpose since it can spot the dense time of particular crimes as can be seen with the *red clusters* in Figure 39. For example, as the last three data calendars in Figure 39 suggest, there is a *stable robbery trend* in December. There are also a few red clusters in the last data calendar belonging to the months June, July and December. These are the times that probably must be carefully examined.

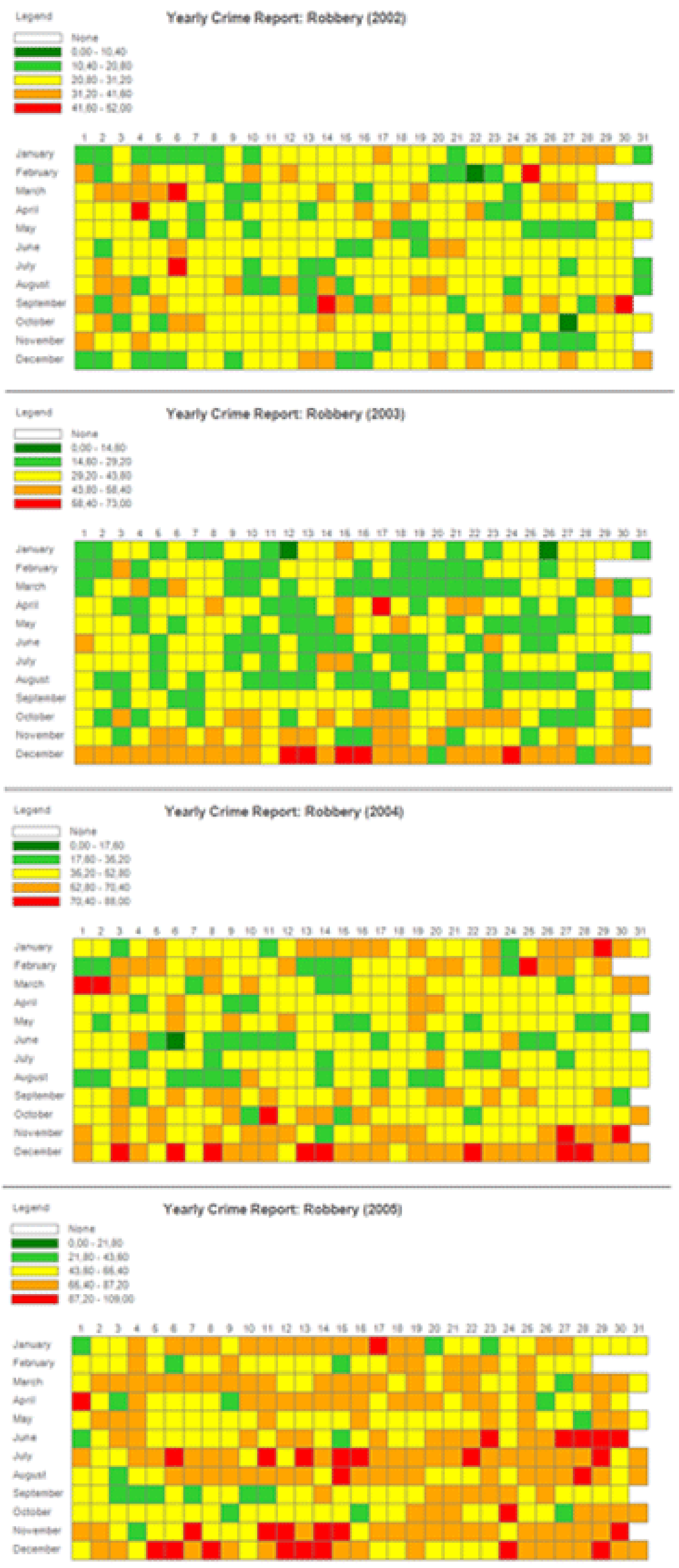


**Figure 39**: Data calendars comparing the crime times of four successive years.

Since data calendar is a 2-dimensional chart, distribution of crimes can be inspected in both vertical and horizontal axes. Inspecting the data calendar vertically means searching *in which months* there are crimes above the average. Inspecting horizontally means, on the other hand, looking at the *days of months* for the crimes, which may have a result that a particular crime may steadily be committed in the middle of all the months, for example. This may be relevant with the corporate salary payment days of the state, for the robbery crimes for instance. In situations like those, data calendars point out that there are some dates that require special attention.

The data calendars can also display the increases or decreases of the crimes over time. For instance, the regular and stable increase in robbery incidents between 2002 and 2005 can be seen from Figure 39.

### 3.4.4.4. Data Clock

Data clocks are much like data calendars in that they display the crime distribution over a time interval. However, rather than showing the months, they display the *days* and *hours* of crimes over a particular week. In other words, they present the crime distribution in the time domain in a narrower interval. This means they can denote the peak days and hours of a specific crime.

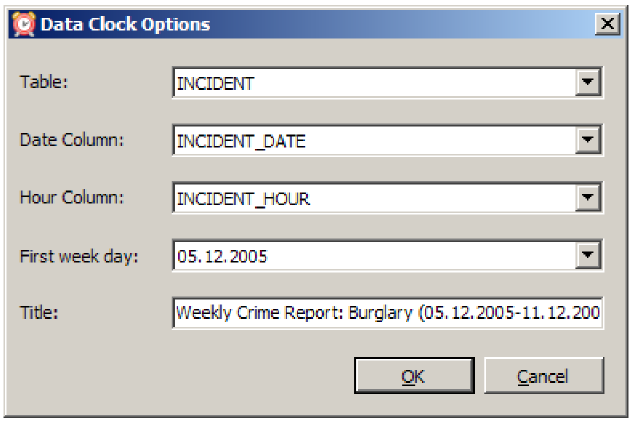

Data Clock Options

Table: INCIDENT

Date Column: INCIDENT_DATE

Hour Column: INCIDENT_HOUR

First week day: 05.12.2005

Title: Weekly Crime Report: Burglary (05.12.2005-11.12.200

**Figure 40**: Data Clock Options dialog.

Generating a data clock in Crime Analyzer is very similar to generating a data calendar. The only difference is that data clock requires additional information, namely the hour column of the incident table as in Figure 40. Clicking the OK button will generate and display the data clock as in Figure 41.

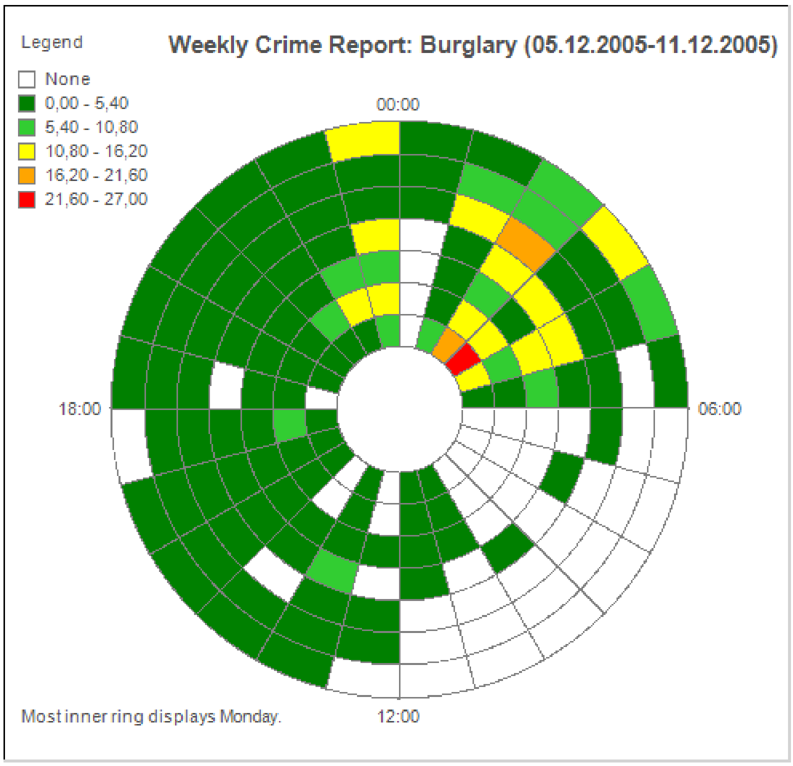


**Figure 41**: Data clock.

Data clocks are much the same as regular clocks. The sole difference is that data clocks divide a circle into 24 pieces rather than 12. Each slice corresponds to a single hour. The hours start from 00:00 at the top and increases clockwise, as in Figure 41. On the other hand, the nested 7 rings represent the 7 days of a week, the most inner ring being Monday.

Like data calendars, data clocks allow 2-dimensional analyses. The clockwise sweeping represents hours of a day and going from the inner to the outer rings represents the days of a week. This means data clock has the capability to show the crime distribution with respect to days or hours.

Generating multiple data clocks and comparing the crime distributions can also give superior results. For example, Figure 42 suggests that the burglaries in a successive four week period are regularly and steadily committed between 22.00 and 04:00 AM. With this information, the analyst may conclude that there must be *more patrol forces* during this time interval. With regard to day dimension, the clock in Figure says that there are no particular days that contain peak crimes.

