Design and development of prototype Airport Noise Information System
(A case study of Split Airport, Croatia)

by

Yongjun Zhao

Master of Science in Geoinformatics (GFM.2)

February 2002
Design and development of Prototype Airport Noise Information System  
(A case study of Split Airport, Croatia)  

by  

Yongjun Zhao  

Thesis submitted to the International Institute for Geo-information science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in Geoinformatics.  

Degree Assessment Board  
Chairman: Prof.Dr. Menno-Jan Kraak  
External examiner: Prof. Dr. Ferjan Ormeling  
Supervisor: Ms. Yuxian Sun M.Sc.  
Second supervisor: Ms. Drs. Connie Blok  

ITC  

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION  
ENSCHENDE, THE NETHERLANDS
Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.
Acknowledgement

First of all I would like to thank the Netherlands government and ITC sincerely for providing the financial support and a good study environment for me. Special thanks go to Chinese State Bureau of Surveying and Mapping which offers me this study opportunity. I wish to thank HeiLongJiang Bureau of Surveying and Mapping and Harbin Cartographic Publishing House, and specifically Mr. ZhiGang Li and Mr. Xue Dong who encouraged and supported me in all ways during my study!

My heartfelt gratitude goes to my first supervisor, Ms. Yuxian Sun. When the first time you mentioned those many books that are completely new to me, I was half scared and half excited by the challenge! Your deep insight for the research, countless inspired discussions, and the quite special way of supervision guided me step by step to finish this research.

To my second supervisor Ms. Connie Blok, I want to show my very special thanks and appreciations. You have given a lot of supervision and time to my research and so much invaluable suggestions, much more than I could ever expect. I will never forget your patient guidance and careful corrections for the entire thesis.

My sincere thanks go to Martina Baucic and Stipe Baucic who proposed this research and accompanied me during my whole fieldwork in Croatia. You have always encouraged and supported me. Thanks for your great help! I also want to say thanks to all Geodata staff who provide me unforgettable working environment and a lot of good memories of Croatia. I must acknowledge Split airport, Croatia, that gave me essential support for user requirements gathering and data collection. Especially thanks Mr. Luksa Novak, Mr. Slavko Roguljic and Mr. Josko Tadin.

To my student advisor Mr. Gerrit Huurneman, thank you for your 18 months guidance and many help, especially for the great support and encouragement for my research!

I would like to thank Mr. Bright Oseitwumasi for our interesting discussions and your valuable suggestions! My special thank goes to Mr. Thomas Tsiboah for your encouragement and discussions and generous offer of your computer to ensure me finish this research. Many thanks for Mr. Zhongwei Sun’s inspired discussions and valuable suggestions!

Thanks all GFM.2 2000 and GFM.3 2000 classmates for giving me very happy environment to enjoy my study.

I wish to thank the entire Chinese group at ITC who gave me a lot of help and spirit support to relief my homesick, especially Ms. Lichun Wang who gave me so much caring and encouragement. I must mention my best friend Ms. XiaoBai Yao who is current doing Ph.D in U.S, for our countless email exchange, which gave me great moral support and inspiration on my research!

I cannot express enough of my appreciation to my parents who took care of my young daughter during my long study and always encouraged me to study hard. I owe them too much! And I owe my daughter who needs her mother beside her, but I cannot... At last but not least, I want to say thanks to my husband Mr. LianPing Zhang, who encourages me to apply for this study opportunity from the beginning and kept giving me your support and love to help me finish my study successfully!

My thanks also go to those whose names are not mentioned here but encouraged me and helped me during my research in one way or another. Thank you all!
Abstract

With the fast growth of aviation, the aircraft noise problem is gaining more and more public attentions. Airport managers are pressured by the government to solve airport noise problems. Residents who are under noise disturbance need the government and airport to work out corresponding regulations and solutions in order to keep the noise disturbance under a certain level.

The study area for this research is Split airport, Croatia. The airport has been warned by the local government that the airport noise is damaging old churches and disturbing residents. Currently the airport has a noise monitoring and a noise modelling system. The airport needs to integrate these two existing systems together with GIS capabilities, especially spatial analysis functionalities to aid the airport noise management. Therefore, an Airport Noise Information System (ANIS) is needed.

The objective of this research is to design and develop a prototype ANIS. The main advantage of adopting software engineering and object-oriented technology is to ensure a system that has high quality, is reusable, and easy to maintain. Unified Modelling Language (UML) is a tool for object-oriented system design. It consists of different types of diagrams corresponding to different design stages. In the research, a use case diagram of Split ANIS is developed for user requirements gathering and analysis. A class diagram is developed to represent all the classes and their attributes, operations, associations, and data types within Split ANIS. It models the static aspects of this system. A data dictionary is developed to describe the terminologies used in class diagram. Interaction diagram is developed for illustrating how system messages are sent between objects and classes. It models the dynamic aspects of the Split ANIS. Pre and postconditions of each system operation in the class diagram are developed to describe how the system responsibilities are assigned to different objects and classes. Based on the class diagrams, interaction diagrams and system responsibility descriptions, a prototype is implemented using Visual Basic for Applications (VBA) embedded in ArcObjects.

The main GIS functionalities of this prototype ANIS include:

- Detects areas which are affected by noise currently and potentially
- Analyses land use compatibility with noise distribution
- Visualizes noise change and noise disturbance based on census information
- Presents user defined areas

The testing result shows that the prototype ANIS provides sufficient functionalities for airport noise management and the user interfaces are easy to operate.

The research questions are all answered and the research objective is reached.
Table of Contents

1. INTRODUCTION................................................................................................................. 1-1
   1.1. BACKGROUND............................................................................................................... 1-1
   1.2. CASE STUDY AREA ...................................................................................................... 1-1
   1.3. RESEARCH OBJECTIVE ............................................................................................... 1-5
   1.4. RESEARCH QUESTIONS ............................................................................................... 1-5
   1.5. RESEARCH METHODOLOGY ....................................................................................... 1-5
       1.5.1. Literature review.................................................................................................. 1-5
       1.5.2. System design and development.......................................................................... 1-6
   1.6. THESIS STRUCTURE .................................................................................................... 1-6

2. LITERATURE REVIEW ON SOFTWARE ENGINEERING AND OBJECT-ORIENTED TECHNOLOGY ........................................................................................................................................................................... 2-9
   2.1. INTRODUCTION.......................................................................................................... 2-9
   2.2. SOFTWARE ENGINEERING AND OBJECT-ORIENTED TECHNOLOGY ......................... 2-9
       2.2.1. Software development models ........................................................................... 2-10
       2.2.2. Object-oriented technology ............................................................................. 2-12
   2.3. UNIFIED MODELLING LANGUAGE (UML) FOR SOFTWARE DESIGN ......................... 2-15
       2.3.1. Analysis—Domain modelling............................................................................. 2-15
       2.3.2. Analysis—Software specification ....................................................................... 2-19
       2.3.3. Design............................................................................................................... 2-20
   2.4. IMPLEMENTATION AND TESTING.............................................................................. 2-22
       2.4.1. Introduction....................................................................................................... 2-22
       2.4.2 Software quality ................................................................................................ 2-23
       2.4.3 Testing .............................................................................................................. 2-24
   2.5. SUMMARY ................................................................................................................. 2-24

3. AIRPORT NOISE MANAGEMENT AND CASE STUDY AREA .............................................. 3-25
   3.1. INTRODUCTION.......................................................................................................... 3-25
   3.2. AIRPORT NOISE MEASUREMENT ........................................................................... 3-25
       3.2.1. Noise............................................................................................................... 3-25
       3.2.2. Airport noise metrics ...................................................................................... 3-25
   3.3. FACTORS AFFECTING AIRPORT NOISE.................................................................... 3-27
   3.4. LAWS AND REGULATIONS RELEVANT TO AIRPORT NOISE .................................. 3-30
       3.4.1. Aviation Noise Abatement Policy of the United States ...................................... 3-30
       3.4.2. Airport noise compatibility planning in the USA .............................................. 3-31
       3.4.3. Laws and regulations relevant to aircraft noise of Croatia............................... 3-32
   3.5. THE EXISTING AIRPORT NOISE MONITORING AND MODELLING SYSTEMS IN SPLIT AIRPORT, CROATIA........................................................................................................................................................................................................................................... 3-34
       3.5.1. Noise Monitoring System (NMS) ...................................................................... 3-34
       3.5.2. Integrated Noise Modelling (INM)..................................................................... 3-34
       3.5.3. Problems with the current systems .................................................................. 3-35
   3.6. SUMMARY .................................................................................................................. 3-35
4. ANALYSIS AND DESIGN OF THE PROTOTYPE AIRPORT NOISE INFORMATION SYSTEM

4.1. USE CASE DIAGRAM AND USER REQUIREMENT ANALYSIS

4.1.1. Use case diagram

4.1.2. User requirement analysis

4.2. CLASS DIAGRAM FOR SYSTEM DESIGN

4.2.1. Class diagram

4.3. SYSTEM BEHAVIOUR MODELLING

4.3.1. Interaction diagram

4.3.2. Assigning system responsibilities

4.4. SUMMARY

5. SYSTEM IMPLEMENTATION AND TESTING

5.1. GEODATABASE

5.1.1. Geodatabase introduction

5.1.2. Ways to create a geodatabase

5.2. USER INTERFACE DESIGN AND IMPLEMENTATION OF SPLIT ANIS, CROATIA

5.2.1. Airport noise monitoring

5.2.2. Airport noise modeling

5.2.3. Airport noise GIS

5.3. SYSTEM TESTING

5.3.1. Developer’s acceptance testing

5.3.2. User’s acceptance testing

5.4. SUMMARY

6. CONCLUSION AND RECOMMENDATIONS

6.1. REVIEW RESEARCH OBJECTIVES AND RESEARCH QUESTIONS

6.2. CONCLUSIONS

6.3. RECOMMENDATIONS

REFERENCES

APPENDIX A: DATA DICTIONARY

APPENDIX B: USE CASE DIAGRAM

APPENDIX C: CLASS DIAGRAM
List of Tables

Table 1 General information of Split Airport, Croatia (From Split airport, 2001) .............................1-3
Table 2 Land use compatibility with different noise level (From PCR Environmental, 2000) ............3-31
List of Figures

Figure 1-1 Study area............................................................................................................. 1-3
Figure 2-1 Waterfall software development model (Barroca et al., 2000)................................. 2-10
Figure 2-2 Iterative and incremental development, modified (from Larman, 2002).................... 2-11
Figure 2-3 Iterative and incremental development model (Barroca et al., 2000)....................... 2-11
Figure 2-4 Iterative and Incremental development (Modified from Fowler, 1997)..................... 2-12
Figure 2-5 An Address Component (From Barroca et al., 2000)............................................. 2-13
Figure 2-6 Video Copy Class diagram (From Barroca et al., 2000)............................................ 2-17
Figure 2-7 Object illustrations .............................................................................................. 2-17
Figure 2-8 Notation of aggregation.......................................................................................... 2-18
Figure 2-9 Notation of interface.............................................................................................. 2-19
Figure 2-10 Sequence diagram (From Barroca et al., 2000)..................................................... 2-20
Figure 2-11 Interaction diagram for hire video use case (Modified from Barroca et al., 2000).... 2-22
Figure 2-12 Software quality factors (Modified from Barroca et al., 2000)............................... 2-23
Figure 3-1 Different noise sources and noise metrics comparison ............................................ 3-27
Figure 3-2 Aircraft Approach (Modified from Gulding et al., 1999)......................................... 3-29
Figure 3-3 Aircraft Departure (Modified from Gulding et al., 1999)......................................... 3-29
Figure 4-1 Use case diagram for Split airport noise information system................................. 4-38
Figure 4-2 Legend of use case diagram ................................................................................... 4-39
Figure 4-3 Overview of the class diagram .............................................................................. 4-42
Figure 4-4 Class diagram of Split airport noise information system......................................... 4-43
Figure 4-5 Legend of class diagram......................................................................................... 4-44
Figure 4-6 Interaction diagram ............................................................................................... 4-45
Figure 5-1 Geodatabase of Split Airport Noise Information System........................................ 5-52
Figure 5-2 Start user interface of Split ANIS ............................................................................ 5-53
Figure 5-3 Main user interface of Split ANIS.......................................................................... 5-53
Figure 5-4 Noise Monitoring of Split ANIS............................................................................ 5-53
Figure 5-5 Noise Modelling—Radar data .............................................................................. 5-55
Figure 5-6 Noise Modelling—Flight data.............................................................................. 5-55
Figure 5-7 Noise Modelling—Lateral attenuation .................................................................. 5-56
Figure 5-8 Noise Modelling—Atmospheric absorption............................................................. 5-56
Figure 5-9 Noise Modelling—Noise metric ............................................................................ 5-56
Figure 5-10 Noise Modelling—Terrain.................................................................................... 5-56
Figure 5-11 Noise Modelling—Weather data........................................................................ 5-56
Figure 5-12 Airport Noise GIS............................................................................................... 5-57
Figure 5-13 Land use compatibility detection......................................................................... 5-57
Figure 5-14 Noise zone 1....................................................................................................... 5-59
Figure 5-15 Noise zone 2....................................................................................................... 5-59
Figure 5-16 Result of noise affected areas detection............................................................... 5-59
Figure 5-17 Noise prediction ................................................................................................. 5-61
Figure 5-18 Sensitive locations detection.............................................................................. 5-61
Figure 5-19 Result of sensitive locations under noise level..................................................... 5-61
Figure 5-20 Noise and population.......................................................................................... 5-63
Figure 5-21 Bar chart for visualizing noise affected census in 1998......................................... 5-63
Figure 5-22 Pie chart for visualizing noise affected census in 1999......................................... 5-63
Figure 5-23 User defined area............................................................................................... 5-65
Figure 5-24 Result of user defined area............................................................................... 5-65
1. Introduction

1.1. Background

Of all forms of transportation in the world, especially in relatively large cities, aviation is projected to have the fastest growth. With such dynamic growth in aviation, aircraft noise will undoubtedly remain a challenging environmental problem and one that will draw more and more attention from planners, decision makers, ordinary residents and, of course, researchers.

Many airports have been initially developed outside of urban areas they service. However as the time passed by, urbanization has sprawled so that airports are now surrounded by cities. As airport noise problem is becoming worse and worse, International Civil Aviation Organization (ICAO) issued an international standard for aircraft noise. The standard covers the following issues [ICAO, 1988]:

- Possible ways for measuring airport noise;
- Acceptable levels of noise for people;
- Classification of aircrafts regarding the noise which they produce;
- Possible procedures in taking on and off for aircrafts which can result in noise mitigation;
- Control of land use in surrounding areas of airports

Based on the standards, airport proprietors are responsible for airport noise mitigation and land use compatibility detection in airport surroundings. Airport Noise Information System (ANIS) will be a useful tool for airport noise information management.

1.2. Case study area

The case study area in this research is Split Airport, Croatia. Croatia is a south-eastern European country. Split Airport, Croatia, is located in the centre of the Croatian Coast (See Figure 1-1). Split is a famous historical city and this region is characterized by very favourable weather conditions. As a result the airport is capable of accepting all aircrafts year round. It is the most important airport at the eastern side of the Adriatic Sea and represents an indispensable factor for Croatian tourism. Table 1-1 provides general information of the airport. Split Airport got warnings from government that the airport noise was damaging protected buildings (such as old churches) and disturbing residents’ normal lives. A request comes from the airport to use GIS to aid airport noise management. For this reason, the case study area is selected as Split Airport, Croatia.
Split Airport is the only airport in the Republic of Croatia, which has a noise monitoring system for the airport and its vicinity. The system, which has been installed at the end of May 1998, enables registration of noise caused by aircrafts.