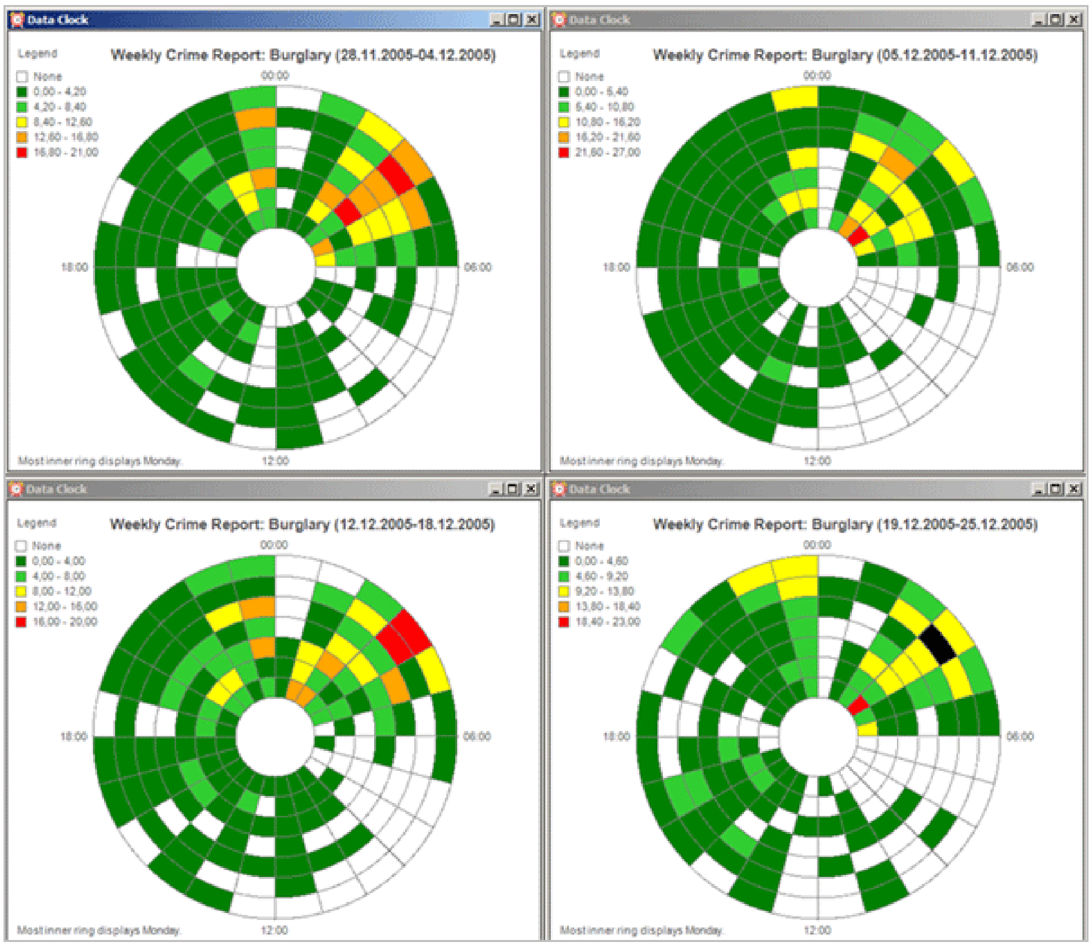


**Figure 42**: Data clocks comparing the crime times of four successive weeks.

### 3.4.5. Chart Analyses

#### 3.4.5.1. Bar Chart

Another non-spatial analysis type is bar chart. Since using charts is a common way to visualize data, Crime Analyzer allows the analyst to generate bar charts. Generating a bar chart first requires connecting to a database and then selecting the source table and name column. Like the previous analysis types, chart analyses can utilize the Query Builder dialog in Figure 16 to create flexible queries against the source table. In this way, the analyst can generate various filters as in Figure 43.

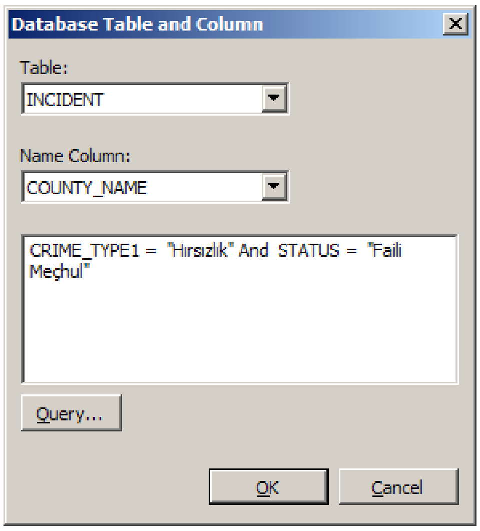

Database Table and Column

Table: INCIDENT

Name Column: COUNTY_NAME

CRIME_TYPE1 = "Hırsızlık" And STATUS = "Faili Meçhul"

**Figure 43**: Generating a bar chart.

The **Query Builder** dialog in Figure 16 can be opened by clicking the Query… button in Figure 43. After specifying the source table, name column and the filter clause, clicking the OK button will pop up the **Bar Chart** dialog in Figure 44 to specify the chart title:

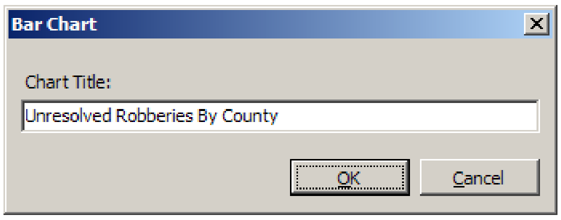

Bar Chart

Chart Title: Unresolved Robberies By County

**Figure 44**: Setting the chart title.

Hitting the OK button in Figure 44 generates and displays the bar chart as in Figure 45:

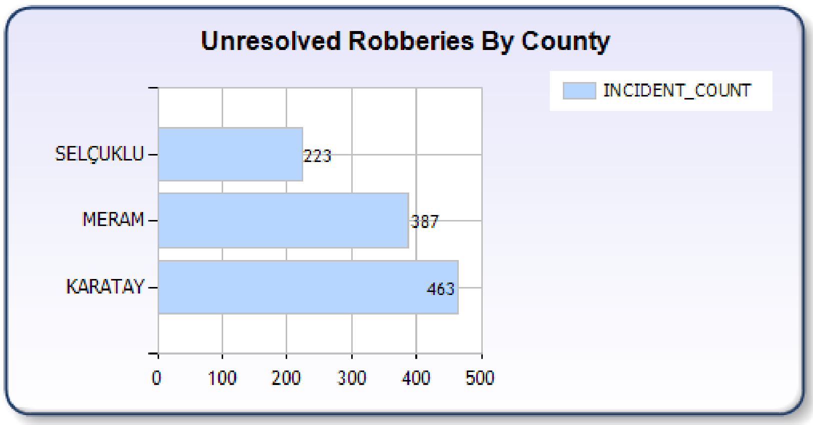


**Figure 45**: Bar chart displaying the rates of unresolved robberies by county.

In this way, bar charts can be generated with various flexible filters.

### 3.4.5.2. Pie Chart

Generating pie charts in Crime Analyzer is exactly the same as generating bar charts, that is, it involves specifying the source table, the name column, the query clause and then the chart title. A typical pie chart may be as in Figure 46:

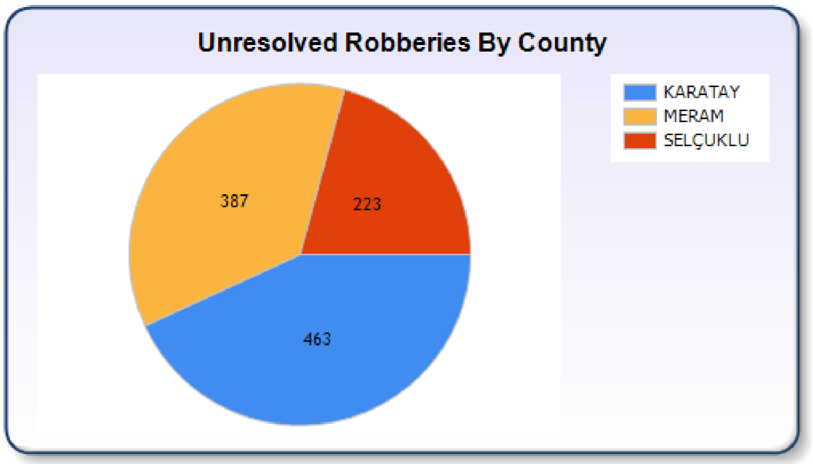


**Figure 46**: Pie chart displaying the rates of unresolved robberies by county.

# CHAPTER IV

# ARCHITECTURE

This chapter gives the high level design features such as development environment and framework, projects making up the system, project dependencies and functionalities of the project packages.

## 4.1. DEVELOPMENT ENVIRONMENT AND FRAMEWORK

Crime Analyzer has been developed using Microsoft Visual Studio (VS) 2010 and the .NET Framework 4.0. VS 2010 is a powerful development environment for enterprise, desktop or mobile projects. The programming language used in Crime Analyzer is C#. It is a modern, object-oriented, type safe and general purpose language created by Microsoft.

One alternative technology could be *Java Swing* but Java does not provide built-in Component Object Model (COM) support. Although it possible to use some bridge libraries for Java-COM interaction, it is not a practical and comfortable method compared with .NET. COM support is required to be able to display and control MapInfo maps within the application. As a result, C# and the .NET Framework seemed to be the best choice for developing spatial crime analysis software with MapInfo.

## 4.2. PROJECT COMPONENTS

Crime Analyzer is a desktop application consisting of two projects. The descriptions of the projects are as follows:

**CrimeAnalyzer**: This is the main project and is of type *Windows Forms Application*. It mainly consists of the UI components of the application. Examples of these components are the main program window, menus, navigation links and child windows hosting maps, tables and charts.

**MapInfoX**: This is a class library which provides all of the core functionality. With this library, it is possible to display maps, add layers to maps, create thematic or temporal analyses or pull data from a supported database.

There are also two external libraries, that is, DLL (Dynamic Link Library) files that are used by the CrimeAnalyzer project. These two DLL files are ready-to-use libraries from CodeProject as a MS Outlook-style navigation bar component [20]. This navigation bar appears on the left of the main window of Crime Analyzer.

Project dependencies for Crime Analyzer can be depicted as in Figure 47:

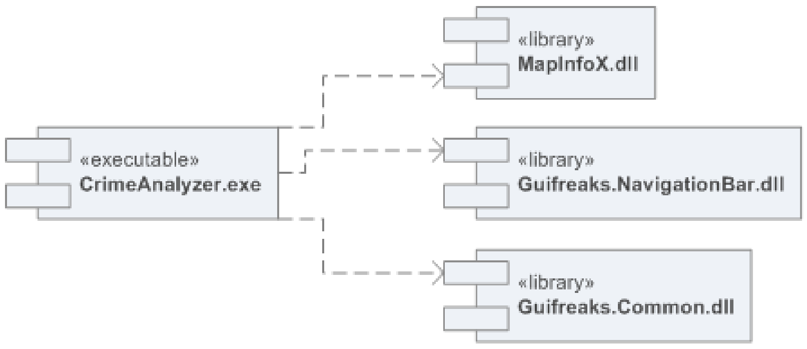


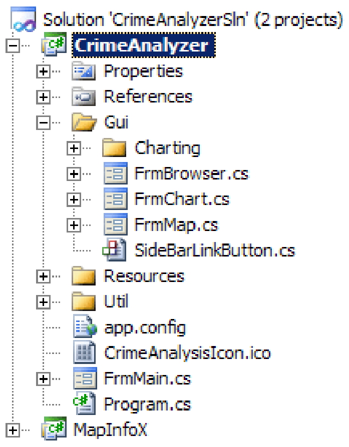
**Figure 47**: Project dependencies.

The components of the two projects will be examined in separate sections.

## 4.3. CRIMEANALYZER PROJECT COMPONENTS

Although being the main project in the solution, CrimeAnalyzer does not contain too much items in it. Its main function is being the entry point of the program (via the *Program* class) and hosting the main UI components like the main window, menus and child windows. It heavily depends on the MapInfoX project.

Figure 48 displays the CrimeAnalyzer project items in VS solution explorer:



```
Solution 'CrimeAnalyzerSln' (2 projects)
  CrimeAnalyzer
    Properties
    References
    Gui
      Charting
      FrmBrowser.cs
      FrmChart.cs
      FrmMap.cs
      SideBarLinkButton.cs
    Resources
    Util
    app.config
    CrimeAnalysisIcon.ico
    FrmMain.cs
    Program.cs
  MapInfoX
```

**Figure 48**: CrimeAnalyzer project items.

The main window is called FrmMain and it is an *MDI parent form* containing all of the child windows, menus and other UI items. Main folders, i.e. packages, of the project can be listed as follows:

- *Gui* folder under the project includes the child windows such as map windows, the chart components such as bar charts and the link button control that appears in the navigation bar.
- *Resources* folder contains the required resource files such as icons.
- *Util* folder includes the classes with utility methods performing common operations like logging.

## 4.4. MAPINFOX PROJECT COMPONENTS

MapInfoX is the core project providing the actual functionalities such as mapping, connecting to a database and performing both spatial and non-spatial analyses.

The name *MapInfoX* comes from the combination of the two product names of Pitney Bowes Software, namely MapInfo and MapX. MapInfo is a desktop GIS *program* and MapX is an object oriented mapping *component*. MapInfoX is a wrapper library that combines the power of MapInfo and the practicability of MapX and thus the project name was determined as MapInfoX.

Figure 49 displays the MapInfoX project items in VS solution explorer:

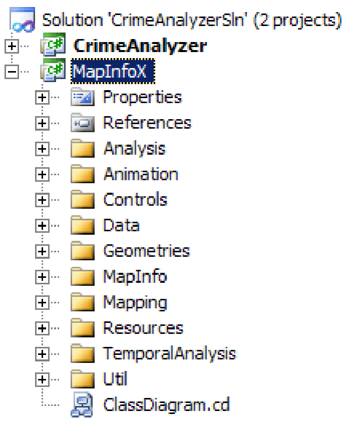


**Figure 49**: MapInfoX project items.

The project folders and their functionalities are listed below:

- *Analysis*: Contains some special analysis implementations.
- *Animation*: Contains the classes for generating animation files.
- *Controls*: One of the most important folders. Contains the map control, map toolbar, map statusbar, map tool buttons etc.
- *Data*: Provides the data access layer and connection classes for relational databases. Also contains catalog, table enumerator, table info and column info classes for spatial tables.
- *Geometries*: Contains the abstract base class for all geometry objects and the concrete implementing classes like point and rectangle. Also includes the distance structure and distance unit enumeration.
- *MapInfo*: Contains the low-level internal classes communicating directly with MapInfo. This communication involves the OLE (Object Linking and Embedding) callbacks to get map information such as current cursor coordinates and manipulating the map by sending commands to MapInfo.
- *Mapping*: Stands at the heart of MapInfoX. Includes the map class, different layer types, layers collection, map selection etc. Also contains the classes doing the actual work in generating the thematic analyses.
- *Resources*: Contains the required resource files such as icons.
- *TemporalAnalysis*: Contains the classes performing non-spatial time analyses such as data calendar and data clock.
- *Util*: Exposes utility classes such as color utilities and temporary file generator.

In addition to providing the main functionalities, MapInfoX converts the cumbersome MapBasic commands into map object methods. This will be detailed in the next chapter.

# CHAPTER V

# DEVELOPMENT OF THE SYSTEM

This chapter explores how the system has been implemented.

## 5.1. MAPPING FRAMEWORK

The system at first needs a powerful mapping framework in order to display maps and perform spatial analyses. MapInfo is full-featured desktop mapping software which is commonly used around the world. Hence it is suitable to utilize it in a crime analysis and mapping application.

MapInfo is currently produced by Pitney Bowes Software (Formerly MapInfo Corp.) The actual name of the product is *MapInfo Professional* but it is commonly referred to as just *MapInfo*. We also use this convention and MapInfo means MapInfo Professional in this study.

For the most part, MapInfo is used for mapping and spatial analysis. MapInfo makes it possible to create, store, manipulate, visualize, analyze, interpret and output both spatial and non-spatial data.

Also there is a Visual Basic-like programming language, namely *MapBasic*, to customize MapInfo. Using MapBasic, it is possible to write custom applications running inside MapInfo. Another benefit of MapBasic is the capability to automate repetitive tasks such as batch geoprocessing operations. Two examples of these operations can be large data conversions or geodatabase compaction. MapBasic also allows developers to programmatically control MapInfo through integrated mapping, which is the next and important topic to focus.

## 5.2. INTEGRATED MAPPING

Integrated mapping allows a client application to programmatically control and manipulate MapInfo using MapBasic commands. In fact, it is a type of IPC (Inter-

Process Communication). This communication is performed via either OLE automation or DDE (Dynamic Data Exchange). DDE is a relatively old technology and has some restrictions compared with OLE automation. OLE automation, on the other hand, is a newer and more stable technology and it is the recommended way for integrated mapping by the vendor. Hence, MapInfoX uses this method when implementing integrated mapping.

Integrated mapping applications are written with external programming languages, i.e. languages other than MapBasic. These languages *must* provide COM support since the MapInfo process will be added as a COM reference to the application. The most commonly used languages for integrated mapping are Visual Basic, Delphi, C++, C# and Visual Basic .NET.

In an integrated mapping application, every UI component except for the map itself belongs to the client application. For example, in Figure 50, the window, the toolbar and the buttons all belong to the client (C#) application. The map with the white background, however, is a MapInfo map which is re-parented by a window of the C# application. One exception to this rule is that some MapInfo dialogs such as *Layer Control* may be displayed within the application.

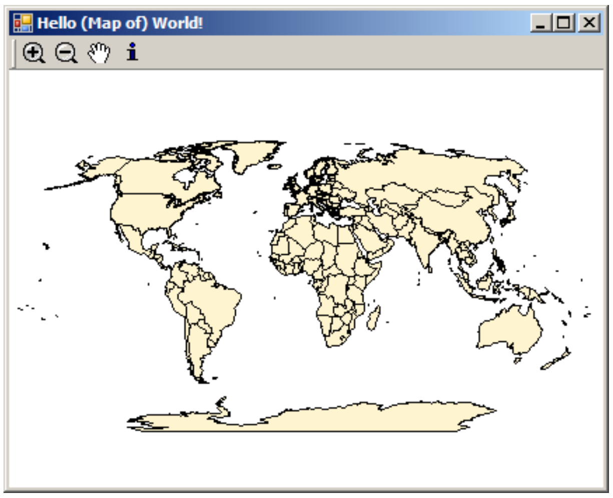


**Figure 50**: A *Hello (Map of) World* application with integrated mapping.

Typically the map is hosted within a container .NET component such as panel or picture box. When the application starts, a new MapInfo process is created silently in the *background* and the map is shown in the container component as mentioned above. In other words, MapInfo splash screen does not appear at all. The map displayed is an original, live and interactive MapInfo map and it is not an image or any type of snapshot. In other words, the user of the application can change the map by zooming, panning, selecting etc. and even add new layers to the map.

Controlling MapInfo from the client application takes place in the form of MapBasic commands. In other words, the client application sends MapBasic commands to MapInfo as strings and MapInfo runs these incoming commands and manipulates the map accordingly.

### 5.2.1 Elements of Integrated Mapping

Integrated mapping basically consists of a client program, MapInfo and optionally one or more compiled MapBasic programs as shown in Figure 51 [21].