After Split airport replaced the Boeing 737-200 aircraft type with less noisy aircraft’s types such as Airbus A319 and A320, the noise monitors have recorded an average 10 dB less relative noise in the vicinity of the airport as a consequence of the aircraft’s types replacement. Although the airport noise control for environment protection has changed significantly, there are still pressure coming from government requires that the airport should take procedures on continuing noise mitigation and noise impact detection.

![Study area](image)

**Figure 1-1 Study area**

<table>
<thead>
<tr>
<th>Table 1 General information of Split Airport, Croatia (From Split airport, 2001)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Surface area:</strong> 911.667m$^2$</td>
</tr>
<tr>
<td><strong>Runway, taxi-ways, apron area</strong> 169.012m$^2$</td>
</tr>
<tr>
<td><strong>Internal roads and parking area</strong> 15.467m$^2$</td>
</tr>
<tr>
<td><strong>Opening hours</strong> 06.30 - 22.00 LT, 24 hours on request</td>
</tr>
<tr>
<td><strong>Site</strong> In Kastela, Trogir - 8 km, Split - 25 km</td>
</tr>
<tr>
<td><strong>No. of employees</strong> 370</td>
</tr>
<tr>
<td><strong>Runway</strong> Direction 05/23, length 2550 m, width 45 m, stop-way 250 m</td>
</tr>
<tr>
<td><strong>Apron</strong> 9 narrow body A/C (B737, DC9) 1 wide body (B747, L1011)</td>
</tr>
<tr>
<td><strong>Alternative</strong> 11 medium range A/C</td>
</tr>
<tr>
<td><strong>Passenger Terminal</strong> Area 8500 m$^2$, annual maximum capacity of 1,500,000 passengers</td>
</tr>
<tr>
<td><strong>Cargo Warehouse</strong> Area 250 m$^2$</td>
</tr>
</tbody>
</table>
1.3. **Research objective**

Following software engineering and object-oriented technologies, the objective of this research is to design and develop a prototype airport noise information system for Split Airport, Croatia. The Split ANIS takes the advantage of GIS spatial analysis and visualization functionalities to model airport noise distributions and detect land use compatibility with airport surroundings.

1.4. **Research questions**

The research is guided by the following research questions:

- What are the user requirements?
- Which functions should be included in this system?
- Which software engineering methodology should be used to design the prototype ANIS?
- How to combine different data for land use conflict detection and airport noise impact analysis?
- How to visualize airport noise and its impact?
- How to test the system?

1.5. **Research methodology**

The research methodology is composed of literature review, system design and development two stages.

1.5.1. **Literature review**

Literature review is used to get a theoretical basis for this research from two aspects as following:

- Theory of software engineering, object-oriented technology, and Unified Modelling Language (UML)
- Theory of airport noise modelling and noise impact analysis

The reason for adopting software engineering concepts for the design and development of the prototype ANIS is that, without systematic software engineering:
• In many cases, customers are dissatisfied with the final system because software development projects are frequently undertaken with only a vague indication of customer requirements. Communication between customer and software developer is often poor.

• System quality is often a problem. Often the so-called waterfall model has been used in system design and development. Each development phase can only start after the previous phase has been completed. For example, the implementation phase starts only when the design phase is entirely finished. It delays the implementation and testing. Once high-risk problems are recognized in later phases, the whole system is in danger.

• System can be very difficult to maintain. System maintainability has not been emphasized as an important criterion for software acceptance.

There is no single best solution for these problems. However by adopting the concept of software engineering, we can try to solve these problems. System engineering methods include many parts as following: user requirements analysis, system design, implementation and testing. A sound quality system can be achieved by combining a series of methods for all phases in software development, better tools for automating these methods, more powerful building blocks for software implementation, and better techniques for software quality assurance [Roger S. Pressman, 1994].

The object-oriented technology is used to achieve flexibility in the development of software, and to ensure software reuse. UML is a standard notation for modelling object-oriented systems.

1.5.2. System design and development

The system design and development is carried out based on iterative and incremental model, which contains four stages within each iteration: user requirements analysis, design, implementation and testing. User requirements gathering and analysis has been done by a fieldwork at the study area through several interviews with airport staff, and trying the existing systems of airport noise as well. UML is used for system modelling. A use case diagram and a class diagram are developed to model system static aspects, a interaction diagram is developed to model system dynamic aspects. ArcObjects and Visual Basic for Application are selected as implementation platforms, and system testing has been guided by a testing plan.

1.6. Thesis structure

The thesis contains six chapters, which are named as introduction, literature review on software engineering and object-oriented technologies, airport noise management and case study area, analysis and design of the prototype airport noise information system, implementation and testing, and conclusion and recommendations.

Chapter 1 gives the background of this research. Research objectives, research questions and research methodologies are elaborated in this chapter as well.
Chapter 2 presents a literature review on software engineering, object-oriented system design and development, and UML.

Chapter 3 explains the basic theory of airport noise and measuring, factors that generate and influence airport noise, laws and regulations relevant to aircraft noise. It also describes existing systems for airport noise measuring and modelling in Split Airport, Croatia, and discusses problems in existing systems.

Chapter 4 discusses object-oriented methodology for ANIS design using UML. A use case diagram and class diagram are developed to model system static aspects, and an interaction diagram and preconditions and post conditions of each system operation are developed to model system dynamic aspects.

Chapter 5 describes implementation of the design of the system. A geodatabase for data management is established and a prototype Split ANIS is developed and tested.

Chapter 6 reviews research objectives and research questions, and presents conclusions and recommendations for future research.
2. Literature review on software engineering and object-oriented technology

2.1. Introduction

To design and develop an Airport Noise Information System (ANIS), we should make a plan in a long run. Since the technology is developing so fast nowadays, adding new functionalities to existing system should be possible and easy, system maintenance should be simple and fast as well. These can be achieved by software engineering and object-oriented technology.

Since Unified Modelling Language (UML) provides a standard notation for modelling and design, it ensures the ease of communication between designer and users as well as designer and system developer.

This chapter presents several theories applied in this research, including software engineering, Object-oriented technology, and UML methodology for software design, system implementation and testing.

2.2. Software engineering and object-oriented technology

Nowadays, software is playing a very important role in almost any business. Since changes in business occur frequently due to all kinds of reasons, these changes will inevitably result in changes of software (modify some functions or add new functions) in order to keep a business competitive. Software engineering concepts and object-oriented technology are particularly suited to develop software that can be easily adapted to the business changes needed [Barroca et al., 2000]. The ANIS proposed in the research may need to be adapted by added other functionalities according to user requirements in the future. It may also need to be integrated in other airport management systems such as, an airport parking area management system, an airport runway use system, etc. Therefore, software engineering and object-oriented technology are adopted in this research to ensure the ANIS is more adaptable to changes.

There are many significant factors that will influence the design and development of a high quality software system. From the manager’s point of view it is crucial to have well-defined activities that make it possible to estimate when a final product will be completed and the effort required achieving it. Form the developer’s point of view it is important to have well-defined tasks and clear outcomes, and to know which documents and artifacts should be produced by each of the activities undertaken.
From the customer’s point of view it is fundamental to be reassured that the required product is the one being built and that it will be delivered on time and within budget. To achieve above, software engineering and object-oriented software development are needed. Roger S. Pressman (1994) explains it as follows: “Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines”.

2.2.1. Software development models

There are several development models used for software engineering. Two typical models are described here, namely waterfall development model and iterative and incremental development model.

2.2.1.1 Waterfall model

The waterfall model is a traditional software development model, shown in Figure 2-1.

![Waterfall software development model (Barroca et al., 2000)](image)

The waterfall model encompasses typical sequential phases known as requirements analysis, design, implementation and testing. Each phase can only start after the previous one has finished. It requires that a set of user requirements is defined beforehand and they should be kept unchanging, and user requirements must be completed within a short time. All of these are almost impossible practically.

The waterfall model is the oldest and most widely used paradigm for software developing. Although it has many weaknesses, it provides a template into which methods for analysis, design, coding, testing and maintenance can be placed [Roger S. Pressman, 1994].

2.2.1.2 The iterative and incremental development model

A recently more popular and powerful model is the iterative and incremental development model. Larman (2002) defines the model as follows: “The iterative and incremental development model is based on the successive enlargement and refinement of a system through multiple iterations, with cyclic
clique feedback and adaptation as core drivers to converge upon a suitable system. The system grows incrementally over time, iteration by iteration” (see Figure 2-2).

Barroca et al. (2000) describes this model as: “incremental development involves partitioning up-front the intended functionality; some or all of the subsequent development activities can be carried out in parallel. The final product results from the total integration of the partitions”. The term *iterative* means the repetition of one or more activities; the term *incremental* means that development proceeds from an initial subset of the requirements to more and more complete subsets, until the whole system is produced. Although there are still analysis, design, implementation and testing cycle within this development model, it is much more flexible compared to that in waterfall development model. In this model, each cycle is short and provides feedback for the next cycle, in which a more refined and enhanced development is achieved (see Figure 2-3).

According to Fowler (1997), “the essence of incremental development is the development stages that are a series of increments. Each increment builds a subset of the full software system. An increment is
a full lifecycle of analysis, design, implementation, testing, and integration. It may be delivered to the user, but not necessarily”.

Execution of the development model is preceded by an exploration phase that aims to understand those areas that are most unknown. Typically the phase includes:

- Finding use cases
- Developing an outline model of the problem domain
- Developing an outline architecture which describes how the major pieces of technology fit together

![Diagram](image-url)

**Figure 2-4 Iterative and Incremental development (Modified from Fowler, 1997)**

To summarize the above, we can see that although Larman, Barroca et al., and Fowler use different diagrams and explanations for the iterative and incremental development model, the ideas are quite similar. As each iteration deals with a small set of requirements, the system’s complexity is kept under control. Early feedback is obtained from a partial implementation, high risks are likely to be identified in the early stages.

### 2.2.2. Object-oriented technology

#### 2.2.2.1 What is object-oriented?

Barroca et al. (2000) describe object-oriented as follows: “A software system whose basic structuring is around things rather than around actions is said to be object-oriented”. The reason is that although businesses change frequently, it has been observed that the things, which are manipulated by computer systems, stay quite stable, while the ways in which they are used change rapidly. This leads to structuring software around things that are manipulated instead of around actions that manipulate them. For example, we may develop one unit of software for payment, another unit of software for order etc. No matter how business will change, these units are still needed; only the way of how they are coupled will change.

To define an object, we need firstly to define a class. A **class** is a category or group of things that have similar attributes and common behaviours. An **attribute** is a property. An **object** is an instance of a class—a specific thing that has specific values of the attribute and operations. An **operation** is a function or transformation type that is applicable to objects of the class. Only operations specified by the class can be applied to objects in that class. An operation may also involve objects of other classes, as
specified by parameters in the operation signature. An **interface** is a collection of operations that a class carries out, which designers might want to use again and again throughout their model.

Designing and programming in an object-oriented language consists mostly of writing definitions of classes, rather than of individual objects. One of the main challenges is to identify the properties that must be represented for one system.

Low coupling between classes and high cohesion within classes are very important design principles. Coupling is a measure of how strongly one class is connected to, or relies on other classes. Low coupling ensures one classes to be independent as much as possible, which provides great opportunities for reuse. Cohesion is a measure of how strongly the responsibilities of a class are related. High cohesion is one whose responsibilities are related to one particular functional area. Low coupling and high cohesion result in software that is easier to extend and maintain, and provide great opportunities for reuse.

It can be said that object-oriented software development is a component-based development. In generic terms, a component is just a unit of software that can be put together with other units to make a larger system. Software engineers took their inspiration from how other industries worked with components. The idea of component in software is similar to that of electrical components that are wired together to form larger components. The wires have the same functions as interconnections in software.

Component-based software has a number of points of interconnection that are named interfaces. A required interface defines the operations that the component will need to request from other components, while a provided interface defines the operations that the component offers to other components. Components are completely encapsulated since there is no way to deal with components except through their interfaces.

![Figure 2-5 An Address Component (From Barroca et al., 2000)](image)

A component should either "plug" into another system or into an adapter and the adapter "plugs" into another system. So the specification of a component and specification of the connectors are very im-
important. Specification of the component should include the detailed description of provided interface, required interface, vocabulary used, the validation code, and any non-functional properties of the component, such as security, performance, and reliability. Specification of the connectors includes explicit call and return and workflow where objects are transferred between components etc [Barroca et al., 2000].

2.2.2.2 Why object-oriented?

Within the context of software engineering, an object has two main roles: as abstraction and decomposition. Firstly, an object is the natural representation of some entities in the problem domain of real world. An object helps abstraction by offering an encapsulated unit for requirements analysis. Secondly, an object provides a modular unit for decomposition by dividing a program into a set of components.

An object encapsulates data (the attribute values that define it), operations, other objects (composite object), constants (set values), and other related information. When data are entirely hidden behind a wall of procedures that alone can manipulate them, they are said to be encapsulated [Barroca et al. 2000]. Encapsulation is one of the central concepts of object-oriented technology. Encapsulation means that all of this information is packaged under one name and can be reused as one specification or program component [Roger S. Pressman, 1994].

An object has a set of variables, with a group of procedures that can act on the data. Actually all programs contain procedures and data. The profound difference is the tight syntactic coupling between data and procedures in an object-oriented language. In an object-oriented system, the only procedures that are allowed to access the data in an object are the procedures attached to that object. So it is guaranteed that no other part of the system could touch the data.

As an instance of a class, an object has all the characteristics of its class, this is called inheritance. Not only can an object inherit from a class, a class can also inherit from another class, i.e. inheritance allows different classes to share the implementation of operations. By inheritance, we can define one class basically as similar to another, and just implement the ways in which they differ. For example, class Employee and Customer has many similarities, which can be abstracted as a class Person, but both have their own attributes different from each other. In this case, class Person is super class, and classes Employee and Customer are subclasses. One purpose of inheritance is to allow the implementations of two different classes to share code. Inheritance can be single inheritance as described above; it can also be multiple inheritances, in which a class is inherited from several super classes.

Sometimes an operation has the same name in different classes, in object-orientation, each class “knows” how that operation is supposed to take place, this is called polymorphism. It is one of the most powerful concepts of object-oriented software design. One can add new classes that support the same interface, and they will work harmoniously with the old. One can also change the implementation within a class, and no client will be affected because they use the class solely through its interface.

One of the advantages claimed for Object-oriented development is its amenability to reuse. Reuse is the process of creating software systems from existing assets, rather than building software systems
from scratch. Encapsulation encourages better designs that can be reused in a more reliable way, supplying us with the exact knowledge of which operations access which data. Polymorphism restricts the assumptions made of an object to a well-defined interface. It allows for components to be "plugged" in together, each having a different implementation for the same interface. Inheritance allows a developer to reuse a class whose behaviour provides some of the behaviour of the new class required, so that only the different parts need to be designed. These three concepts of Object-oriented technology can be used to achieve flexibility in the development of software, and to ensure software reuse [Barroca et al., 2000].

2.3. Unified Modelling Language (UML) for software design

In this research, Unified Modelling Language (UML) is used for ANIS modelling and design. The reason for it is that UML dominates recently object-oriented techniques. UML is the result of the merging of several object-oriented methods that appeared during the 1980s and early 1990s in an attempt to end the diversity of object-oriented methods, it is becoming widely used and accepted. UML is a standard notation for modelling object-oriented systems. It's important to realize, however, that the UML is only a standard notation. Essentially it defines a number of diagrams to describe a system, and what these diagrams mean. It does not describe the process developers use to go about building software. Such a process description includes a list of tasks that need to be done, in which order they should be done, the deliverables produced, the kinds of skills required for each task etc [Fowler, 1997].