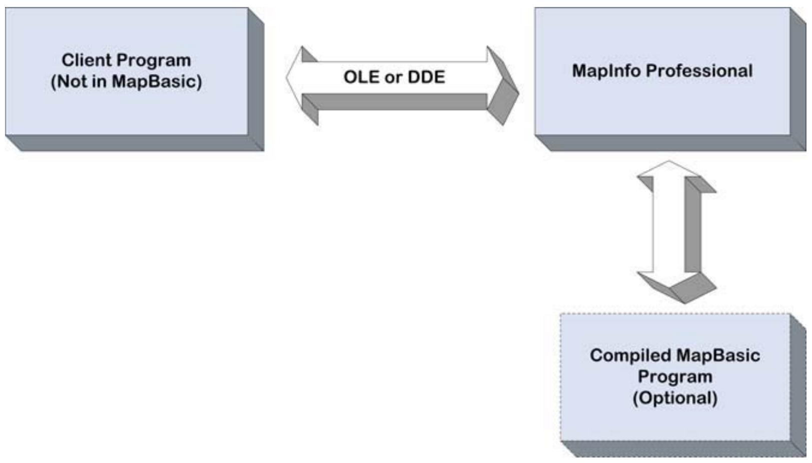


**Figure 51**: Elements of an integrated mapping application [21].

The optional element, namely compiled MapBasic program, is produced by the MapBasic compiler and has the .mbx extension. MBX files are executable programs but they are not standalone executables, i.e. only MapInfo can execute them. It can be a good alternative to use existing MapBasic programs within an integrated mapping application instead of writing the functionality from scratch.

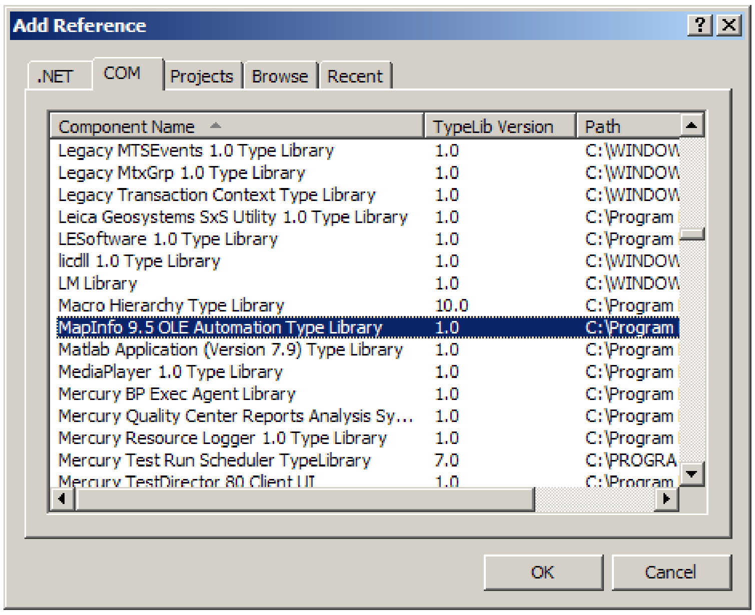
### 5.2.2 Steps to Write Integrated Mapping Applications

Writing an integrated mapping application with Visual Studio roughly involves the following steps:

- Creating a Windows Application in VS.
- Adding a container component such as a panel or picture box to the form in order to host a map.

- Adding a toolbar to the form in order to host the map tool buttons for zooming and panning.
- Adding the MapInfo reference to the project. (Figure 52)
- Writing code to display the map on the form.
- Writing code to function the toolbar buttons.

To add the MapInfo reference to the project, **Add Reference** dialog of VS in Figure 52 is used.



| Component Name | TypeLib Version | Path |
|---|---|---|
| Legacy MTSEvents 1.0 Type Library | 1.0 | C:\WINDOW |
| Legacy MtxGrp 1.0 Type Library | 1.0 | C:\WINDOW |
| Legacy Transaction Context Type Library | 1.0 | C:\WINDOW |
| Leica Geosystems SxS Utility 1.0 Type Library | 1.0 | C:\Program |
| LESoftware 1.0 Type Library | 1.0 | C:\Program |
| licdll 1.0 Type Library | 1.0 | C:\WINDOW |
| LM Library | 1.0 | C:\WINDOW |
| Macro Hierarchy Type Library | 10.0 | C:\Program |
| MapInfo 9.5 OLE Automation Type Library | 1.0 | C:\Program |
| Matlab Application (Version 7.9) Type Library | 1.0 | C:\Program |
| MediaPlayer 1.0 Type Library | 1.0 | C:\Program |
| Mercury BP Exec Agent Library | 1.0 | C:\Program |
| Mercury Quality Center Reports Analysis Sy... | 1.0 | C:\Program |
| Mercury Resource Logger 1.0 Type Library | 1.0 | C:\Program |
| Mercury Test Run Scheduler TypeLibrary | 7.0 | C:\PROGRA |
| Mercury TestDirector 80 Client UI | 1.0 | C:\Program |

**Figure 52**: Adding MapInfo to the project references.

In the COM tab of the **Add Reference** dialog, there is an entry with name *MapInfo X.X OLE Automation Type Library*, where X.X denotes the MapInfo version installed on the machine. This component will provide the required classes to control MapInfo from the client application.

### 5.2.3 Controlling MapInfo

After displaying the map within the client application, the next step is to add new mapping features to the application by controlling MapInfo. This is done with sending MapBasic commands in the form of strings to MapInfo.

MapInfo exposes a public *Do()* method to accept MapBasic commands. It also exports a *RunMenuCommand()* method to run its menu commands. Controlling MapInfo is often a matter of calling the Do() method of the MapInfo object using a string parameter holding a MapBasic command. For example, if we define a variable with name *mi* of type MapInfo application, adding a layer to the map can be achieved in C# with

```
mi.Do(“Add Map Layer CITY”);
```

This statement sends the command between the quotes and then MapInfo executes the command

```
Add Map Layer CITY
```

and adds the CITY layer to the map.

### 5.2.4 Querying Data from MapInfo

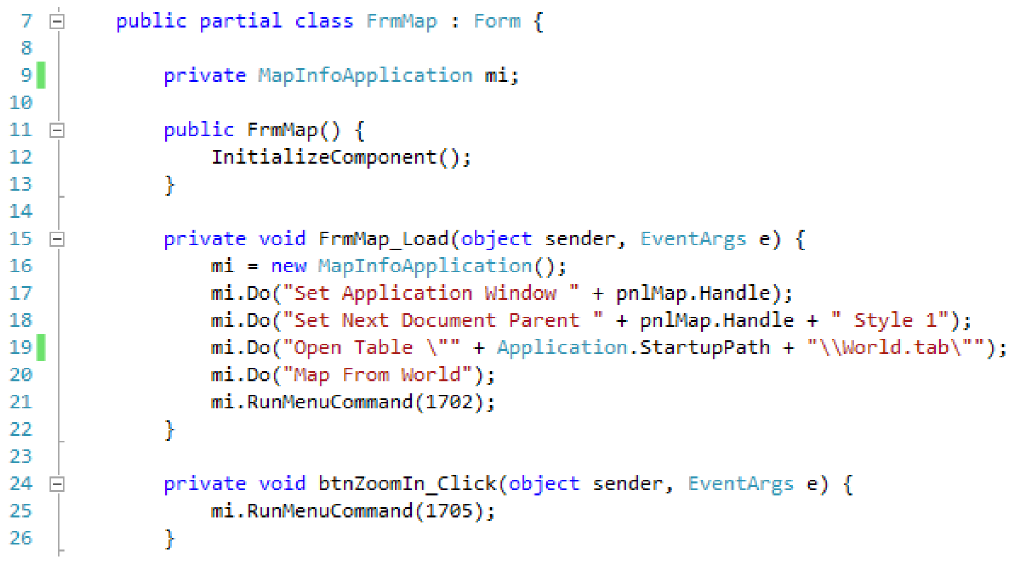
Sometimes it is necessary to get information about MapInfo objects like tables. For example, it is a common operation to loop through the records of a MapInfo table. For this, one must first get the row count of the table. The client application cannot know this information directly and it has to ask this information to MapInfo. Thanks to the MapInfo’s public *Eval()* method, it is possible to get this type of information. For instance, the following expression returns the number of countries in the World table:

```
mi.Eval("TableInfo(\"World\", 8)")
```

Here TableInfo() is a MapBasic function returning information about an open MapInfo table.

### 5.2.5 Inspecting the Hello (Map of) World Application

The important parts of the *Hello (Map of) World* application are given in the following code:



```
 7   public partial class FrmMap : Form {
 8
 9       private MapInfoApplication mi;
10
11       public FrmMap() {
12           InitializeComponent();
13       }
14
15       private void FrmMap_Load(object sender, EventArgs e) {
16           mi = new MapInfoApplication();
17           mi.Do("Set Application Window " + pnlMap.Handle);
18           mi.Do("Set Next Document Parent " + pnlMap.Handle + " Style 1");
19           mi.Do("Open Table \"" + Application.StartupPath + "\\World.tab\"");
20           mi.Do("Map From World");
21           mi.RunMenuCommand(1702);
22       }
23
24       private void btnZoomIn_Click(object sender, EventArgs e) {
25           mi.RunMenuCommand(1705);
26       }
```

This can give a concrete idea to see what integrating mapping applications look like. On line 9, the MapInfo object is declared as of type MapInfoApplication. MapInfoApplication is an interface standing in the MapInfo assembly that is defined with the COM reference in Figure 52.

Between the lines 15 and 22 the actual works are achieved. Line 16 creates a new instance of MapInfo. Then the MapInfo map that has just been created is re-parented in the application on lines 17 and 18. Line 19 opens the World.tab table and loads it into the physical memory. Line 20 displays the map of the World table. Finally the map is switched to the pan mode as if we clicked the pan button on the toolbar of MapInfo.