Although the traditional methodology consists of both notations and a method, the idea of using UML is that by standardizing on the notation, software developers can better communicate to provide all the deliverable artifacts in different design stages.

UML contains several important diagrams including use case, class, sequence, and interaction diagrams etc., which are useful in different software development phases.

No matter which system development model is used, it always contains analysis, design, implementation, and testing four phases. The analysis phase can be partitioned further into domain modelling and software specification.

2.3.1. Analysis—Domain modelling

Analysis emphasizes on the investigation of problems and requirements, rather than a solution. Object-oriented analysis (OOA) emphasizes on finding and describing the objects or concepts—in the problem domain. Object-oriented analysis is concerned with creating a description of the domain from the perspective of classification by objects. A decomposition of the domain involves an identification of the concepts, attributes, and associations that are considered noteworthy. The result can be expressed in a domain model, which is illustrated in a set of diagrams that show domain concepts or objects [Larman, 2002].

Domain modelling is concerned with understanding and modelling a situation independently of a decision to use a software system to deal with it. A domain model is a representation of the main concepts
in the real-world problem [Barroca et al., 2000]. UML Use-case diagrams and class diagrams are important tools in this stage.

2.3.1.1 Use case diagram

Use cases are widely used mechanism to discover and record requirements (especially functional). They influence many aspects of a project, including OOA and Object-oriented Design (OOD). Use-case diagram provides a way of describing the external view of the system and its interactions with the outside world. A use case is a description of a system’s behaviour from a user’s standpoint. For system developers, it’s a powerful technique for gathering system requirements from a user’s point of view. Drawing a use case diagram is a vital part of object-oriented development even though use case itself is not object-oriented. System developers should use them any time they want to understand the requirements of proposed systems. Each use case needs a name and a few paragraphs of description. They are central to planning the incremental development process. They should also drive system testing [Larman, 2002], [Fowler, 1997].

Within a use case diagram, an actor is something with behaviour, such as a person (identified by role), computer system, or organization. Each use case can involve more than one actor, and each actor may be involved in many use cases. A use case diagram shows the high level functionality of the business expressed as use cases, and it also shows which actors are involved in which use cases by association between them. Use-cases are then typical interactions that the actor has with the system.

2.3.1.2 Class diagram

The class diagram is a central modelling technique that runs through nearly all object-oriented methods. This diagram describes the types of objects in the system and various kinds of static relationships that exist between them. There are three principal kinds of relationships: associations (a noise monitor uses radar data for correlation), subtypes (Boeing aircraft is a type of aircraft) and aggregation (an engine is part of an aircraft).

Identifying a set of objects or conceptual classes is at the heart of object-oriented analysis, and well worth the effort in terms of payoff during the design and implementation work. The identification of conceptual classes is part of an investigation of the problem domain (See Figure 2-6).

The following definitions of elements described in the diagram are summaries derived from Schmuller (1999), Barroca et al. (2000), Fowler (1997) and Larman (2002).

- A **class** is expressed by a rectangle with three parts inside. The first part is the class name. For example, **VideoHireBusiness** is a class name. The second part contains all the attributes of the class. For example, **address** is an attribute. The third part contains all the operations within this class, for example, **hireTransaction** is an operation.
• An **object** is an instance of a class, with specific values of the attribute and operations, see Figure 2-7. The notation of object is a colon plus underlined class name, for example, :VideoHireBusiness is one of instances of VideoHireBusiness class.

<table>
<thead>
<tr>
<th>:VideoHireBusiness</th>
</tr>
</thead>
<tbody>
<tr>
<td>address=&quot;Hengelostraat 99&quot;</td>
</tr>
<tr>
<td>TaxReference= 5</td>
</tr>
</tbody>
</table>

**Figure 2-7 Object illustrations**

• An **operation** is a function or transformation type that is applicable to objects of the class. Only operations specified by the class can be applied to objects in that class. An operation may also involve objects of other classes, as specified by parameters of the operation signature.
• **Multiplicity** defines how many instances of a class can be associated with one instance of another class. The multiplicity value communicates how many instances can be validly associated with another, at a particular moment, rather than over a span of time [Larman, 2002]. The following show the meaning of notations of multiplicity from [Barroca et al., 2000]

<table>
<thead>
<tr>
<th>Multiplicity</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exactly one</td>
</tr>
<tr>
<td>0..1</td>
<td>Zero or one</td>
</tr>
<tr>
<td>p..q</td>
<td>Between p and q</td>
</tr>
<tr>
<td>* or (0..*)</td>
<td>Zero or more</td>
</tr>
<tr>
<td>1..*</td>
<td>One or more</td>
</tr>
</tbody>
</table>

• **Associations** represent relationships between instances of classes. For example, association between :Member and :Loan means one member may make zero or 3 Loan(s), but one Loan can only be made by 1 Member.

• **Constraint** is a condition imposed on the elements of the model. Constraint is not behaviour, but some other kind of restriction on the design or project. It is also a requirement, but is commonly called “constraint” to emphasize its restrictive influence. UML uses the brace { } notation to show constraints on the structural model. For example, {Only special member can make reservation}

• **Generalization** is the activity of identifying commonality among concepts and defining superclass and subclass relationships. For example, *SpecialMember and StandardMember are very similar, both have common attributes as MembershipCode, Name, Address, Date and NumberofLoan, but they do have their own attributes as well, so it is useful to generalize them into class hierarchy to ensure inheritance possible.* The notation is a line between superclass and subclass with an open triangle on the line near superclass.

• **Aggregation** is another kind of association among objects. Sometimes a class consists of a number of component classes; this special type of relationship is called aggregation. For example, keyboard, monitor and mouse are part of computer (See Figure 2-8). The notion in UML is a line that joins a whole to a component, with a black diamond on the line near the whole.

![Figure 2-8 Notation of aggregation](image)
• The notation of interface is a small circle connected by a line to a class (Figure 2-9).

Figure 2-9 Notation of interface

2.3.2. Analysis—Software specification

Based on domain modelling, software specification is one step further in the process of understanding the domain. Specification modelling assumes that a decision has been taken to develop a software system to solve a specific problem within a business. A specification model represents software elements used in a software solution to a problem, and is mainly concerned with the definition, at a high level of abstraction, of the services provided by the software [Barroca et al., 2000]. In this stage, it is still considered what the software system will do instead of how to do. The diagrams described below play a role in this phase.

2.3.2.1 Detailed use case diagram

With the basis of the use case diagram defined in Section 2.3.1.1, in this stage, we begin to define dependency relationships between use cases. A dependency relationship states that a change in the specification of one thing may affect another thing that uses it. It has two types:

• A use case includes another when the included use case is part of its behaviour.
• The extend relationship is used to represent alternative behaviour; a use case can extend an existing one by defining an optional or conditional behaviour at the point of extension.

The boundary between users and the system to be developed needs also to be defined. The system boundary is represented by a box around the use cases. It defines what is part of the functionality of the software system and what is left out. The use case model for the domain is used as a starting point to identify actors and use cases from a software specification perspective.

2.3.2.2 Analysis model----relatively detailed class diagram

When building an analysis model, we focus on the structure of the software and identify which domain objects will have a software specification, which attribute should be defined in the classes and which relationships exist between classes. At this stage, attributes at fairly high level are defined to get an analysis class. An analysis class is concerned about the relationships at conceptual level while the navigability of the associations is still not important for the moment.
2.3.2.3 Behaviour model

Behaviour of the objects represents all the ways in which objects can change as a result of the way they are used, in use cases. A behaviour model is achieved by using the following approaches.

- A sequence diagram is used to represent a scenario.

![Sequence Diagram](From Barroca et al., 2000)

Sequence diagram shows the temporal ordering of events. In this example, it shows a sequential series of transactions for a scenario in video business.

- Pre- and post-conditions are to be used to describe systems operations.

Preconditions describe what kinds of conditions must be possessed in order to trigger system operations while post-conditions describe what kinds of conditions can be achieved after executed system operations. They are important tools for model dynamic aspects of system.

2.3.3. Design

So far, a software system is dealt as a black box that responds to events from the actors in the environment. The response from the system results from a serious of interactions between objects. The main focus of design is to look inside the black box to identify how these objects need to interact to achieve the expected behaviour of the system, and determine the responsibilities that each of the class
of objects should take. Design emphasizes a conceptual solution that fulfils the requirements, rather than its implementation. Object-oriented design is concerned with defining software objects and their collaborations. The elements described below are important in this phase.

2.3.3.1 More detailed use cases: scenarios

Details of use cases are described by giving possible scenarios for each one and representing the interactions of a scenario using system sequence diagrams. At this stage, all possible ‘unusual’ behaviour that arises from unusual events caused by actors in the environment should also be described as scenarios.

2.3.3.2 User interface design

The design of user interfaces should be guided by user requirements and design principles for user interfaces, such as using simple and natural dialogue, screen layout should be consistent, allowing recovering from user’s errors and incorporating some form of user guidance.

Storyboards are used to produce prototypes of user interfaces and to facilitate discussion with users. Storyboards are sort of diagrams used to describe how the user interfaces look like. It helps the user and the software developer to visualize user interfaces quickly and cheaply and facilitates discussion and allows the screen design to be considered in detail.

2.3.3.3 The behaviour model

In the design stage, the behaviour model should take into account of following aspects, it will describe how objects within the system interact with each other to fulfil the pre and post condition of each of the system operations; assign to each class of objects the responsibilities that it has to affect each system operation and identify the operations needed to carry out the responsibilities assigned to each class [Barroca et al., 2000].

A common notation to illustrate collaborations between objects is the interaction diagram. Interaction diagrams are models that describe how a group of objects collaborate in some behaviour. The diagrams show a number of example objects and the messages that are passed between these objects within the use case (See Figure 2-11).

This interaction diagram describes actor Cashier sends message hireTransaction() to object VideoBusiness, and VideoBusiness sends a message createLoan() to object VideoCopy, VideoCopy sends a message create() to object Loan, and so on. It shows clearly the messages passing between these objects. In the same time, it also shows which object takes what responsibility.

A responsibility can be defined as an obligation on a class to perform a particular service. A goal of design is to embody encapsulation in a design before it is implemented. A critical, fundamental ability in OOA/OOD is to skilfully assign responsibilities to software components. Because it is one activity that must be performed, either while drawing a UML (design) diagram or programming, and it strongly influences the robustness, maintainability, and reusability of software components.
2.3.3.4 Design class model

The product of design is the design class model for the software system under development. Based on the analysis model and the specification of the system operations developed during the analysis phase, interaction diagrams are developed to describe how the preconditions of each system operation need to be fulfilled and how each system operation is fulfilling its post-conditions effects.

2.4. Implementation and testing

2.4.1. Introduction

The aim in implementation is to develop classes of objects where encapsulation is respected. The reason behind it is that encapsulation is the basis of reusable software components. Object-oriented programming should promote the construction of software by reusing software components. When a suitable component is not available, a new component should be developed both for that application and for reuse by other future applications.
2.4.2 Software quality

Software quality is a complex mix of factors that will vary from different applications. McCall et al. (1997) have proposed useful software quality factors, and Barroca et al. (2000) modified it as in Figure 2-12.

They focus on three aspects of a software product as follows:
- Its operational characteristics (product operations);
- Its ability to evolve (product revision);
- Its adaptability to new environments (product transition).

Description of each software quality factor is given below.

Correctness: the software should perform its tasks exactly as defined by its specification.
Reliability: the software should perform its intended functions.
Efficiency: the ability of a software system to place as few demands as possible on hardware resources.
Integrity: a high degree of security against unauthorized access to data.
Usability: the ease with which people with varying backgrounds and abilities can learn to use the software.
Maintainability: the ease with which it is possible to understand and correct defects in the software.
Flexibility: the ease with which it is possible to adapt an operational system.
Testability: the ease with which a system can be tested to ensure that it meets its specification.
Portability: the ability of the software to be transferred from one platform to another.
Reusability: the ability of the software to be reused in another applications.
Extensibility: the ability of the software to be extended beyond its original specification.
Interoperability: the ease with which it is possible to couple the software with another system.
Robustness: the ability of a software system to react appropriately to abnormal conditions.

Figure 2-12 Software quality factors (Modified from Barroca et al., 2000)
It is hard to rank software quality factors in order of importance, but reusability and extensibility are main goals in object-oriented software.

2.4.3 Testing

Testing is the process whereby test data is run on a program and its response compared with the expected response. It should be a systematic activity and in object-oriented development, it starts actually from the very beginning as the use cases are defined. Each use case is a good guide for a set of tests to be developed for the final system. Tests check for correct behaviour against a detailed specification. Test criteria are also set when specifying system operations by giving their pre- and post conditions. By allocating responsibilities to classes during design, we are defining the methods that need to be tested individually and in combination.

The final product testing includes both system developer’s acceptance testing and user’s acceptance testing.

2.5. Summary

This chapter describes the basic theory for this research, which includes software engineering, object-oriented technology, UML for software system design, system implementation and testing.

Software engineering and object-oriented technology are emphasized in this chapter to ensure the ANIS proposed in the research is reusable and easy to maintain; UML is described for developing all kinds of design diagrams in different system design phases, it helps to document the development systematically and concisely. System implementation and testing descriptions are guidance for implementation and testing the proposed system. This chapter plays the essential role in the evolution of this research.
3. Airport noise management and case study area

3.1. Introduction

In general noise is any forms of unwelcome sound. How people perceive loudness or noisiness of aircraft noise depends on several measurable physical characteristics of the noise. This chapter first discusses different aircraft noise metrics and their suitable use under different conditions. Next, relevant laws about airport noise of United States Federal Aviation Administration and Croatia are introduced. Factors affecting airport noise are discussed. The two existing systems in Split Airport for noise monitoring and modelling as well as their problems are introduced in the last part of this chapter.

3.2. Airport noise measurement

3.2.1. Noise

The terms noise and sound are used interchangeably. Noise is unwanted sound; noise level is defined in the same way as sound level. The unit of noise is decibels (dB); a change of 3 dB is noticeable to most people. There are four main factors that influence how the sound is perceived by human being. These factors are: Intensity: a ten decibel increase in intensity may be considered a doubling of the perceived loudness or noisiness of a sound. Frequency Content: sound with concentration of energy between 2,000 Hz and 8,000 Hz is perceived to be more noisy than sounds of equal sound pressure level outside this range. Changes in Sound Pressure Level: sound that is increasing in level is judged to be somewhat louder than those decreasing in level (consider police and emergency vehicle sirens). Rate of Change of Sound Pressure Level: impulsive sound (reaching a high peak very abruptly, such as jack hammers) are usually perceived to be very noisy [Woosley, 2000].

The expression of noise is logarithmic, not linear. For example, the ratio of barely audible to painful sound pressures is 1:1,000,000,000,000 (10 to the power of 12). If we use a linear scale the numbers of sound would be unmanageable, besides people do not hear sound in that way [Woosley, 2000].

3.2.2. Airport noise metrics

According to Woosley [2000] and [Gulding et al., 1999], the following are the most common metrics used for describing airport noise. They are listed in the order from simple to complex. All of them be-
long to A-weighted noise metric that is for standard noise analysis. A-weighted means that aircraft noise spectra are modified by depressing noise levels in low and high frequency bands, which is close to the frequency response of the human ear. A-weighted noise metric is almost universally accepted.

- Maximum noise level (Lmax) is the maximum noise level for a given aircraft noise event.
- Sound Exposure Level (SEL) is a measure of the physical energy of the aircraft noise event that takes into accounts both intensity and duration; SEL is typically used to compare noise events of varying durations and intensities.
- Equivalent Sound Level (Leq) is the steady sound level over any specified period. Leq is used to identify the average sound level over a given period of time. Leq is the appropriate metric for examining the impact of aircraft noise on schools and similar receptors where sleep-interference is not a factor.
- Day-Night Average Sound Level (DNL) is introduced as a simple method for predicting the effects on a population of the average long-term exposure to noise. It is a noise measure used to describe average aircraft noise levels over a 24-hour period, typically an average day over the course of a year. It is based on the Equivalent Sound Level, with an added correction for nighttime noise events. DNL penalizes aircraft operations that occur between the hours of 10 p.m. and 7 a.m. by 10 decibels to account for increased annoyance when ambient noise levels are lower and people are trying to sleep. DNL can be determined for individual locations or expressed in noise contours. DNL is currently the only measure for aircraft noise analysis that is accepted by the U.S. Federal Aviation Administration.
- Yearly day-night average sound level (YDNL) means the 365-day average day-night average sound level. The symbol for YDNL is \( L_{\text{dn}} \).

From the Figure 3-2, we can see that we get different noise levels from the same noise source by using different noise metrics. For example, for the same noise source Impulse Noise, its SEL=100dB, Leq=105dB, and Lmax=106; for noise Aircraft Flyover, SEL=100dB, Leq=82dB, Lmax=87; for Roadway Noise, SEL=100dB, Leq=71dB, Lmax=73. We also can perceive these figures in another way: compare three very different types of sounds in different Leq and Lmax with same SEL=100dB. Impulse noise (clap): Lmax=106; Aircraft flyover: Lmax=87; Roadway noise: Lmax=73.
3.3. Factors affecting airport noise

Airport noise is a complex physical phenomena which is generated and influenced by many factors as follows:

- Types and numbers of aircrafts operating at the airport

  The consideration of aircraft types usually requires aggregating aircrafts having similar performance and noise characteristics into representative categories [SAE, 1986]. For example aircraft type A320-211 has the following characteristics: engine type: jet; number of engines: 2; Maximum gross takeoff weight: 73482 kg; Maximum gross landing weight: 64410 kg; Maximum landing distance: 893 meters; static thrust: 25800 pounds.

  Engine type includes: turbojets, turbojet propeller-driven, and piston-engine propeller-driven [Gulding et al., 1999]. Thrust is produced by aircraft’s jet engines and allows an aircraft to climb. The more thrust a jet can produce, the steeper the angle it can climb. The thrust of the jet engine is also used to slow down the aircraft on the landing roll. Components of the engine, called thrust reverse, are activated by the pilot after touchdown. This is why jet engine noise becomes louder after landing [Esser, 1997].

- Aircraft’s operations and flight procedural profiles

  Aircraft’s operations include five different types: approach, departure, circuit flight, touch and go, and flyover. Accordingly there are five types of flight procedural profiles. Within each flight procedural profile there are maximum nine types of procedure steps described as following [Gulding et al., 1999]:

  - Takeoff- start-roll to takeoff rotation
  - Climb- departure climb to final altitude at constant calibrated airspeed
  - Accelerate- departure climb and accelerate to final speed
• Level- maintain altitude and speed
• Level stretch- special step used to designate where to stretch a circuit flight profile to fit a touch-and-go track
• Cruise Climb- climb at constant angle to final altitude
• Descend- descend at constant angle to final altitude
• Land- land and roll a given distance
• Decelerate- brake with starting thrust for a given distance

Five types procedural profiles are combined with different steps depending on the flight operation, and each procedure produces different noise because of the different engine power setting and flap setting. Flap is part of the wing on an aircraft that can be lifted in flight to change the aircraft’s upward direction.

Take the departure profile as an example:
1. Takeoff using Maximum takeoff thrust and extended flaps.
2. Climb to 1000 feet (308.4 meters) using Maximum takeoff thrust and takeoff flaps.
3. Three steps of accelerate at different speed and different altitude.
4. Four steps climbs at different altitude and with different thrust and flap settings.

• Aircraft flight paths, tracks and runways

Flight paths are described by ground tracks and vertical flight profiles. Tracks can be represented by pieces of straight segments that are called track segments. An approach track starts in terminal airspace and finishes at the displaced approach threshold on the runway (see Figure 3-3). A departure track starts at the displaced takeoff threshold on the runway, and it finishes in terminal airspace (see Figure 3-4). A touch-and-go track starts and finishes at the displaced approach threshold on a runway. An over flight track starts and finishes in terminal airspace.

The essential intent of design flight path is to route aircraft departures and arrivals over coastal and water corridors instead of residential area to reduce the noise disturbance. The design would utilize extended and/or accelerated climb paths for turbojets, bringing them to higher altitudes (resulting in reduced noise impacts at ground level) prior to over flying along the ultimate intended destinations [FAA, 1976]. Some flight procedures use turns during climb out step to avoid residential areas. Prevailing flight patterns and runway usage should be established. Certain runways should not allow to be used for nighttime operations [SAE, 1986].

• Scheduled flight (day flight, night flight)

Because of the potential of aircraft operations to create sleep disturbance at night, noise investigators and regulatory agencies recognize that more restrictive noise standards should be applied during the evening and early morning hours (i.e., 10:00 p.m. to 7:00 a.m.) as compared to daytime hours (i.e., 7:00 a.m. to 10:00 p.m.).
- Airport status and weather conditions

Local terrain of airport surroundings, runway elevation and runway gradient can also influence noise distribution. There are several factors related to weather conditions such as air temperature, wind speed, humidity, and atmospheric pressure. Atmospheric absorption is one of the important parameters that affect noise distribution.

- Positions of observation

The position of observation relative to flight path influences the noise effect very much. For example, observing location directly under a flight path has higher noise level compared to locations
on the side of the flight path. This is because of the ground absorption effect. This effect is called lateral attenuation. It is one of the important parameters in airport noise calculation.

3.4. Laws and regulations relevant to airport noise

Aircraft noise had become a growing problem in the 1960’s with the introduction of jet aircraft and the rapidly increasing number of commercial aircraft operations in the world. Aircraft noise, and its adverse impacts on residential and other noise sensitive land uses, was recognized as a major constraint on the further development of the aviation system, limiting the further construction and expansion of airports. For this reason, many countries issued laws and regulations addressing airport noise problems, and they are still working on new policies to deal with new airport noise problems.

3.4.1. Aviation Noise Abatement Policy of the United States

The airports in the United States are busier than those in the rest of the world. Therefore, there are more people who are affected by airport noise there. The United States began to deal with airport noise issues very early compared to other countries. And it has relatively complete airport noise laws and regulations; many other countries are following or referring to them for tackling their own countries’ airport noise problems.

Although the primary mandate of the Federal Aviation Administration (FAA) of United States is aircraft and flight safety, the FAA directly addresses airport noise issues through aircraft and engine noise certification, and research. The first comprehensive Aviation Noise Abatement Policy (ANAP) was issued by the Secretary of Transportation and the Administrator of FAA on November 18, 1976. While the number of Americans exposed to significant levels of aviation noise has been dramatically reduced since the 1976 Policy was issued, a large number of people still remain impacted. FAA issued the Aviation Noise Abatement Policy 2000 to readdress airport noise issue. Building on past successes in the area of aviation noise, the FAA’s ANAP goals are: (a) Continue to reduce aircraft noise at the source; (b) Use new technologies to mitigate noise impacts; (c) Encourage development of compatible land uses; (d) Design air traffic routes and procedures to minimize aviation noise impacts [FAA, 2000].

ANAP defines the authorities and responsibilities of government and airport proprietors as follows:

“The Federal Government has the authority and responsibility to control aircraft noise by the regulation of source emissions, by flight operational procedures, and by management of the air traffic control and navigable airspace in ways that minimize noise impact on residential areas” [FAA, 1976]. “Airport proprietors are primarily responsible for planning and implementing action to reduce the effect of noise on residents of the surrounding area” [FAA, 1976].
3.4.2. Airport noise compatibility planning in the USA

Federal Aviation Regulation (FAR) of FAA Part 150 has been going at numerous airports throughout the United States. The goals of FAR 150 are (a) to measure noise at airports to better delineate noise impacting urban neighbourhoods, and (b) to determine exposure of individuals to noise that results from the operations of an airport.

The FAA places great emphasis on reducing the number of persons residing in areas of significant noise exposure around airports. Each airport with areas of significant noise exposure outside its boundary is encouraged to evaluate its current and projected noise levels, and to develop a program that both reduces the number of persons significantly impacted by noise, and prevents new non-compatible development from occurring.

According to [PCR Environmental, 2000], FAA issued the criteria for land use compatibility with yearly day-night average sound levels. Table 3-1 shows whether the different land use types are compatible to the different noise levels or not. For example, land use of residential is compatible with noise level below 65, but not compatible with noise levels of 65-70, 70-75, 75-80, 80-85, and over 85.

Table 2 Land use compatibility with different noise level (From PCR Environmental, 2000)

<table>
<thead>
<tr>
<th>Land Use</th>
<th>Yearly day-night average sound level (L_{dn}) in decibels</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Below</td>
</tr>
<tr>
<td>Residential, other than mobile homes and transient lodging</td>
<td>Y</td>
</tr>
<tr>
<td>Mobile home parks</td>
<td>Y</td>
</tr>
<tr>
<td>Transient lodgings</td>
<td>Y</td>
</tr>
<tr>
<td>Schools</td>
<td>Y</td>
</tr>
<tr>
<td>Hospitals and nursing homes</td>
<td>Y</td>
</tr>
<tr>
<td>Churches, auditoriums, and concert halls</td>
<td>Y</td>
</tr>
<tr>
<td>Government services</td>
<td>Y</td>
</tr>
<tr>
<td>Transportation</td>
<td>Y</td>
</tr>
<tr>
<td>Parking</td>
<td>Y</td>
</tr>
<tr>
<td>Offices, business and professional</td>
<td>Y</td>
</tr>
<tr>
<td>Wholesale and retail--</td>
<td>Y</td>
</tr>
</tbody>
</table>
3.4.3. Laws and regulations relevant to aircraft noise of Croatia

In Croatia, only an “Air traffic law” [Croatian Parliament, 1998] and “Aircraft noise law (draft) of Croatia” [Zagreb, 1994] are currently available. These laws give general requirements for aircraft noise control. They are mostly based on International Civil Aviation Organization (ICAO) Environmental Protection, Annex 16. Detailed specifications will be developed in future regulations.
3.4.3.1 Air Traffic Law [Croatian Parliament, 1998]

The Croatian air traffic control agency, aircraft pilots, aircraft operators and airport operators are obliged to take measures on noise protection, according to regulations written in accordance with the Law. Operators of military jets and of civil flights airports are obliged to continuously measure noise at airport and its vicinity caused by aircraft taking on and off.

Conditions and methods of noise measurement are stated in regulations written in accordance with the Law. (5th part Aircraft noise protection, Article 141, Noise measurement)

Based on noise measurements stated in Article 141, the noise protection areas are areas with equivalent aircraft noise zone 1 that is greater than 67 dB but less than 75 dB and noise zone 2 that is over 75 dB, and they have to be stated in submitting documents.

Conditions and methods of stating noise protection areas, and construction conditions in these areas are stated in regulations written in accordance with the Law (Noise protection areas, Article 142).

3.4.3.2 Aircraft Noise Law (draft) of Croatia

The Aircraft Noise Law (draft) [Zagreb, 1994] proposed future standards for aircraft noise management of Croatia airports. It covers the following issues in terms of the extension of noise protection area, noise measurement, noise detection, and inhibition of construction within noise protection area:

- Noise protection areas are divided into two protection zones according to noise level. Protection zone 1 covers the area with equivalent permanent aircraft noise level over 75 dB, and protection zone 2 covers the remaining noise protection area, that is between 67 dB and 75 dB.

- Equivalent permanent aircraft noise level at any point around an airport (measuring points) is stated according to the highest noise level and the longest lasting of noise caused by aircraft flying over that particular point. Stating of noise level is based on month intervals of most intensive traffic. Daytime flights (from 06:00am to 22:00pm) are evaluated differently from night flights (from 22:00pm to 06:00am).

- The biggest noise level at a noise measuring point, for every aircraft passes, is stated according to noise produced, taking into consideration flight path distance and noise spreading conditions. Noise lasting time at measuring points is the time during which noise level passed the level, which is 10 dB lower than the highest noise level.

- Airport noise protection area must be repeatedly detected if the change of facilities or change of use of airport causes significant change of noise impact in airport surroundings. Significant noise impact change is the one that cause the increase of equivalent permanent noise level for more than 4 dB at outer noise protection limits.
• Within noise protection area hospitals, old people homes, holiday resorts, schools and other facilities that request noise protection must not be built. Residential area must not be built in noise zone 1.

To summarize above, airport proprietors are compulsory to take procedures to mitigate aircraft noise impact and to repeatedly detect the land use compatibility in airport surrounding areas. Detect noise zone 1 and noise zone 2 are essential tasks of the airports in Croatia. Aircraft noise law also states the criteria for measuring aircraft noise.

### 3.5. The existing airport noise monitoring and modelling systems in Split Airport, Croatia

Currently Split Airport has two existing systems for airport noise monitoring and noise propagation modelling. They are Noise Monitoring System (NMS) and Integrated Noise Model (INM). From Section 3.4.3 we can see that airport proprietors are obliged to work on noise disturbance report, land use compatibility detection, and noise mitigation. It means that it is not enough for airport just to measure noise and model noise distribution. The Airport also needs GIS functionalities for noise spatial analysis and management. Split Airport Noise Information System (ANIS) will serve the airport to fulfil tasks from the government, and in the meantime, it will also help for the airport’s management and future development planning.

#### 3.5.1. Noise Monitoring System (NMS)

The NMS of Split Airport includes three Noise Measuring Terminals (NMT), located in City Trogir and Kastela. The criteria for selecting the locations are that they should be important places and noise sensitive, such as school, hospital and old church. These NMTs continuously measure noise level at the selected locations.

The system uses meteorological data to correct noise measurement (temperature, humidity, wind and pressure). Based on the physical characteristics of recorded noise, the NMS can separate noise events generated by aircrafts from other recorded noise, such as church bell ringing, car engine etc. To get noise events caused by aircraft, the NMS system uses flight data to perform a correlation between noise events recorded by a NMT and expected aircraft position above a NMT according to the time of the noise event and flight schedule.

#### 3.5.2. Integrated Noise Modelling (INM)

This software is made by the USA Federal Aviation Agency (FAA), which can model noise propagation based on aircraft position, types of engines, procedures of taking off or on and terrain elevation data etc. The INM is used to model how noise is spread in airport surrounding areas.
3.5.3. Problems with the current systems

- So far, NMS and INM have been working separately. Calibration of the parameters in INM has not been done by comparison of modelling results with real measurements from NMS.

- The input of terrain elevation data in INM can only be text files in the format of USGS elevation data instead of the common digital terrain model formats, such as TIN or GRID.

- Very often it is not possible to correlate noise events with aircrafts data in NMS. These is because an aircraft may change its path and cause a noise event when it was not expected according to scheduled data, or an aircraft can fly at higher than expected altitude and thus it will not cause a noise event as expected according to scheduled data. Therefore, radar data that will show the exact aircraft path is needed to correctly correlate noise events with aircrafts.

- Currently the output of the NMS cannot be exported to other applications.

- The NMS is only for noise measuring and the INM is only for noise distribution modelling. As it is discussed in Section 3.5, airports need to work on noise disturbance report and land use compatibility detection etc. These functionalities are not included in the current systems.