There is also another method on line 24 that switches to the zoom in tool. RunMenuCommand() method activates a MapInfo menu command with a given integer parameter. The parameter 1705 here corresponds to the zoom in menu in MapInfo.

## 5.3. IMPLEMENTATION OF THE MAPINFOX PROJECT

In order to construct crime mapping software, it is vital to have a practical and usable mapping framework. Integrating mapping provides a powerful way since it allows controlling *all* aspects of MapInfo. This means that using integrated mapping it is possible to achieve *everything* that MapInfo can do. But the problem with integrated mapping is that it is not a practical and reusable way of constructing software. Moreover, sending MapBasic commands to MapInfo often requires generating and concatenating cumbersome strings. Even worse, when generating these strings it is necessary to use nested quotes, which requires escape characters etc. This trouble also increases the probability of run-time errors caused by simple syntax errors since these strings are executed *at run-time* by MapInfo.

These issues proved the need for developing a better mapping framework. MapInfoX provides this framework and it has been developed as an *object oriented* and *reusable* class library, i.e. DLL project.

### 5.3.1 Features and Benefits of MapInfoX

The most significant feature of MapInfoX is that it transforms the integrated mapping into an easy to use *map component* like MapInfo MapX or ESRI ArcObjects. MapInfoX achieves this by wrapping the low-level and complicated integrated mapping implementations as *ready to use objects*.

One of the benefits is that it sets a *type-safe* environment to use the MapBasic commands. This greatly prevents the possible run-time errors triggered by incorrectly generated command strings.

Using MapInfoX as the map component also offers a better and easier debugging experience.

Code using the MapInfoX library will be much more readable and cleaner since it provides a concise syntax by means of objects. Integrating mapping, on the other hand, typically requires generating long strings. Comparing the following usages of the two in Table 1 demonstrates the difference.

**Table 1**: Getting the row count of a table with and without MapInfoX.

Without MapInfoX

```
int rowCount = int.Parse(mi.Eval(string.Format(@"TableInfo(""{0}"", 8)", “City”)));
```

With MapInfoX

```
Table table = map.Catalog[“City”];
int rowCount = table.RowCount;
```

As Table 1 suggests, using MapInfoX to get the row count of a table is much easier and has a safer syntax. The difference is even more obvious with relatively complicated operations like getting the columns of a table as in Table 2:

**Table 2**: Getting the columns of a table with and without MapInfoX.

Without MapInfoX

```
List<string> columnNames = new List<string>();
MapInfoApplication mi = MapContext.GetMap().MapInfoApplication;
int columnCount = int.Parse(mi.Eval("NumCols(" + “City” + ")"));
for (int i = 0; i < columnCount; i++) {
string columnName = mi.Eval(string.Format(@"ColumnInfo(""{0}"", ""{1}"", 1)", “City
”, "Col" + (i + 1)));
columnNames.Add(columnName);
}
```

With MapInfoX

```
Table table = map.Catalog[“City”];
string[] columnNames = table.ColumnNames;
```

To present these benefits, MapInfoX handles the complex implementation details in *internal* classes, which are accessible to only the classes in the same assembly. Then it exposes the handy public APIs (Application Programming Interface) to the external world.

Development of the MapInfoX library is the most crucial and time-consuming part of this study.

### 5.3.2 Implementing the Map Component

Two of the most important classes in MapInfoX are *MapControl* and *Map* classes. The MapControl class extends System.Windows.Forms.PictureBox and it is a visual component that can be dropped into a form surface from the VS toolbox.

The Map class is a high level abstraction of the MapInfo application and provides the necessary properties and methods to access and manipulate various map elements such as layer collection, tools, catalog and selection.

The Map class implements the *IMap* interface. This interface based design has been constructed to provide flexibility. For instance, if another mapping framework is to be used instead of MapInfo, the existing framework can be replaced with a new one without breaking the existing design.

Key elements of the Map class will be examined in the next sub sections.

Figure 53 depicts the expanded class diagrams of the MapControl and Map classes and the IMap interface.

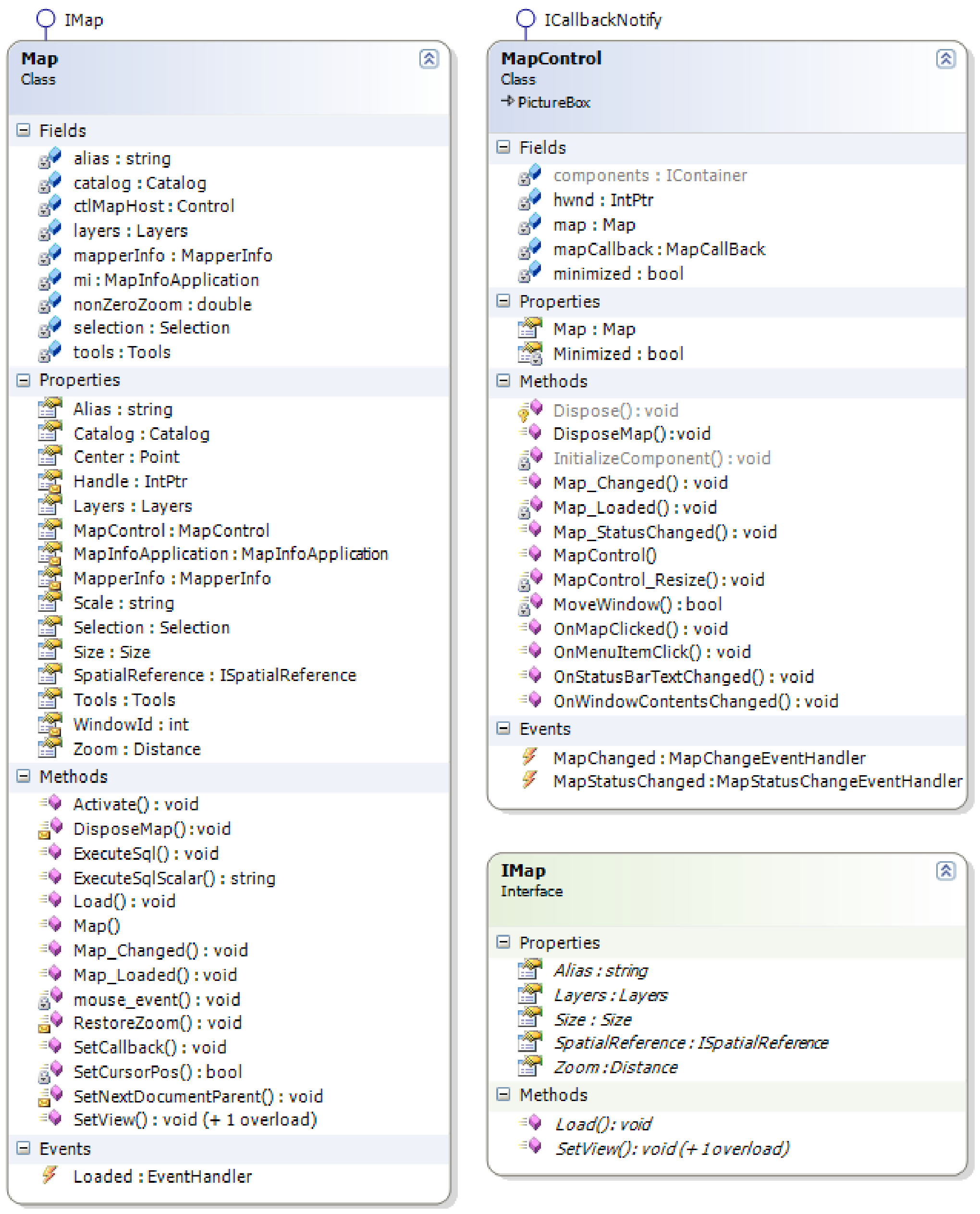


**Figure 53**: Class diagram for MapControl, Map and IMap.

### 5.3.2.1 Layers Collection

Layers class represents a collection of layers in a map and provides methods to perform standard operations such as enumerating, adding, removing and moving layers. It also implements some standard list ADT (Abstract Data Type) methods like Count and RemoveAt. Layers class extends the CollectionBase class and implements the IList and ICollection interfaces.

Layers class also has a property CosmeticLayer and this property returns the cosmetic layer of a map. Cosmetic layer is a special layer in MapInfo and it is something like a scratch paper for the map. It is commonly used for temporary display operations.

Figure 54 displays the members of the Layers class.