3.6. Summary

Airport noise is gaining more and more public attention due to the disturbance of people’s daily life. Under the pressure of law, airport proprietors need to have an airport noise information system for their airport noise management. Noise can be measured by different noise metrics depending on different noise characteristics. Factors that generate noise and influence noise distributions are types and numbers of aircrafts operating at the airport, aircraft’s operations and their profiles, flight paths, tracks and runways, flight schedule, airport status and weather condition, and observation position. Establishing an airport noise information system should be guided by the relevant laws and regulations. The existing systems in study area are important for problem analysis and user requirements gathering.
4. Analysis and design of the prototype Airport Noise Information System

For system analysis and design Visio software is used in this research to create all kinds of UML diagrams. UML is used for object-oriented system design. It contains different types of diagrams corresponding to different design stages. A use case diagram is developed for user requirements gathering and analysis. A class diagram is developed to represent all the classes and their attributes, operations, associations, and data types within the Airport Noise Information System. It models the static aspects of this system. An interaction diagram is developed to illustrate how system messages are sent between objects and classes. It models the dynamic aspects of the system. Finally, pre-conditions and post-conditions of each system operation in the class diagram are identified to describe how the system responsibilities are assigned to different objects and classes.

As discussed in Chapter 2, the iterative and incremental model is used for system design and development in this research. Many iterations have been gone through for the analysis and design of the prototype ANIS. Each iteration contains user requirements analysis, system design, implementation, and testing four phases and it is a refinement to the previous one. It is impossible to describe all the iterations in the thesis due to space limitation. Therefore, the user requirements analysis and system design are summarized in this chapter.

4.1. Use case diagram and User requirement analysis

Any system design should be based on user requirement gathering and analysis. The prototype airport noise information system is no exception. In this research, the user requirements gathering is fulfilled through a fieldwork at Split Airport, Croatia, at the beginning of the research.

4.1.1. Use case diagram

At the analysis phase of system design, several questions are particularly crucial:
1. Who are users?
2. What exactly are the user requirements?
3. How will the system be used?

It is not always easy for users to express precisely how they will use the system. The use case diagram is a useful tool for stimulating users to talk about the system from their viewpoints. It is a construct that helps system designers to work with users to determine system usage. A collection of use cases describes what users intend to use the system for [Schmuller, 1999]. In this research, staffs of Split Airport, Croatia, are users for the proposed system. Based on several interviews with airport staffs and
practicing the existing noise monitoring and modelling systems, the use case diagram is developed (As shown in Figure 4-1, please refer to appendix B for enlarged page).

Figure 4-1 Use case diagram for Split airport noise information system
4.1.2. User requirement analysis

Based on the use case diagram, the functional user requirements are summarized and grouped into three groups of functions.

Airport Noise Monitoring, analysis, and report

• Measure noise with noise monitors at selected sensitive locations, such as old churches, schools, and hospitals etc.
• Correlate noise events with a specific aircrafts according to radar data, the airport flight schedule, and the noise monitoring results.
• Visualize noise events
• Single noise event report
• Periodically noise events report
  - Daily
  - Weekly
  - Monthly
  - Yearly

Airport Noise Modelling

• Provide system manager the possibilities to input and change any noise modelling parameters
• Calibrate noise model with noise monitoring results.
• Calculate noise distribution grid
• Calculate noise distribution contour

Airport Noise GIS

• Noise distribution change
• Noise affected area detection in different years, seasons.
  - Display noise zone 1: noise level is larger than 75 dB
  - Display noise zone 2: noise level is larger than 67 dB but less than 75 dB
  - Detect noise affected areas
  - Detect land use compatibility

Figure 4-2 Legend of use case diagram
• Predict future noise affected area
  o What changes in noise disturbance will occur because of different types of aircraft operations, different time periods, etc. (for example, daytime and night time)?
  o What if runways are expanded?
  o What if land use types are changed?
  o What if demographic change happens?
• Detect under which noise level are sensitive locations, such as hospitals, schools etc.
• Present noise disturbance information based on population census information
  o Show the number of blocks and people affected by noise
  o Report detailed census information about noise affected population
  o Visualize the amount of people within noise affected areas on the orthophoto
• Provide possibilities to search areas that comply with user requirements. For example, a user indicates a noise level of interest and gives a distance from this noise level, the system will show it and detect if the area is under noise disturbance or not.

4.2. Class diagram for system design

Object-oriented design of a prototype Airport Noise Information System is a comprehensive work; it needs several iterations to complete the design. Within each iteration, the following methods are used. The first step is domain modelling, which is conceptual design, to identify classes within this system. Secondly, attributes for these classes have to be defined; in this step software specification is taken into account but not in much detail yet. Thirdly system operations, data types of all attributes and return data types of all operations and associations between classes are defined. Detailed software implementation should also be considered in this stage.

4.2.1. Class diagram

To develop a class diagram, the first step is to identify classes in the ANIS. The class names and their corresponding meanings are defined as follows:

1. Aircraft: defines aircraft that operates on the airport.
2. Substitution: some aircraft’s types are not standard, but they might be substituted by one or more standard types.
3. RepreByOne: an aircraft type is substituted by another type.
4. RepreByMore: an aircraft type is substituted by more than one other type.
5. Engine: defines aircraft engine type, numbers, and thrust they produce.
6. Thrust: a sort of force produced by the aircraft’s engine to enable the aircraft to climb when take-off and to decelerate when landing.
8. NDPCurve: Noise power and distance curve.
9. FlapCoefficient: Flap is part of the wing on an aircraft that can be lifted in flight to change the aircraft’s upward direction. Flap coefficients are derived from manuals that contain measured data.
from the manufacturer. The flap coefficient depends on the type of operation (approach or departure), it is used for calculating noise.

10. Airport: defines general status of an airport.
11. Schedule: defines an airport flight schedule data.
12. Runway: defines runway properties
14. TrackIdentifier: track identifier associates a track to its corresponding runway, sub-tracks etc.
15. TrackSeg: tracks are divided into several segments according to the aircraft operation and runway use. Each segment is a straight line that generates different noise.
16. AircraftOpe: defines aircraft operations, which include departure, approach, touch and go, circuit flight, and over flight.
17. ProfileIdentifier: profile identifier associates an aircraft with its profile parameters.
18. ProceduralProfile: a procedural profile defines operational procedures for various phases of the flight.
19. Departure: defines aircraft’s thrust setting and detailed information at departure.
20. Approach: defines aircraft’s thrust setting and detailed information at approach.
21. Touch&Go: defines aircraft’s thrust setting and detailed information at touch and go.
22. Circuit: defines aircraft’s thrust setting and detailed information at circuit.
23. WeaCondition: defines general weather condition of an airport.
27. NoiseMonitor: Measures and records aircraft noise events.
28. Correlation: correlates noise events with aircrafts according to radar data and flight data.
34. NoiseContour: defines noise distribution contour.
35. UserDefinedArea: user defines area by given corresponding requirements.
36. AffectedArea: defines airport noise affected area.
38. FutureAffeArea: predicts future noise affected area.
39. IncreasingOpe: increases aircraft operations to predict noise change in the future.
40. LanduseData: defines land use data source.
41. ComCriteria: land use compatibility criteria.
42. ConflictArea: analyses land use compatibility to detect conflict areas.
43. SensiLocation: defines noise sensitive locations used for conflict detection.
44. ConfliReport: defines the land use conflict report.
47. OrthoPhoto: defines orthophoto for noise visualization.
48. NoiseChange: defines and stores noise change within a given time interval.
49. Render: defines render for noise visualization.
Based on the above identified classes, attributes are added to all the classes to develop one step further class diagram, operations, associations (including inheritance relationships and aggregation relationships), and data types are added to class diagram to develop the complete class diagram. Figure 4-3 gives the overview of the class diagram. It only shows class names and their associations. Figure 4-4 shows the complete class diagram (refer to appendix C for the enlarged page). Refer to the data dictionary (See Appendix A) for the detailed explanation of the class diagram.

Figure 4-3 Overview of the class diagram
Figure 4-4 Class diagram of Split airport noise information system
4.3. System behaviour modelling

System behaviour modelling is to model the dynamic aspects of the system. In this research, an interaction diagram and pre and post-conditions of system operations are developed for behaviour modelling.

4.3.1. Interaction diagram

An interaction diagram describes dynamically how system messages are sent between different objects in order to accomplish system tasks within the airport noise information system.

Figure 4-6 is an interaction diagram to describe how messages are sent between one actor and five objects to fulfil the task to detect affected areas by airport noise.

Firstly, the system manager (actor in this system) sends a message startmodelling() to an object :NoiseModel to ask it to start noise modelling; after executing this operation, :NoiseModel sends a message getDistri() to an object :NoiseDistri to ask it to calculate a noise distribution grid; after calculation noise distribution grid, :NoiseDistri sends a message calCo() to the object :NoiseContour to ask it to calculate the noise distribution contour; when the noise distribution contour is calculated, :NoiseContour sends a message CheckData() to the object :LanduseData to check whether the land use data is ready for use, if that is the case, then :NoiseContour sends another message calAffec() to the object :AffectedArea to ask it to calculate the noise affected area. The system manager can only get the noise affected area after the system operation calAffec() is executed successfully.

We can see in the interaction diagram. Below that for each system message, there is a corresponding message sending back described by a dashed line. It shows that after an object gets a system message, it executes the corresponding system operation, and returns a message to the sender of the initial message.
4.3.2. Assigning system responsibilities

The pre and postconditions descriptions are important means for assigning system responsibilities to each system class. They define clearly which system operation should take which responsibilities; which preconditions are needed to fulfil these responsibilities and which postconditions must be satisfied after the system executes this operation. In fact they define the implementation specifications precisely. The following pre and post conditions descriptions are for operations related to airport noise GIS since they are more complicated compared to other operations.

**Class: NoiseMonitor**

**Operation: meaNNoiseValue**

**Preconditions:**
An instance of NoiseMonitor is created by the system.
An instance of Radar is created by the system and linked to NoiseMonitor.
An instance of WeaCondition is created and linked to NoiseMonitor.

**Postconditions:**
The instance of NoiseMonitor is updated.
An instance of NoiseReport is created
Class: SystemManager
Operation: startNoiseModel

Preconditions:
An instance of SystemManager is created and linked to NoiseReport. Operation calibrationModel should be checked if needed or not.

Postconditions:
An instance of NoiseModel is created and linked to SystemManager.

Class: NoiseModel
Operation: calNoiseDistri

Preconditions:
An instance of NoiseModel is created.
An instance of SystemClock is created and linked to NoiseModel.
An instance of SystemManager is created and linked to NoiseModel.
An instance of Radar is created and linked to NoiseModel.
An instance of Runway is created and linked to NoiseModel.
An instance of Terrain is created and linked to NoiseModel.
An instance of NoiseMetric is created and linked to NoiseModel.
An instance of AbsorpModel is created and linked to NoiseModel.
An instance of AttenuaModel is created and linked to NoiseModel.

Postconditions:
An instance of NoiseDistri is created and linked to NoiseModel.

Class: NoiseDistri
Operation: calNoiseCon

Preconditions:
An instance of NoiseDistri is created.

Postconditions:
An instance of NoiseContour is created and linked to NoiseDistri.
Class: NoiseChange
Operation: calNoiseChange

Preconditions:
An instance of NoiseChange is created and linked to NoiseContour.
An instance of Orthophoto is created and linked to NoiseChange.

Postconditions:
The instance of NoiseChange is updated.

Class: NoiseChange
Operation: visNoiseChange

Preconditions:
The operation of calNoiseChange is executed.
An instance of Render is created and linked to NoiseChange.

Postconditions:
The instance of Render is updated.

Class: UserDefinedArea
Operation: calDefinedArea

Preconditions:
An instance of UserDefinedArea is created and linked to NoiseContour.

Postconditions:
The instance of UserDefinedArea is updated.
Class: AffectedArea
Operation: calAffeArea

Preconditions:
Specify to get currently affected area or predict future affected area

- If it is for currently affected area detection, specify which kind of flight data should be used,
  - Averaged by different year
  - Averaged by different season
  - Measured by different noise metrics

- If it is for future affected area prediction, specify which kind of data will be changed in the future
  - By estimated increasing operations
  - By building new runways
  - By land use change
  - By demographic change

A NoiseContour instance is created and linked to NoiseDistri.
An AffectedArea instance is created and linked to NoiseContour.
A LanduseData instance is created and linked to AffectedArea.

Postconditions:
The instance of AffectedArea is updated.
An instance of CurrentAffeArea is created and inherited from AffectedArea.
An instance of FutureAffeArea is created and inherited from AffectedArea.
An instance of ConflictedArea is created and inherited from AffectedArea.

Class: ConflictArea
Operation: calConfliArea

Preconditions:
An instance of ConflictArea is created and linked to AffectedArea.
An instance of ComCriteria is created and linked to ConflictedArea.

Postconditions:
The instance of ConflictedArea is updated.
An instance of ConfliReport is created and linked to ConflictedArea.
Class: ConflictArea
Operation: visConfliarea

Preconditions:
The operation of calConfliArea is executed.
An instance of Render is created and linked to NoiseChange.

Postconditions:
The instance of Render is updated.

Class: ConflictArea
Operation: calSenLocation

Preconditions:
Operation of calConfliArea is executed already.

Postconditions:
An instance of SensiLocation is created and linked to ConflictedArea.

Class: PopuCensus
Operation: calAffeBlock

Preconditions:
An instance of NoiseContour is created.
An instance of PopuCensus is created and linked to NoiseContour.

Postconditions:
An instance of CensusReport is created and linked to PopuCensus.

Class: PopuCensus
Operation: calAffeNumber

Preconditions:
An instance of NoiseContour is created.
An instance of PopuCensus is created and linked to NoiseContour.

Postconditions:
If an instance of CensusReport is created already, then update it.
If an instance of CensusReport is not created yet, then create it and link it to PopuCensus.
**Class: PopuCensus**  
**Operation: reportInfo**

Preconditions:  
An instance of NoiseContour is created.  
An instance of PopuCensus is created and linked to NoiseContour.  
Operations calAffeBlock and calAffeNumber must be executed already.

Postconditions:  
The instance of CensusReport is updated.

**Class: PopuCensus**  
**Operation: visNoiseImpact**

Preconditions:  
The operation of calAffeBlock is executed.  
The operation of calAffeNumber is executed.  
An instance of Render is created and linked to NoiseChange.

Postconditions:  
The instance of Render is updated.

### 4.4. Summary

This chapter is the core part of this research. Based on the theory of software engineering, object-oriented system design, airport noise theory and corresponding laws and regulations, UML methodology is used for user requirement analysis and system design. A use case diagram and class diagram model the static aspects of the system. The use case diagram includes all the possible ways in which a user will use the system; the class diagram identifies all the classes in the Airport Noise Information System, and defines all the attributes, operations, associations and data types. Behaviour modelling and pre and post-conditions are used for modelling the system dynamic aspects. Behaviour modelling shows how the system messages are sent between different objects; post-conditions assign responsibilities to each system class and preconditions specify the requirements needed to fulfil the responsibilities. This chapter produces complete design documents for every stage and based on these documents, the implementation will be carried out and is described in next chapter.
5. System implementation and testing

This chapter discusses the issues related to the implementation of the ANIS. ESRI ArcCatalog is used for establishing the geodatabase to manage the data in ANIS. ESRI ArcObjects™ is selected as software development platform. ArcObjects provides an infrastructure for application customisation that lets developers concentrate on serving the specific needs of users [Zeiler, 2001]. Visual Basic for Applications (VBA), which is embedded within ArcGIS, is selected as programming language in this research. Based on the system design, a prototype for Split ANIS is developed. Testing issue is discussed as well.

5.1. Geodatabase

5.1.1. Geodatabase introduction

Geodatabase is one of the products of Environmental Systems Research Institute (ESRI). According to Zeiler (1999), a geodatabase is an instance of an object-relational database that has been enhanced by adding geographic data storage, referential integrity constraints, map display, feature editing and analysis functions. It is a top-level unit of geographic data and it is a collection of feature datasets, feature classes, object classes and relationship classes. A key purpose of the geodatabase is to handle complex geographic data with a uniform data model independent of the relational database underneath. Geodatabases are usually organized into broad categories of data such as land use, terrain, flight data, and noise distribution etc. Geodatabase manage seamless geographic data. Hence there is no partitioning of a geographic area into tiled units, but geodatabases use effective spatial indexing for a continuous representation of an extent. For this research, a wide variety of spatial data as well as nonspatial data are used. Geodatabase provides an easy and efficient way for data management for Split ANIS.

5.1.2. Ways to create a geodatabase

Whatever method is going to be used to create a geodatabase, the first step is always to design the geodatabase, by taking into consideration of the data to be used, projections, objects, feature classes and their subtypes, the relationships between them, validation rules, and so forth. Once the design is complete, depending on what kind of data sources we have already, there are three methods for creating a geodatabase [MacDonald, 1999].

- Creating a new geodatabase from scratch
- Migrating existing data into the geodatabase
- Building a geodatabase with CASE tools
In this research, a personal geodatabase ANIGeodatabase schema is designed, taking into account the different data uses. In figure 5-2, we can see that there are 6 feature datasets that are defined with same projection in the geodatabase. They are airport, land use, land use conflict, noise distribution, population census, and terrain. Each feature dataset contains different feature classes. Besides feature datasets in the geodatabase, there are 15 tables containing different types of flight data for airport noise distribution modelling.

Figure 5-1 Geodatabase of Split Airport Noise Information System
5.2. User interface design and implementation of Split ANIS, Croatia

According to system design, Split ANIS contains three groups of functions, named noise monitoring, noise modelling, and noise GIS. The principles of design user interfaces are that user interfaces should be clear, concise, easy to operate, and easy to navigate. Every part of the interface has a title to describe which main functionality it will supply to fulfil tasks, and each user interface has a return button so that the user can easily go back to the main interface.

Figure 5-2 is the start page of this system, it also gives the names of the M.Sc researcher and supervisors; Figure 5-3 is the main user interface, it has three buttons for the three main system functionalities, and contains very short explanation about this interface to guide user to start. The help button is also provided to help user whenever the user has problems.

5.2.1. Airport noise monitoring

Figure 5-4 is the main interface of the Split airport noise monitoring, it has seven functionalities based on accessing airport noise information database and the way that user requires noise information: list a single noise event; correlate a noise event with a specific aircraft according to radar data, noise data and flight data; report daily, weekly, monthly, and yearly noise events; show noise change in chart. These seven functions correspond to the seven buttons in Figure 5-4.
5.2.2. Airport noise modeling

Figure 5-5 to figure 5-11 are interfaces of Split airport noise modelling. It has seven factors represented by the tabs, and each factor has its own parameters that accept user input. All these parameters are derived from the class diagram; they are attributes of corresponding classes within class diagram. The seven factors are Radar, terrain, weather condition, flight data, noise metrics, atmospheric absorption, and, lateral attenuation.
DESIGN AND DEVELOPMENT OF PROTOTYPE AIRPORT NOISE INFORMATION SYSTEM

Figure 5-7 Noise Modelling—Lateral attenuation

Figure 5-8 Noise Modelling—Atmospheric absorption

Figure 5-9 Noise Modelling—Noise metric

Figure 5-10 Noise Modelling—Terrain

Figure 5-11 Noise Modelling—Weather data
5.2.3. Airport noise GIS

Figure 5-12 is the main interface of Split airport noise GIS, it provides six main functionalities: visualize noise distribution changes, detect noise affected area and land use conflict area, predict noise affected area, detect noise affect on sensitive locations, provide population census information in terms of noise disturbance, and present user defined area. Due to time limitation, implementation of Split ANIS focuses mainly on this noise GIS part of the prototype. The first button is for noise change visualization that shows the sequential years noise distribution. The other buttons are described in following part.

Figure 5-12 Airport Noise GIS

Figure 5-13 is the interface for detecting noise affected area and land use conflicts. Based on user input of year and season’s noise data and also which noise level will be used, this interface provides four buttons to fulfil four different tasks. According to the Croatian noise regulations, detecting noise zone 1 (noise level greater than 75 dB) and zone 2 (noise level greater than 67 dB but less than 75 dB) are important tasks of the airport. Therefore, the first two buttons are for detecting zone 1 and zone 2 noise level. Figure 5-14 shows the noise zone 1 in 1999, and Figure 5-15 shows the noise zone 2 in 1999.

The third button in this user interface is for noise affected area detection based on the land use data. Figure 5-16 shows the result of the detection of noise affected areas. In this case, the user selects 1999’s noise data, and wants to detect land use compatibility under noise level >85, 85-80, and 80-75. The ANIS detects all the land use types under these three noise levels, and visualizes them in different colours.

The fourth button in this user interface is for land use conflict detection; it is based on the affected areas detected in the previous step plus land use compatibility criteria. Due to the time limitation, it is not implemented yet.
Figure 5-17 is the user interface to predict future land use conflict. The user has the possibility to input data for future planning, such as increased aircraft operations at daytime and night time; expanded runways; changed land use; and changed demography. Based on user input, the system will predict how the noise conflict will be. Due to the unavailability of future data, this function is not implemented yet.
Figure 5-17 Noise prediction

Figure 5-18 Sensitive locations detection

Figure 5-18 is the user interface to detect land use conflict on sensitive locations. User input the year and the season of the noise data that they are interested in, and system will detect in which noise level the sensitive locations are. Figure 5-19 is the part of result in 2000.

Figure 5-19 Result of sensitive locations under noise level

Figure 5-20 is the user interface for population census under airport noise detection. Users can select which year’s noise data they are interested in firstly. Next by clicking the button named list the number of affected people, the system will show a table, which is produced based on the affected area detected in the previous step plus population census data, showing the number of affected blocks and people. By clicking the button named report detailed information, the system will present detailed census information for the area under noise disturbance. This information is very useful for future land use planning as well as airport development planning. By clicking the button named visualize, users get the possibilities to select using different charts (For example, bar chart, pie chart, and stacks chart), colour, and size to visualize noise impact on the land use map.

Figure 5-21 shows a bar chart to visualize affected people according to the census of 1998. The length of the bars represents total population, the red part is the population that is affected by noise, the blue part is the unaffected population.

Figure 5-22 shows pie charts to visualize the number of noise affected people according to the census of 1999. The size of the pie means total number of people in that area, green is the unaffected part and the purple part is the affected people in that area.
Figure 5-20 Noise and population

Figure 5-21 Bar chart for visualizing noise affected census in 1998

Figure 5-22 Pie chart for visualizing noise affected census in 1999
Figure 5-23 is the user interface to define areas according to user requirements. For example, to find the area from noise level 80-75 dB 200 kilometres away in year 1998. The system will present the area that satisfies the conditions while the representation colour and style can also be user defined. Figure 5-24 shows the result of the user defined area.

![User defined area](image1)

![Result of user defined area](image2)

**Figure 5-23 User defined area**  **Figure 5-24 Result of user defined area**

### 5.3. System testing

In this research, system testing consists of two parts: developer’s acceptance testing and user’s acceptance testing. The aim is to find answers for the following questions from both testing:

1. Is the help file easy to find?
2. Is the user interface easy to understand?
3. Is the system easy to navigate?
4. Are system functions carried out correctly?
5. Does the system give information for guiding user to fulfil tasks?
6. Is the system easy to learn?
7. Are user errors prevented?
8. Are all the use cases included in the system?

#### 5.3.1. Developer’s acceptance testing

For developer’s acceptance testing, system operation questions are designed to guide the Split ANIS testing in terms of system correctness, reliability, efficiency, usability, and user friendly. The Split ANIS operation testing contains three parts: noise monitoring, noise modelling, and noise GIS testing.
Noise monitoring testing
1. List daily noise events.
2. List monthly noise events.
3. Show yearly noise report in chart.

Noise modelling testing
1. Input weather data for noise modelling: Temperature: 23; Humidity: 59; Pressure: 75; Wind speed: 1 wind direction: 41.
2. Set noise metrics as DNL for noise modelling.

Noise GIS testing
1. Show noise distribution change and describe the trend of the change.
2. Detect noise zone 1 for summer in 1998.
3. Detect noise zone 2 for winter in 1999.
4. For summer in year 1999, detect noise affected area under noise level 85 and 75-70.
5. Input new land use data to predict future land use conflict.
7. List the number of noise affected people in 1999, and visualize the result with bar chart.
8. As a user, you want to find an area which is for summer in year 2000, this area is from noise level 70-67 dB 100 km away, and please draw this area with solid green colour.

As we discussed in chapter 2, testing is one of system development phase within each development iterations. Therefore, the developer’s acceptance testing is carried out by several iterations, and each iteration is the refinement to the previous one. The last testing result shows that all the operation tasks can be fulfilled correctly, but more work still need to be done on user friendly interfaces and system efficiency.

5.3.2. User’s acceptance testing

In this research, it is planned to use think aloud method for user’s acceptance testing. The testing tasks can be similar to the above. Elzakker (1999) described the method as: “This method consists of asking subjects to execute a particular problem-solving task and to speak out their thoughts while solving that problem and analysing the resulting verbal protocols. The thinking aloud is recorded by means of audio and/or video techniques, which result in the verbal protocols.”

The reason planning to use this method is mainly because that it lets test persons to speak out loudly what they are feeling about the system while they are doing the testing tasks. They do not need to remember and describe their thoughts afterwards because of using audio and video techniques. It is easy to watch what was happening during the testing by retrospect. Therefore, thinking aloud method can help system developers to find system functional problems as well as user interfaces problems precisely.

Due to the software unavailability in ITC test lab as well as the time limits, the user’s acceptance of the system is not carried out.
5.4. summary

The prototype Split ANIS is developed to implement the system design in chapter 4. A Geodatabase is established to manage a wide variety of data in this research. The system contains three parts of functionalities: airport noise monitoring, airport noise modelling, and airport noise GIS. For the first two functions, the implementation is mainly carried out on user interfaces and user input design. The most difficult part for airport noise modelling is to define all the input parameters, the rest of the work is just a matter of formulas and calculations. The third part, airport noise GIS, is implemented to use the advantages of the GIS spatial analysis functionalities in ArcGIS. Developer’s acceptance testing has been guided by series evaluation questions through fulfilling testing tasks. The result shows that the prototype ANIS provides sufficient functionalities for airport noise management and the user interfaces are easy to operate.
6. Conclusion and recommendations

6.1. Review research objectives and research questions

According to the research objectives, this research follows software engineering principles and adopts object-oriented technology to design and develop a prototype airport noise information system for Split Airport, Croatia. Research questions and the corresponding answers are summarized as follows:

1. What are the user requirements?

According to relevant noise laws and regulations, airport proprietors are obliged to take procedures to mitigate airport noise and to detect land use compatibilities. Therefore airports need an airport noise information system (ANIS). In this research, the case study area is Split Airport, Croatia, and Split Airport staff are users of the proposed system. Several interviews with users have been carried out and the existing systems have been tried during fieldwork. User requirements for ANIS are summarized as follows:

- To integrate all kinds of related data to model airport noise distribution and to use real measuring data to calibrate noise model.
- To analyse airport noise versus flight track, runway, different aircraft operations, and using different noise metrics.
- To detect land use compatibility with airport surroundings.

2. Which functions should be included in this system?

A use case diagram is developed based on the user requirements, and detailed user requirements are developed through communicating with users again using use case diagram. The functionalities included in ANIS contain three parts: airport noise monitoring, airport noise modelling, and airport noise GIS. The detailed system functionalities are described in Chapter four, Section 4.1.2.

3. Which software engineering methodology should be used to design the prototype system?

In chapter two, after comparing different software system development models, iterative and incremental model is selected as software engineering methodology for this research. Especially using the advantages of object-oriented technology, the system aims to promote software reusability and extensibility.
4. How to combine different data for land use conflict detection and airport noise impact analysis?

To answer this question, a lot of works need to be done in advance. Firstly, noise distribution modelling needs to be carried out. To do so, a wide variety data are needed. Airport status and weather condition data, aircraft types, engine numbers, different operations, procedural profiles and thrust and flap settings, flight tracks, runways, flight schedule, different noise metrics, and observation positions are used for noise distribution modelling. Secondly, airport noise needs to be measured by noise monitors, and these measurements are also used to calibrate the noise model. Thirdly, through GIS spatial analysis, different land use types under different noise levels are detected. Fourthly, with the land use compatibility criteria, land use conflicts are detected. Lastly, some noise sensitive locations and noise impact on population are detected and analysed.

5. How to visualize airport noise and noise impact?

The question is answered from three aspects:

- Noise distribution changes are visualized through a primitive animation of noise distribution over different time periods.

- Noise affected areas and conflict areas visualization:
  To provide possibilities to control which year and which season’s data users are interested in, and which noise level they would like to use. Each area that is under the selected noise level(s) is detected and is visualized dynamically (the affected areas one after another while doing the detection). The conflict areas should be visualized in two colours simply to show land use is compatible or not under the land use compatibility criteria (due to the time limits, this function is not implemented yet).

- Conflicted population census visualization:
  Based on the result of the number of census blocks and people in the noise conflict area, use charts to visualize these results on the land use map (orthophoto). The user has possibilities to select which chart they like to use, and options to visualize it in different ways.

6. How to test the system?

This question is answered in Chapter 5 Section 5.3.

6.2. Conclusions

- Software engineering concepts and object-oriented technology are suitable for designing and developing the ANIS that aims at reusability and extensibility. In this research, an iterative and incremental software development model is used, and much iteration has been done to design and develop the ANIS. The ANIS is based on classes and objects; it takes the advantages of inheritance, encapsulation, and polymorphism to ensure software reuse.
• UML is a powerful tool for ANIS design; it makes it easy to communicate with users as well as with people involved in different phases. It helps setting well-defined tasks and clear outcomes for each system design and development stage. In this research, use case, class, and interaction diagrams are used to model the ANIS.

• User requirements analysis is the crucial part of the ANIS design. System functionalities, system design and implementation as well as testing are all based on it.

• The GIS provides analysis and visualization possibilities in the ANIS. For land use compatibility detection, much GIS functionality is used to help the ANIS better serve for airport noise management.

• Visualization is a tool to help user to better understand noise characteristics and its impact. In the ANIS, noise distribution changes as well as noise impact on land use and population are visualized through different methods.

• The research questions are all answered and the research objective is reached.

6.3. Recommendations

• The main reason for selecting Visual Basic for Application (VBA) as implementation language is because VBA is embedded inside ArcGIS, and ArcGIS is very suitable for spatial analysis and map presentation. Another reason is the personal knowledge of the software. It is however recommended to also try the possibilities of implementation of the ANIS with other software.

• Due to the ArcGIS is a very big software, it runs slower compared to other software. Therefore, visualization of noise distribution change in the implementation does not give satisfied result. It is recommended to try other possibilities.

• Because the Split airport noise monitoring and modelling modules are not the main parts of implementation in this research, only small parts of these two functionalities are implemented. They need further implementation in the future.