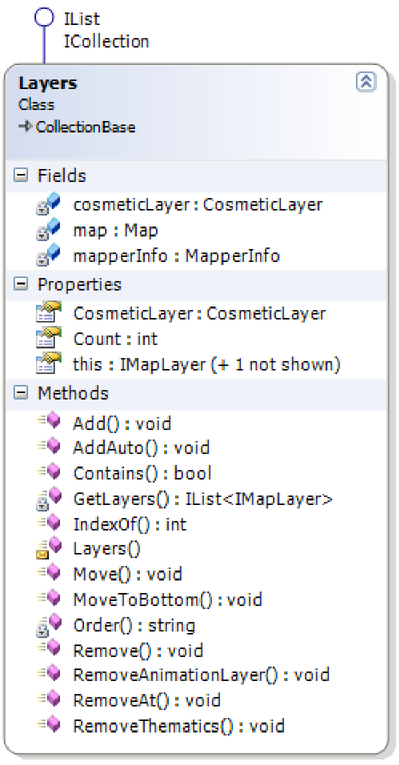


**Figure 54**: Layers class members.

One of the most used properties of the Layers class is the indexer that returns a map layer according to the layer name. Its signature is

```
public IMapLayer this[string name]
```

Also there is an overload of this property that returns a map layer according to its index, i.e. layer order, as follows:

```
public IMapLayer this[int index]
```

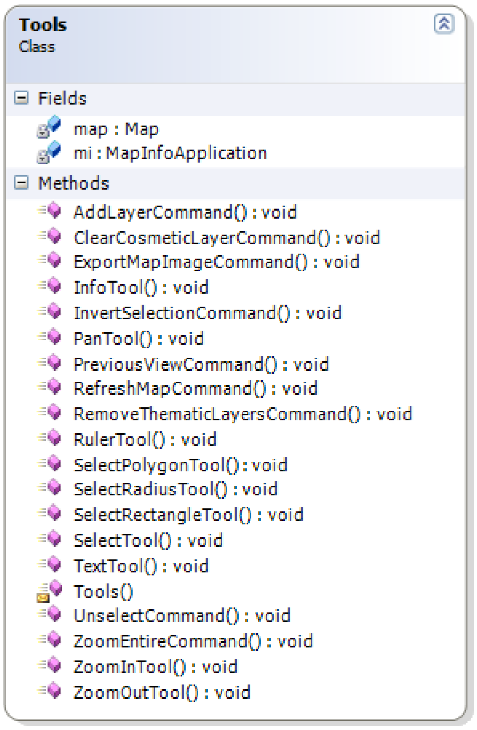
These two properties are important since they allow accessing a map layer easily and then perform subsequent operations like creating themes on that layer.

### 5.3.2.2 Tools

Tools class has the essential methods to function the map toolbar buttons like zoom in and pan. In fact, it delegates the map toolbar functions to the MapInfoApplication class, which does the actual work by activating the commands and tools. A typical method in the Tools class is the ZoomInTool() method below:

```
public void ZoomInTool() {
    mi.ZoomIn();
}
```

The members of the Tools class are given in Figure 55:



```
Tools
Class

Fields
    map : Map
    mi : MapInfoApplication
Methods
    AddLayerCommand() : void
    ClearCosmeticLayerCommand() : void
    ExportMapImageCommand() : void
    InfoTool() : void
    InvertSelectionCommand() : void
    PanTool() : void
    PreviousViewCommand() : void
    RefreshMapCommand() : void
    RemoveThematicLayersCommand() : void
    RulerTool() : void
    SelectPolygonTool() : void
    SelectRadiusTool() : void
    SelectRectangleTool() : void
    SelectTool() : void
    TextTool() : void
    Tools()
    UnselectCommand() : void
    ZoomEntireCommand() : void
    ZoomInTool() : void
    ZoomOutTool() : void
```

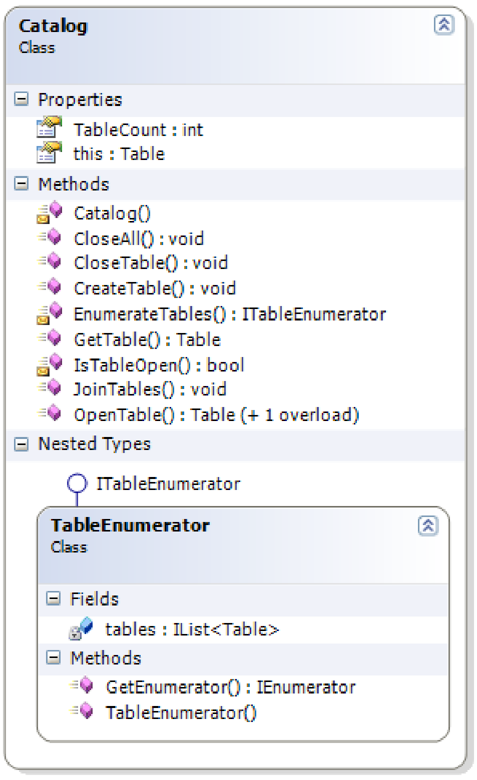
**Figure 55**: Tools class members.

### 5.3.2.3 Catalog

Catalog class provides access to the currently open MapInfo tables. It presents methods to create a new table or to open, close, enumerate and join the tables in a MapInfo session.

The catalog concept here in fact is similar to a database in that it encapsulates MapInfo tables in a MapInfo session. But there is a difference between them. In a typical relational database, all database objects such as tables or views are available to making queries or updates once connected. MapInfo tables, on the other hand, are not available until they are *opened*.

Catalog class members are shown in Figure 56:



**Catalog**
Class

Properties
- TableCount : int
- this : Table

Methods
- Catalog()
- CloseAll() : void
- CloseTable() : void
- CreateTable() : void
- EnumerateTables() : ITableEnumerator
- GetTable() : Table
- IsTableOpen() : bool
- JoinTables() : void
- OpenTable() : Table (+ 1 overload)

Nested Types

ITableEnumerator

**TableEnumerator**
Class

Fields
- tables : IList<Table>

Methods
- GetEnumerator() : IEnumerator
- TableEnumerator()

**Figure 56**: Catalog class members.

### 5.3.2.4 Selection

Selection class represents a query result set from a table or a collection of the selected objects on the map and essentially is a temporary MapInfo table. The table on which a query is executed is called the *base table*. Selection class has three properties returning the base table name, row count and temporary table name. It also has methods to iterate through the records of a query result.

The members of the Selection class are provided in Figure 57.

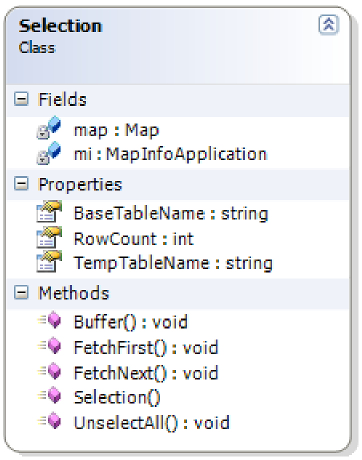


**Figure 57**: Selection class members.

### 5.3.3 Implementing the Map Toolbar

The map toolbar includes the map tools such as zoom in and pan to manipulate and interact with the map. The name of the underlying class is also MapToolBar and it extends System.Windows.Forms.ToolStrip.

All map toolbar buttons extend the MapTool abstract class, which extends System.Windows.Forms.ToolStripButton. BaseCommand and BaseTool classes are the base classes of the command and tool buttons, respectively. For example, UnselectCommand button extends BaseCommand and ZoomInTool button extends BaseTool, as in Figure 58.

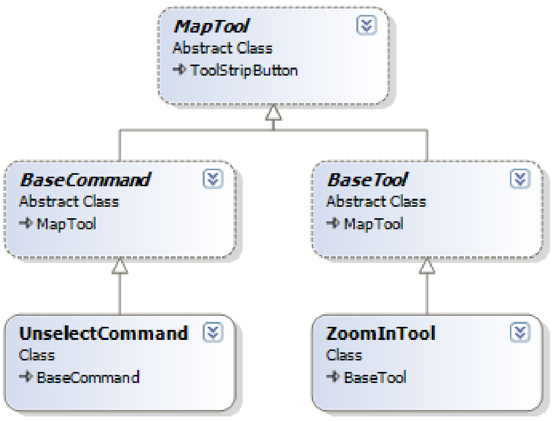


**Figure 58**: Inheritance hierarchy of BaseCommand and BaseTool classes.

Adding buttons to the MapToolBar is done in the MapToolBar constructor with a code like

```
Items.Add(new ZoomInTool());
```

Creating a new toolbar button is a matter of writing a static constructor and overriding the OnClick() method from the base class as in the following code:

```
internal class UnselectCommand : BaseCommand {

    static UnselectCommand() {
        icon = Resources.Unselect as Image;
        caption = "";
        tooltip = strings.UnselectTool;
    }

    public override void OnClick(object sender, EventArgs e) {
        if (MapControl != null) {
            MapControl.Map.Tools.UnselectCommand();
        }
    }
}
```

Here MapControl is a protected property from the base class MapTool which returns the current map window's map control object.

### 5.3.4 Implementing the Data Access Layer

A Data Access Layer (DAL) is a software layer providing convenient and simplified access to persistent data sources such as databases. DAL is typically implemented with data access components for different data sources. "These components abstract the logic required to access the underlying data stores. They centralize common data access functionality in order to make the application easier to configure and maintain. Some data access frameworks may require the developer to identify and implement common data access logic in separate reusable helper or utility data access components [22]."

Crime Analyzer has a DAL that implements access to two databases: MS Access and Oracle. In this implementation, there are two concrete classes, namely MSAccessConnection and OraConnection, extending an abstract base class DataBaseConnection as in Figure 59:

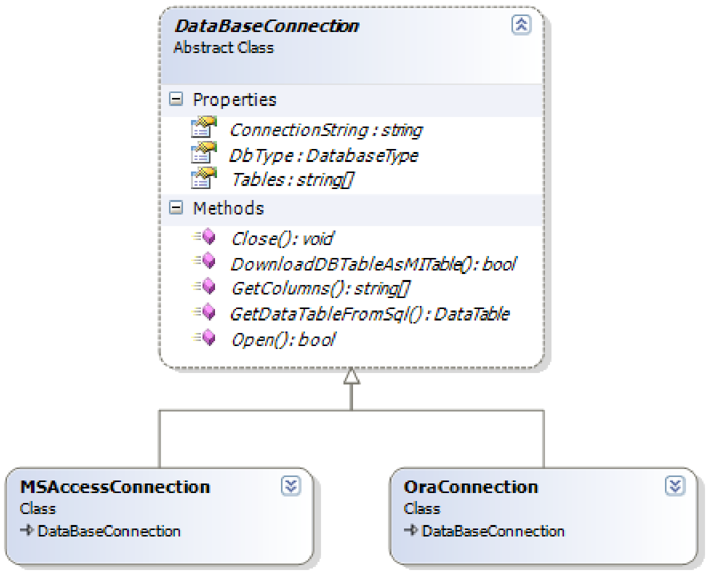


**Figure 59**: DAL components.

MSAccessConnection and OraConnection classes both implement the abstract methods in the DataBaseConnection class. MS Access and Oracle both are relational databases and they implement data query and manipulation via SQL (Structured Query Language). But they differ in some points such as connection strings and SQL syntax. Thus, separate implementation classes (MSAccessConnection and OraConnection) are given for each database. These two classes implement the standard operations in a typical database such as opening and closing the connection, enumerating database tables, returning a result set from an SQL clause etc. There is also a MapInfo-specific method, namely DownloadDBTableAsMITable() with signature

```
public abstract bool DownloadDBTableAsMITable(string table, string path);
```

which downloads a database table as a MapInfo table. This method is required for pulling data from a corporate database and joining it with a MapInfo table; for generating thematic maps for example.

### 5.3.5 Implementing the Thematic Analyses

Generating a thematic map basically requires a MapInfo table and a thematic column. In other words, there is a common operation with different implementations. Hence it is suitable to abstract out all thematic map types with an abstract class, namely AbstractTheme. This abstract class also implements the ITheme interface, which gives three methods to implement:

- FeatureLayer: Represents the MapInfo layer upon which the thematic map will be generated.
- Expression: Typically represents the thematic column name. It can also be an expression from two or more columns.
- Legend: Represents the thematic legend object displaying the legend items.

Class diagram of the AbstractTheme class together with the two of the implementing classes and the ITheme interface are shown in Figure 60.

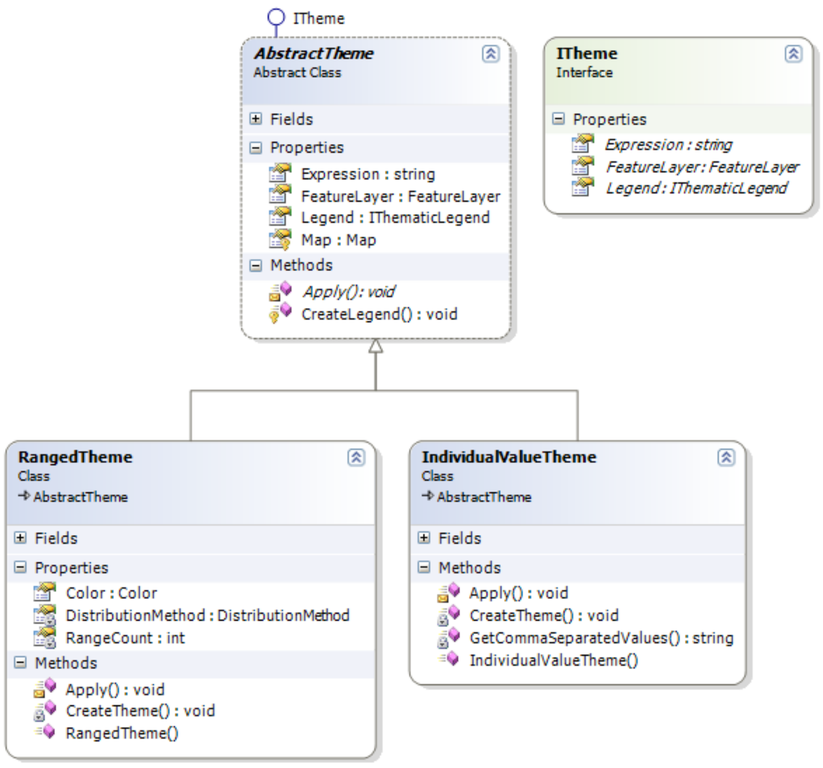


**Figure 60**: AbstractTheme base class and two implementing classes.

The IThematicLegend interface offers two properties Title and SubTitle to implement as shown in Figure 61:

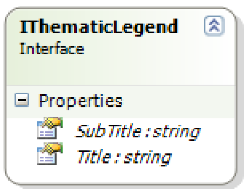


**Figure 61**: IThematicLegend interface members.

AbstractTheme class also provides a default implementation for creating legend by means of the protected CreateLegend() method.

After completing the implementations of the thematic analysis classes such as RangedTheme and IndividualValueTheme, generating thematic maps are trivial. A typical code to create a thematic (IndividualValueTheme) map can be as follows:

```
IndividualValueTheme theme = new IndividualValueTheme(fLayer, expr, true);
theme.Legend.Title = string.Format("{0}:", strings.IndividualValues);
theme.Legend.SubTitle = expr;
fLayer.ApplyTheme(theme);
```

where fLayer is the FeatureLayer object to apply the thematic, expr is the column expression and the third parameter (namely, the *true* value) specifies that zeroes or blanks will be ignored.

### 5.3.6 Implementing the Temporal Analyses

There are two temporal chart types in the system, namely data calendar and data clock. They are implemented in the DataCalendar and DataClock classes, respectively. These classes extend a common base class TemporalChart. This base class provides both protected constants and methods to be used in deriving classes. Examples of the protected constants are TOP_MARGIN, LEFT_MARGIN and TITLE_FONT. The protected methods are CopyToClipboard() and SaveAsImage(). These common methods are called from the context menus of DataCalendar and DataClock when they are right clicked. While the protected constants ensure format consistency, the protected methods give reusability.

The class diagrams of the three classes mentioned are depicted in Figure 62:

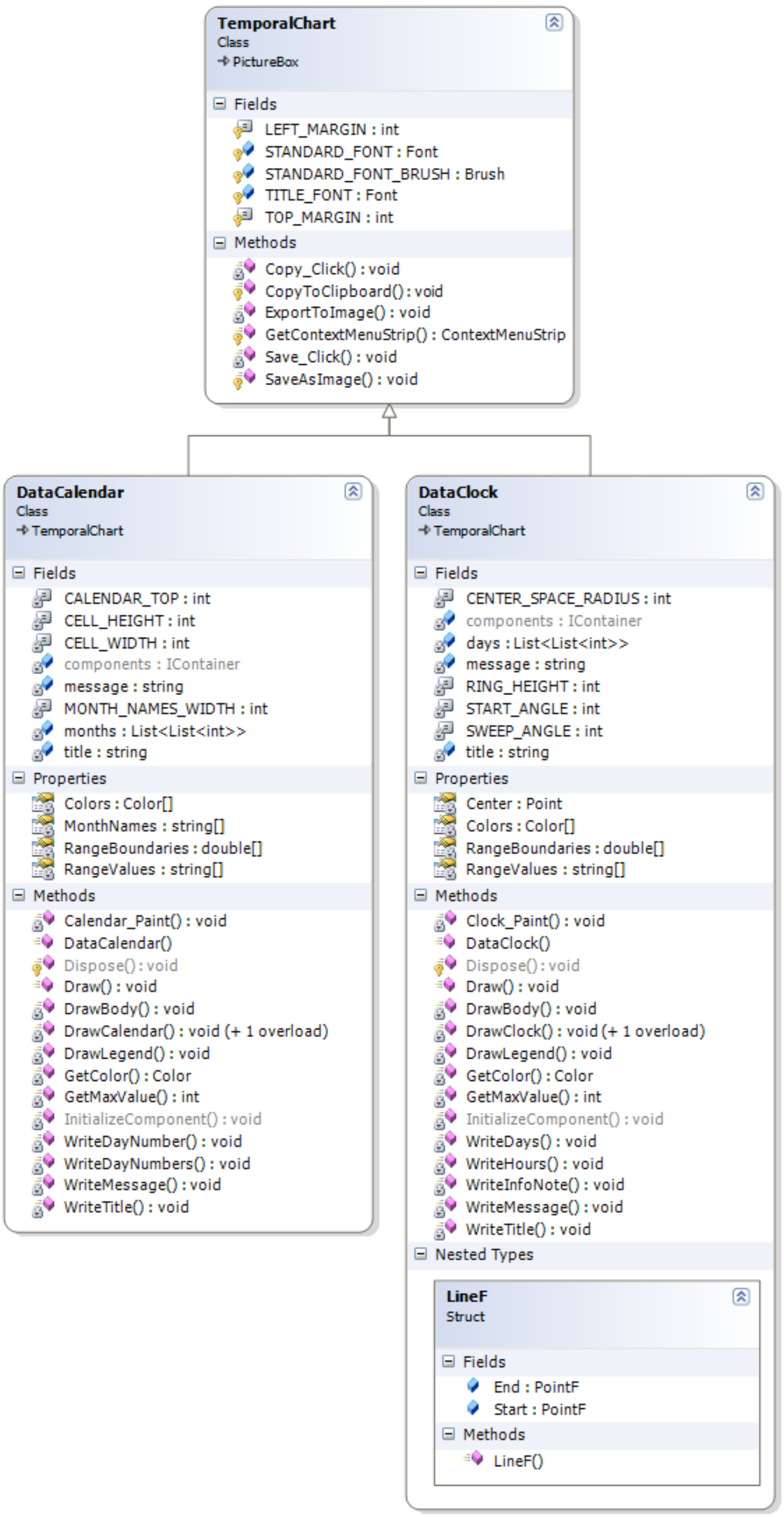


**Figure 62**: DataCalendar and DataClock class diagrams.

As shown in Figure 62, the TemporalChart (and thus DataCalendar and DataClock) extends System.Windows.Forms.PictureBox control to easily perform the drawing operations. These operations utilize the classes in the System.Drawing namespace such as Graphics, Rectangle, Pen and Brush.