• Due to the software unavailability in ITC test lab and time limits, the user’s acceptance for the prototype ANIS is not carried out. It is recommended to do so as well as doing testing by real users in order to improve it.
References

http://www.aee.faa.gov/Noise/


http://comm.db.erau.edu/esser/cont3.html


http://physics.nku.edu/asep/a150.3

http://aee.hq.faa.gov/Noise/

http://www2.awl.com/cseng/titles/0-201-89542-0/techniques/index.htm#Contents


Appendix A: Data dictionary

It is surprisingly common that a term, often technical or particular to the domain, will be used in different ways. This needs to be resolved to reduce problems in communication and ambiguous requirements [Larman, 2002]. And also due to the space limitation within class diagram, many abbreviations are used. It needs to be explained for understanding the class diagram for system modelling. The following data dictionary lists of all the classes, attributes, and operations that have been identified in the design class diagram. For each terminology, it lists the name, category, data type, description, and range of values.

Table A-1 Data dictionary

<table>
<thead>
<tr>
<th>Terminology</th>
<th>Category</th>
<th>Data type</th>
<th>Description</th>
<th>Range of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aircraft</td>
<td>Class</td>
<td></td>
<td>Define aircraft that operates on the airport</td>
<td></td>
</tr>
<tr>
<td>AircraftID</td>
<td>Attribute</td>
<td>String</td>
<td>Aircraft identifier, it is used to associate an aircraft with its flight data and noise data.</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Attribute</td>
<td>String</td>
<td>Aircraft type &amp; its engine type</td>
<td></td>
</tr>
<tr>
<td>Owner</td>
<td>Attribute</td>
<td>String</td>
<td>The owner of aircraft</td>
<td>Commercial, general aviation, military</td>
</tr>
<tr>
<td>LandingWeight</td>
<td>Attribute</td>
<td>Float</td>
<td>Aircraft weight at landing</td>
<td></td>
</tr>
<tr>
<td>TakeoffWeight</td>
<td>Attribute</td>
<td>Float</td>
<td>Aircraft weight at takeoff</td>
<td></td>
</tr>
<tr>
<td>MaxLandingDis</td>
<td>Attribute</td>
<td>Float</td>
<td>Maximum aircraft landing distance, it is measured from the edge of runway to the point of aircraft touchdown.</td>
<td></td>
</tr>
<tr>
<td>landing</td>
<td>Operation</td>
<td>Boolean</td>
<td>Check aircraft is landing or not</td>
<td></td>
</tr>
<tr>
<td>takeoff</td>
<td>Operation</td>
<td>Boolean</td>
<td>Check aircraft is taking off or not</td>
<td></td>
</tr>
<tr>
<td>flyOver</td>
<td>Operation</td>
<td>Boolean</td>
<td>Check aircraft is flying over or not</td>
<td></td>
</tr>
<tr>
<td>touchAndGo</td>
<td>Operation</td>
<td>Boolean</td>
<td>Check aircraft is touching &amp; going or not</td>
<td></td>
</tr>
<tr>
<td>Substitution</td>
<td>Class</td>
<td></td>
<td>Some aircraft’s types do not belong to those of standard, but they might be substituted by one or more standard types.</td>
<td></td>
</tr>
<tr>
<td>SubstitID</td>
<td>Attribute</td>
<td>String</td>
<td>Substitution identifier</td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Attribute</td>
<td>String</td>
<td>Substitution aircraft type</td>
<td></td>
</tr>
<tr>
<td>RepreByOne</td>
<td>Class</td>
<td></td>
<td>Aircraft is substituted by one type</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Attribute</td>
<td>String</td>
<td>Aircraft type name used to substitute the existing aircraft</td>
<td></td>
</tr>
<tr>
<td>RepreBymore</td>
<td>Class</td>
<td></td>
<td>Aircraft is substituted by more than one types</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Attribute</td>
<td>String</td>
<td>Aircraft type name used to substitute the existing aircraft</td>
<td></td>
</tr>
<tr>
<td>Percentage</td>
<td>Attribute</td>
<td>Float</td>
<td>How much proportion</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-----------</td>
<td>-------</td>
<td>---------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Engine</strong></td>
<td>Class</td>
<td></td>
<td>Define aircraft engine type, numbers, and thrust they produce</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Attribute</td>
<td>String</td>
<td>Engine types</td>
<td></td>
</tr>
<tr>
<td>Number</td>
<td>Attribute</td>
<td>Integer</td>
<td>Number of engines used in aircraft</td>
<td></td>
</tr>
<tr>
<td>getEngineThrust</td>
<td>Operation</td>
<td>Thrust</td>
<td>Calculate engine-out thrust values for certain kinds of procedure steps.</td>
<td></td>
</tr>
<tr>
<td><strong>Thrust</strong></td>
<td>Class</td>
<td></td>
<td>Define aircraft thrust used to calculate noise</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Attribute</td>
<td>String</td>
<td>Thrust type</td>
<td></td>
</tr>
<tr>
<td>CoefficientType</td>
<td>Attribute</td>
<td>String</td>
<td>Thrust coefficient type is used to calculate the corrected net thrust per engine, which is the parameter used to calculate noise</td>
<td></td>
</tr>
<tr>
<td><strong>NoiseIdentifier</strong></td>
<td>Class</td>
<td></td>
<td>Define general noise information</td>
<td></td>
</tr>
<tr>
<td>NoiseID</td>
<td>Attribute</td>
<td>String</td>
<td>Associate aircraft noise with relevant noise parameters</td>
<td></td>
</tr>
<tr>
<td>ThrustSetting-Type</td>
<td>Attribute</td>
<td>String</td>
<td>The type of thrust setting used to identify the noise curves</td>
<td></td>
</tr>
<tr>
<td>NoiseModelType</td>
<td>Attribute</td>
<td>String</td>
<td>The type of noise model to apply</td>
<td></td>
</tr>
<tr>
<td>ThrustSetting</td>
<td>Attribute</td>
<td>String</td>
<td>Thrust setting value</td>
<td></td>
</tr>
<tr>
<td><strong>NPDCurve</strong></td>
<td>Class</td>
<td></td>
<td>Noise power &amp; distance curve</td>
<td></td>
</tr>
<tr>
<td>NoiseType</td>
<td>Attribute</td>
<td>NoiseMetric</td>
<td>Showing which noise metric is used</td>
<td></td>
</tr>
<tr>
<td>OperationMode</td>
<td>Attribute</td>
<td>AircraftOpe</td>
<td>Showing which aircraft operation is used</td>
<td></td>
</tr>
<tr>
<td>ThrustSetting</td>
<td>Attribute</td>
<td>String</td>
<td>Showing whether pound or percent is used</td>
<td></td>
</tr>
<tr>
<td>Height</td>
<td>Attribute</td>
<td>Float</td>
<td>The 10 fixed distances are chosen so that they are approximately evenly spaced on a logarithmic scale.</td>
<td></td>
</tr>
<tr>
<td>NoiseLevel</td>
<td>Attribute</td>
<td>Float</td>
<td>Noise level corresponds to each of the height defined above (Height)</td>
<td></td>
</tr>
</tbody>
</table>

|   |   |   |   |

200ft, 400ft, 630ft, 1000ft, 2000ft, 4000ft, 6300ft, 10000ft, 16000ft, 25000ft
<table>
<thead>
<tr>
<th><strong>FlapCoefficient</strong></th>
<th><strong>Class</strong></th>
<th><strong>Flap coefficients are derived from manuals that contain measured data from the manufacturer. Flap coefficients depend on the type of operation (approach or departure) and the flaps and gear configuration of the aircraft.</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>FlapID</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>Operation</td>
<td>Attribute</td>
<td>AircraftOpe</td>
</tr>
<tr>
<td>getRCoefficient</td>
<td>Operation</td>
<td>Double</td>
</tr>
<tr>
<td>getC/DCoefficient</td>
<td>Operation</td>
<td>Double</td>
</tr>
<tr>
<td>getBCoefficient</td>
<td>Operation</td>
<td>Double</td>
</tr>
<tr>
<td><strong>Airport</strong></td>
<td><strong>Class</strong></td>
<td><strong>Define general condition of an airport</strong></td>
</tr>
<tr>
<td>AirportID</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>Name</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>Elevation</td>
<td>Attribute</td>
<td>Float</td>
</tr>
<tr>
<td>XOrigin</td>
<td>Attribute</td>
<td>Double</td>
</tr>
<tr>
<td>YOrigin</td>
<td>Attribute</td>
<td>Double</td>
</tr>
<tr>
<td><strong>Schedule</strong></td>
<td><strong>Class</strong></td>
<td><strong>Define an airport flight schedule data</strong></td>
</tr>
<tr>
<td>Aircraft</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>Date</td>
<td>Attribute</td>
<td>Date</td>
</tr>
<tr>
<td>DayTime</td>
<td>Attribute</td>
<td>Time</td>
</tr>
<tr>
<td>NightTime</td>
<td>Attribute</td>
<td>Time</td>
</tr>
<tr>
<td>FromAirport</td>
<td>Attribute</td>
<td>Airport</td>
</tr>
<tr>
<td>ToAirport</td>
<td>Attribute</td>
<td>Airport</td>
</tr>
<tr>
<td>Operation</td>
<td>Attribute</td>
<td>AircraftOpe</td>
</tr>
<tr>
<td><strong>Runway</strong></td>
<td><strong>Class</strong></td>
<td><strong>Define runway properties</strong></td>
</tr>
<tr>
<td>RunwayID</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>Attribute/Class</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RunwayEnd1</td>
<td>String</td>
<td>The identifier of first end of the runway used by aircraft</td>
</tr>
<tr>
<td>RunwayEnd2</td>
<td>String</td>
<td>The identifier of second end of the runway used by aircraft</td>
</tr>
<tr>
<td>Length</td>
<td>Float</td>
<td>The length of the runway</td>
</tr>
<tr>
<td>Gradient</td>
<td>Float</td>
<td>The gradient of the runway</td>
</tr>
<tr>
<td>RunwayEnds</td>
<td>Class</td>
<td>Define runway ends conditions in detail</td>
</tr>
<tr>
<td>XCoordinate</td>
<td>Double</td>
<td>X Coordinate of runway end</td>
</tr>
<tr>
<td>YCoordinate</td>
<td>Double</td>
<td>Y Coordinate of runway end</td>
</tr>
<tr>
<td>ElevationMSL</td>
<td>Float</td>
<td>Runway end elevation above mean sea level</td>
</tr>
<tr>
<td>DisplacedThresA</td>
<td>Float</td>
<td>Displaced threshold of approach is measured from the physical end of runway to the threshold crossing point.</td>
</tr>
<tr>
<td>DisplacedThresD</td>
<td>Float</td>
<td>Displaced threshold of departure is measured from the physical end of runway to the nominal start roll point.</td>
</tr>
<tr>
<td>GlideSlope</td>
<td>Float</td>
<td>Glide slope in degree is an approach angle of aircraft.</td>
</tr>
<tr>
<td>ThreshCorHei</td>
<td>Float</td>
<td>Thresh crossing Height in meter is the height above the runway at the runway approach threshold.</td>
</tr>
<tr>
<td>TrackIdentifier</td>
<td>Class</td>
<td>Track identifier associates a track to its corresponding runway, sub-tracks etc.</td>
</tr>
<tr>
<td>TrackName</td>
<td>String</td>
<td>The full track name includes the runway end, operation type, and specific name for this track.</td>
</tr>
<tr>
<td>RunwayID</td>
<td>String</td>
<td>Runway is used at this track</td>
</tr>
<tr>
<td>SubTrackNum</td>
<td>Integer</td>
<td>The number of sub-tracks. Sub-tracks are tracks along side of a backbone track. The backbone track and its sub-tracks are collectively called a “dispersed” track.</td>
</tr>
<tr>
<td>PercentOpera</td>
<td>Float</td>
<td>The percentage that aircraft use backbone track and sub-tracks.</td>
</tr>
<tr>
<td>DeltaDistance</td>
<td>Float</td>
<td>Delta distance is the distance adjustment made to the start-roll or touchdown point on the runway.</td>
</tr>
<tr>
<td>TrackSeg</td>
<td>Class</td>
<td>Tracks are divided into several segments according to the aircraft operations.</td>
</tr>
<tr>
<td>SegNum</td>
<td>Integer</td>
<td>The order of the segment</td>
</tr>
<tr>
<td>SegType</td>
<td>String</td>
<td>The type of the segment, straight, left-turn, or right-turn.</td>
</tr>
<tr>
<td>Distance</td>
<td>Float</td>
<td>The distance of the segment</td>
</tr>
<tr>
<td>AircraftOpe</td>
<td>Class</td>
<td>Define possible operations of aircraft</td>
</tr>
<tr>
<td>Approach</td>
<td>Boolean</td>
<td>An aircraft’s operation is approaching</td>
</tr>
<tr>
<td>Depart</td>
<td>Boolean</td>
<td>An aircraft’s operation is departing</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>CircuitFlight</td>
<td>Boolean</td>
<td>An aircraft’s operation is circuit flying</td>
</tr>
<tr>
<td>TouchandGo</td>
<td>Boolean</td>
<td>An aircraft’s operation is touching and going</td>
</tr>
<tr>
<td>FlyOver</td>
<td>Boolean</td>
<td>An aircraft’s operation is flying over</td>
</tr>
<tr>
<td>ProfileIdentifier</td>
<td>Class</td>
<td>Profile identifier associates an aircraft with its profile parameters.</td>
</tr>
<tr>
<td>AircraftID</td>
<td>String</td>
<td>Aircraft identifier</td>
</tr>
<tr>
<td>Operation</td>
<td>AircraftOperation</td>
<td>Aircraft’s operation types</td>
</tr>
<tr>
<td>ProfileGroup</td>
<td>String</td>
<td>The profile group name identifies profiles that belong to a group of similar procedures but of different weights or some other profile parameter.</td>
</tr>
<tr>
<td>ProfileStage</td>
<td>Integer</td>
<td>Profile stage distinguishes members in a profile group. It is used to identify stage lengths for departure profiles. The longer the trip, the heavier the average takeoff weight due to increased fuel requirements.</td>
</tr>
<tr>
<td>ProfileWeight</td>
<td>Float</td>
<td>Profile weight is the weight of aircraft during operation.</td>
</tr>
<tr>
<td>Procedural Profile</td>
<td>Class</td>
<td>Procedural profile defines operational procedures for various phases of the flight.</td>
</tr>
<tr>
<td>AircraftID</td>
<td>String</td>
<td>Aircraft identifier</td>
</tr>
<tr>
<td>ProfileName</td>
<td>String</td>
<td>The name of profile</td>
</tr>
<tr>
<td>StepNumber</td>
<td>Integer</td>
<td>The order of the defined steps</td>
</tr>
<tr>
<td>StepType</td>
<td>String</td>
<td>There are nine types of procedure steps.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Takeoff, Climb, Accelerate, Level, Level stretch, Cruise Climb, Descend, Land, Decelerate</td>
</tr>
<tr>
<td>FlapID</td>
<td>String</td>
<td>Flaps identifier</td>
</tr>
<tr>
<td>Departure</td>
<td>Class</td>
<td>Defines aircraft’s thrust setting and detailed information at departure</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Step types used: Takeoff Climb Accelerate Climb</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ThrustType</td>
<td>String</td>
<td>Thrust types under this procedure</td>
</tr>
<tr>
<td>ClimbRate</td>
<td>Float</td>
<td>Aircraft’s climb rate during departure</td>
</tr>
<tr>
<td>FinalAltitude</td>
<td>Float</td>
<td>The final altitude of departure</td>
</tr>
<tr>
<td>FinalSpeed</td>
<td>Float</td>
<td>The final speed of departure</td>
</tr>
<tr>
<td>Approach</td>
<td>Class</td>
<td>Defines aircraft’s thrust setting and detailed information at approach</td>
</tr>
<tr>
<td>StartAltitude</td>
<td>Float</td>
<td>The aircraft start altitude of approach</td>
</tr>
<tr>
<td>StartSpeed</td>
<td>Float</td>
<td>The aircraft start speed of approach</td>
</tr>
<tr>
<td>DescentAngle</td>
<td>Float</td>
<td>The aircraft descent angle of approach</td>
</tr>
<tr>
<td>TouchdownRoll</td>
<td>Float</td>
<td>The aircraft touchdown roll of approach</td>
</tr>
<tr>
<td>Touch&amp;Go</td>
<td>Class</td>
<td>Defines aircraft’s thrust setting and detailed information at touch and go operation</td>
</tr>
<tr>
<td>Altitude</td>
<td>Float</td>
<td>The aircraft altitude under level step</td>
</tr>
<tr>
<td>Speed</td>
<td>Float</td>
<td>The aircraft speed under level step</td>
</tr>
<tr>
<td>Distance</td>
<td>Float</td>
<td>The aircraft flight distance under the level step</td>
</tr>
<tr>
<td>StartAltitude</td>
<td>Float</td>
<td>The aircraft start altitude under the descent step</td>
</tr>
<tr>
<td>StartSpeed</td>
<td>Float</td>
<td>The aircraft start speed under the descent step</td>
</tr>
<tr>
<td>DescentAngle</td>
<td>Float</td>
<td>The aircraft descent angle under the descent step</td>
</tr>
<tr>
<td>TouchdownRoll</td>
<td>Float</td>
<td>The aircraft touchdown roll under the land step</td>
</tr>
<tr>
<td>ThrustType</td>
<td>String</td>
<td>The aircraft thrust type under the takeoff, climb, and accelerate step type</td>
</tr>
<tr>
<td>ClimbRate</td>
<td>Float</td>
<td>The aircraft climb rate under the climb step type</td>
</tr>
<tr>
<td>FinalAltitude</td>
<td>Float</td>
<td>The final altitude of climb step type</td>
</tr>
<tr>
<td>FinalSpeed</td>
<td>Float</td>
<td>The final speed of accelerate step type</td>
</tr>
<tr>
<td>Circuit</td>
<td>Class</td>
<td>Defines aircraft’s thrust setting and detailed information at circuit</td>
</tr>
</tbody>
</table>