Drawing a data calendar is relatively simple since it mainly involves creating adjacent rectangles. This has been achieved using the Graphics.DrawRectangle() and Graphics.FillRectangle() methods. Other stuff like legend texts and month names has been handled with the Graphics.DrawString() method.

Drawing a data clock, on the other hand, is a more complex operation because of its non-standard cell shape. Unlike a data calendar cell, which is a rectangle, a data clock cell does not have a pre-defined shape in the .NET drawing classes. Hence, it has been implemented using the *System.Drawing.Drawing2D.GraphicsPath* class representing a series of connected lines and curves.

## 5.4. DESIGN PATTERNS USED IN MAPINFOX

In software engineering, a design pattern is a *general reusable solution* to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a *description* or *template* for how to solve a problem that can be used in many different situations. Patterns are formalized best practices that the programmer must implement themselves in the application [23].

With another definition, a design pattern is a *general technique* used to solve a class of related problems. It isn’t a specific solution to the problem. Probably every architect who came up with an observably pleasant room brought light into that room in a different way, and probably every programmer implemented their solution differently. The pattern is the general structure of the solution—a “metasolution” if you will—not the solution itself [24].

Design patterns make it easier to reuse successful designs and architectures. Expressing proven techniques as design patterns makes them more accessible to developers of new systems. Design patterns help you choose design alternatives that make a system reusable and avoid alternatives that compromise reusability. Design patterns can even improve the documentation and maintenance of existing systems by furnishing an explicit specification of class and object interactions and their underlying intent. Put simply, design patterns help a designer get a design right faster [25].

A design pattern has four elements: *Pattern name*, *problem*, *solution* and *consequences*. Design patterns help construct better and flexible software. Not only constructing but also maintaining software is a hard task. Hence, the design of software should allow the developers to add new features easily and to modify the system without breaking any existing functionality. Design patterns, for instance, encourage *designing to interfaces* rather than initially focusing on implementation. They also stimulate loose coupling and high cohesion about objects. Encapsulation is another important concept to be implemented in object-oriented software development. Encapsulating, or *hiding*, implementation details, object data or classes can greatly contribute to flexible software.

MapInfoX has some design and implementation features that use a few design patterns. An example pattern that has been used is the *Observer Pattern*. “Observer pattern defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically [26].” When implementing the callback messages coming from MapInfo such as current cursor coordinates, MapInfoX uses delegates and events to notify the map statusbar and update its contents accordingly.

A second pattern that has been applied is the *Template Method Pattern*. According to this pattern, “we define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure [27].” This pattern is clearly suitable in implementing the DAL since although there are different methods for connecting and querying MS Access and Oracle databases, they share the same conceptual process. “The Template Method gives us a way to capture this common ground in an abstract class while encapsulating the differences in derived classes [28].” Here the abstract class is DataBaseConnection and the derived classes are MSAccessConnection and OraConnection. An important advantage of the Template Method is that it simplifies adding new features to the system without touching the existing classes. For example, it will be both *easy* and *safe* to implement a new database such as MS SQL Server in MapInfoX since this will involve creating a new class (MSSqlServerConnection e.g.) and then extending it from the DataBaseConnection abstract base class.

The *Facade Pattern* is another example. “It provides a unified interface to a set of interfaces in a subsystem. Facade defines a higher-level interface that makes the subsystem easier to use [29].” The IMap interface exposes a high level interface to the external world and the Map class simplifies the complexities of the integrated mapping operations such as callbacks by implementing the IMap interface.

Another implemented pattern is the *Iterator Pattern*. “It provides a way to access the elements of an aggregate object sequentially without exposing its underlying representation [30].” Enumerating the open tables in a map is a common operation in MapInfoX and this can be done easily using the TableEnumerator class, which implements the Iterator Pattern.

“The *Abstract Factory Pattern* provides an interface for creating families of related or dependent objects without specifying their concrete classes [31].” This pattern is useful when coordinating the instantiation of the right chart type such as bar chart or pie chart.

# SUMMARY AND CONCLUSION

This study has both theoretical studies and a software development from scratch. Theoretical researches cover GIS in general and its applications, the relation between GIS and crime analysis, and crime mapping. The software development part involves first designing and implementing a mapping framework in the form of a class library, and then implementing the crime mapping modules using this mapping framework. The significant feature of the framework is that it has been designed as a *reusable* library and it can be easily used for any integrated mapping application developed with MapInfo. It not only provides crime mapping modules but also presents general purpose functionality such as ready to use map controls, table catalog or data access.

By providing 15 different spatial, temporal and graphical analysis tools, Crime Analyzer provides a *practical solution* for crime mapping and crime prevention. It is beneficial for organizations that fight with crime since it allows a crime analyst to display the crime data in various visualizations and to *infer results* from analyses. Managers can also use Crime Analyzer to deploy the field forces at *right locations*. Moreover, after making temporal analyses and seeing the top crime hours in a location, a manager can direct patrol cars in *right hours*.

Another major characteristic of Crime Analyzer is that it is not dependent with a particular database or a table structure. It has been designed and implemented to work with *any* MS Access or Oracle database. Pulling the instant information from a corporate database and performing the required joins with spatial tables are done automatically. Therefore, the software may be used in *any* police or crime research department.

# REFERENCES

1. http://en.wikipedia.org/wiki/Geographic_information_system

2. **MARAŞ, H.H.** (1999), *An Application and Design of a Geographic Information System for Updating of a Geographic Database,* PHd. Thesis, Istanbul Technical University, Istanbul. (in Turkish)

3. ESRI GIS Dictionary at: http://support.esri.com/es/knowledgebase/GISDictionary/term/layer

4. **BOBA, R.** (2005), *Crime Analysis and Crime Mapping*, Sage Publications.

5. http://en.wikipedia.org/wiki/Crime_analysis

6. **BOBA, R.** (2005), *Crime Analysis and Crime Mapping*, Sage Publications.

7. http://en.wikipedia.org/wiki/Crime_analysis

8. http://en.wikipedia.org/wiki/Crime_mapping

9. **KARAKAŞ, E.** (2004), *Burglary Crime Distribution and its Characteristics in Elazığ City*, Elazığ.

10. ESRI UK CrimeAnalyst site at: http://www.esriuk.com/software/arcgis/crimeanalyst

11. http://en.wikipedia.org/wiki/CrimeAnalyst

12. MapInfo Crime Profiler data sheet at: http://www.pb.com/docs/US/pdf/Products-Services/Software/Data-Mining-and-Modeling/Geographic-Data-Mining-Tools/Crime-Profiler/mapinfo-crime-profiler-data-sheet-2012.pdf

13. MapInfo Crime Profiler data sheet at: http://www.pbsoftware.eu/uk/files/download/products/location-intelligence/UK_Crime_Profiler_Datasheet_v1-1_LoRes.pdf

14. CrimeStat site at:

    http://www.icpsr.umich.edu/CrimeStat/about.html

15. Thematic Maps Map Collection & Cartographic Information Services Unit. University Library, University of Washington. (2009)

16. MapInfo User Guide: http://reference.mapinfo.com/software/mapinfo_pro/english/10/MapInfoProfessionalUserGuide.pdf

17. **JENKS, G. F., CASPALL, F. C.** (1971), *Error on Choroplethic Maps: Definition, Measurement, Reduction from the Annals of American Geographers*.

18. **TOMPSON, L., PARTRIDGE, H., SHEPHERD, N.** *Hot Routes: Developing a New Technique for the Spatial Analysis of Crime*, London.

19. http://www.codeproject.com/Articles/11505/NGif-Animated-GIF-Encoder-for-NET

20. http://www.codeproject.com/Articles/43181/A-Serious-Outlook-Style-Navigation-Pane-Control

21. MapBasic User Guide: http://reference.mapinfo.com/software/mapbasic/english/9.5/MapBasicUserGuide.pdf

22. MSDN Article, *Data Layer Guidelines*, at: http://msdn.microsoft.com/en-us/library/ee658127.aspx

23. http://en.wikipedia.org/wiki/Software_design_pattern

24. **HOLUB, A.** (2004), *Learning Design Patterns By Looking At Code*, Apress.

25. **GAMMA, E.** et. al. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

26. **GAMMA, E.** et. al. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

27. **GAMMA, E.** et. al. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

28. **SHALLOWAY, A., TROTT, J.R.** (2002), *Design Patterns Explained*, Addison-Wesley.

29. **GAMMA, E.** et. al. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

30. **GAMMA, E.** et. al. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

31. **GAMMA, E.** et. al. (1995), *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley.

# CV

## PERSONAL INFORMATION

| | | |
|---|---|---|
| Surname, Name | : | ÖZÇETİN, Mustafa |
| Nationality | : | T.C. |
| Birth Date & Location | : | 30 / 06 /1976, Kırklareli |
| Marital Status | : | Married |
| Phone Number | : | 0506 672 08 22 |
| e-Mail | : | mustafaozcetin76@yahoo.com |

## EDUCATION

| **Degree** | **School** | **Graduation Year** |
|---|---|---|
| Bachelor of Science | METU | 1998 |
| High School | Alpullu High School | 1993 |

## PROFESSIONAL EXPERIENCE

| **Year** | **Company** | **Position** |
|---|---|---|
| 2001 | Docuart Computers & Communications | Software Developer |
| 2004 | Alfabim Computer Systems | Project Team Leader |
| 2006-Now | TÜBİTAK | Senior Researcher |

## FOREIGN LANGUAGES

English – Good (METU Proficiency Exam Score: 82)

## HOBBIES

Chess, internet, playing football, reading books