Step types used:
- Aircraft procedures: Accelerate, Climb, FinalAltitude, FinalSpeed
- Approach: Descent, Land, Decelerate
- Touch&Go: Level, Descent, Land, Takeoff, Climb, Accelerate, Level
- Circuit: Takeoff
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ThrustType</td>
<td>Attribute</td>
<td>The thrust type under takeoff, climb, or accelerate step type</td>
</tr>
<tr>
<td>FinalAltitude</td>
<td>Attribute</td>
<td>The final altitude under the climb step</td>
</tr>
<tr>
<td>ClimbRate</td>
<td>Attribute</td>
<td>The climb rate under the accelerate step</td>
</tr>
<tr>
<td>FinalSpeed</td>
<td>Attribute</td>
<td>The final speed under the accelerate step</td>
</tr>
<tr>
<td>Altitude</td>
<td>Attribute</td>
<td>The altitude under the level step</td>
</tr>
<tr>
<td>Speed</td>
<td>Attribute</td>
<td>The speed under the level step</td>
</tr>
<tr>
<td>Distance</td>
<td>Attribute</td>
<td>The aircraft flight distance under the level step</td>
</tr>
<tr>
<td>StartAltitude</td>
<td>Attribute</td>
<td>The start altitude under the descent step</td>
</tr>
<tr>
<td>StartSpeed</td>
<td>Attribute</td>
<td>The start speed under the descent step</td>
</tr>
<tr>
<td>DescentAngle</td>
<td>Attribute</td>
<td>The descent angle under the descent step</td>
</tr>
<tr>
<td>TouchdownRoll</td>
<td>Attribute</td>
<td>The touchdown roll under the land step</td>
</tr>
<tr>
<td>StartThrust</td>
<td>Attribute</td>
<td>The start thrust under the decelerate step</td>
</tr>
<tr>
<td>Temperature</td>
<td>Attribute</td>
<td>The average temperature of airport</td>
</tr>
<tr>
<td>Humidity</td>
<td>Attribute</td>
<td>The average humidity of airport</td>
</tr>
<tr>
<td>Pressure</td>
<td>Attribute</td>
<td>The average pressure of airport</td>
</tr>
<tr>
<td>WindSpeed</td>
<td>Attribute</td>
<td>The average wind speed of airport</td>
</tr>
<tr>
<td>WindDirection</td>
<td>Attribute</td>
<td>The average wind direction of airport</td>
</tr>
<tr>
<td>SpectralClass</td>
<td>Attribute</td>
<td>Define different noise classes according to the type of aircrafts and the number of engines</td>
</tr>
<tr>
<td>WeaCondition</td>
<td>Class</td>
<td>Define airport weather condition</td>
</tr>
<tr>
<td>WeaCondition</td>
<td>Attribute</td>
<td>Float</td>
</tr>
<tr>
<td>getWeaCon</td>
<td>Operation</td>
<td>Assess weather data for modelling</td>
</tr>
<tr>
<td>calAbsorp</td>
<td>Operation</td>
<td>Calculate the atmospheric absorption value</td>
</tr>
<tr>
<td>AttenuaModel</td>
<td>Class</td>
<td>Lateral attenuation model</td>
</tr>
<tr>
<td>Model</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>Status</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>calAttenValue</td>
<td>Operation</td>
<td>Float</td>
</tr>
<tr>
<td>Radar</td>
<td>Class</td>
<td>Radar record aircraft flight path</td>
</tr>
<tr>
<td>FlightID</td>
<td>Attribute</td>
<td>String</td>
</tr>
<tr>
<td>Date</td>
<td>Attribute</td>
<td>Date</td>
</tr>
<tr>
<td>Attribute</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>Time</td>
<td>Time</td>
<td>The time of flight</td>
</tr>
<tr>
<td>X Coordinate</td>
<td>Double</td>
<td>X Coordinate of aircraft at the time point</td>
</tr>
<tr>
<td>Y Coordinate</td>
<td>Double</td>
<td>Y Coordinate of aircraft at the time point</td>
</tr>
<tr>
<td>Aircraft Altitude</td>
<td>Float</td>
<td>Aircraft altitude at the time point</td>
</tr>
<tr>
<td>Aircraft Speed</td>
<td>Float</td>
<td>Aircraft speed at the time point</td>
</tr>
<tr>
<td>Noise Monitor</td>
<td></td>
<td>Measure and record aircraft noise events</td>
</tr>
<tr>
<td>Monitor ID</td>
<td>String</td>
<td>Noise monitor identifier</td>
</tr>
<tr>
<td>Name</td>
<td>String</td>
<td>Noise monitor name</td>
</tr>
<tr>
<td>X Coordinate</td>
<td>Double</td>
<td>X Coordinate of noise monitor</td>
</tr>
<tr>
<td>Y Coordinate</td>
<td>Double</td>
<td>Y Coordinate of noise monitor</td>
</tr>
<tr>
<td>Date</td>
<td>Date</td>
<td>The date of measurement</td>
</tr>
<tr>
<td>Time</td>
<td>Time</td>
<td>The time of measurement</td>
</tr>
<tr>
<td>Noise Value</td>
<td>Float</td>
<td>The measured noise value</td>
</tr>
<tr>
<td>get Radar Data</td>
<td>Radar</td>
<td>Access radar data</td>
</tr>
<tr>
<td>Measure Noise</td>
<td>Float</td>
<td>Measure noise value</td>
</tr>
<tr>
<td>Correlation</td>
<td></td>
<td>Correlate noise events with aircraft according to radar data and flight data.</td>
</tr>
<tr>
<td>get Flight Data</td>
<td>Schedule</td>
<td>Access flight data</td>
</tr>
<tr>
<td>get Radar</td>
<td>Radar</td>
<td>Access radar data</td>
</tr>
<tr>
<td>correlate</td>
<td>File</td>
<td>Execute correlation</td>
</tr>
<tr>
<td>Noise Report</td>
<td></td>
<td>Report noise information</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>The type of the report</td>
</tr>
<tr>
<td>single Event</td>
<td>File</td>
<td>Report single noise event information</td>
</tr>
<tr>
<td>daily Report</td>
<td>File</td>
<td>Report daily noise information</td>
</tr>
<tr>
<td>weekly Report</td>
<td>File</td>
<td>Report weekly noise information</td>
</tr>
<tr>
<td>monthly Report</td>
<td>File</td>
<td>Report monthly noise information</td>
</tr>
<tr>
<td>yearly Report</td>
<td>File</td>
<td>Report yearly noise information</td>
</tr>
<tr>
<td>Noise Metric</td>
<td></td>
<td>Define noise metrics type used for noise modelling</td>
</tr>
<tr>
<td>Type</td>
<td>String</td>
<td>There are 16 different noise metrics that belong to 3 noise categories. Among them, Lmax, Leq, SEL, and DNL are most popular noise metrics for airport noise measurement.</td>
</tr>
<tr>
<td>Terrain</td>
<td></td>
<td>Define terrain for noise modelling</td>
</tr>
<tr>
<td>Data Type</td>
<td>String</td>
<td>The data type of the terrain data</td>
</tr>
<tr>
<td>Name Database</td>
<td>String</td>
<td>The name of the personal geodatabase of the data</td>
</tr>
</tbody>
</table>

Lmax, Leq, SEL, DNL
<table>
<thead>
<tr>
<th>Name</th>
<th>Attribute</th>
<th>String</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameSet</td>
<td>Attribute</td>
<td>String</td>
<td>The name of the feature dataset of the data</td>
</tr>
<tr>
<td>NameFeaCla</td>
<td>Attribute</td>
<td>String</td>
<td>The name of the feature class of the data</td>
</tr>
<tr>
<td>FeaType</td>
<td>Attribute</td>
<td>String</td>
<td>The feature type of the data</td>
</tr>
<tr>
<td>GeoType</td>
<td>Attribute</td>
<td>String</td>
<td>The geometry type of the data</td>
</tr>
<tr>
<td>calTin</td>
<td>Operation</td>
<td>File</td>
<td>Calculate TIN from the elevation contour</td>
</tr>
<tr>
<td>SystemManager</td>
<td>Class</td>
<td></td>
<td>Airport Noise information system manager takes charge of noise model calibration and running the system.</td>
</tr>
<tr>
<td>Name</td>
<td>Attribute</td>
<td>String</td>
<td>Name of the manager</td>
</tr>
<tr>
<td>Password</td>
<td>Attribute</td>
<td>String</td>
<td>Password of the manager for enter system</td>
</tr>
<tr>
<td>Date</td>
<td>Attribute</td>
<td>Date</td>
<td>The date of calibration</td>
</tr>
<tr>
<td>Time</td>
<td>Attribute</td>
<td>Time</td>
<td>The time of calibration</td>
</tr>
<tr>
<td>LastCalibration</td>
<td>Attribute</td>
<td>Date</td>
<td>The date of last calibration</td>
</tr>
<tr>
<td>calibrateModel</td>
<td>Operation</td>
<td>None</td>
<td>Calibrate noise distribution model using real noise measurements.</td>
</tr>
<tr>
<td>startNoiseModel</td>
<td>Operation</td>
<td>None</td>
<td>Execute noise model</td>
</tr>
<tr>
<td>SystemClock</td>
<td>Class</td>
<td></td>
<td>System clock record system running time and if calibration has not executed by system manager for one month, it will start calibration automatically.</td>
</tr>
<tr>
<td>calibrateModel</td>
<td>Operation</td>
<td>None</td>
<td>Calibrate noise distribution model using real noise measurements.</td>
</tr>
<tr>
<td>NoiseModel</td>
<td>Class</td>
<td></td>
<td>Define noise model for modelling noise distribution</td>
</tr>
<tr>
<td>Date</td>
<td>Attribute</td>
<td>Date</td>
<td>The date of modelling</td>
</tr>
<tr>
<td>Time</td>
<td>Attribute</td>
<td>Time</td>
<td>The time of modelling</td>
</tr>
<tr>
<td>Radar</td>
<td>Attribute</td>
<td>Radar</td>
<td>Radar data used for modelling</td>
</tr>
<tr>
<td>Runway</td>
<td>Attribute</td>
<td>Runway</td>
<td>Runway data used for modelling</td>
</tr>
<tr>
<td>Metric</td>
<td>Attribute</td>
<td>NoiseMetric</td>
<td>Noise metrics used for modelling</td>
</tr>
<tr>
<td>Weather</td>
<td>Attribute</td>
<td>Weather-Condition</td>
<td>Weather data used for modelling</td>
</tr>
<tr>
<td>getAbsorValue</td>
<td>Operation</td>
<td>Absor-Model</td>
<td>Access atmospheric absorption value</td>
</tr>
<tr>
<td>getAttenuValue</td>
<td>Operation</td>
<td>At-tenuaModel</td>
<td>Access lateral attenuation value</td>
</tr>
<tr>
<td>getTerrain</td>
<td>Operation</td>
<td>Terrain</td>
<td>Access terrain data for modelling</td>
</tr>
<tr>
<td>getRadar</td>
<td>Operation</td>
<td>Radar</td>
<td>Access radar data for modelling</td>
</tr>
<tr>
<td>getFlightData</td>
<td>Operation</td>
<td>File</td>
<td>Access flight data for modelling</td>
</tr>
<tr>
<td>calNoiseDistri</td>
<td>Operation</td>
<td>Raster</td>
<td>Calculate noise distribution according to all the relevant parameters.</td>
</tr>
<tr>
<td>NoiseDistri</td>
<td>Class</td>
<td></td>
<td>Calculate and generate noise distribution grid.</td>
</tr>
<tr>
<td>Raster</td>
<td>Attribute</td>
<td>Raster</td>
<td>Define noise distribution grid, including grid number, grid coordinates, grid size,</td>
</tr>
<tr>
<td>Attribute/Class</td>
<td>Type</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>---------</td>
<td>--------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>LowestValue</td>
<td>Integer</td>
<td>Lowest noise value within the grid</td>
<td></td>
</tr>
<tr>
<td>HighestValue</td>
<td>Integer</td>
<td>Highest noise value within the grid</td>
<td></td>
</tr>
<tr>
<td>calNoiseCon</td>
<td>Operation</td>
<td>Calculate noise contours</td>
<td></td>
</tr>
<tr>
<td>NoiseContour</td>
<td>Class</td>
<td>Record noise contour calculated by noise distribution.</td>
<td></td>
</tr>
<tr>
<td>Value</td>
<td>Integer</td>
<td>Noise value of the contour</td>
<td></td>
</tr>
<tr>
<td>Interval</td>
<td>Integer</td>
<td>Noise value interval between two adjacent contours.</td>
<td></td>
</tr>
<tr>
<td>AreaBetween</td>
<td>Float</td>
<td>Area between two adjacent contours</td>
<td></td>
</tr>
<tr>
<td>UserDefinedArea</td>
<td>Class</td>
<td>User defined area by given corresponding requirements</td>
<td></td>
</tr>
<tr>
<td>NoiseLevel</td>
<td>Integer</td>
<td>User defined a specific noise level</td>
<td></td>
</tr>
<tr>
<td>Direction</td>
<td>String</td>
<td>User defined direction from a noise contour</td>
<td></td>
</tr>
<tr>
<td>Distance</td>
<td>Float</td>
<td>User defined the distance from a noise contour</td>
<td></td>
</tr>
<tr>
<td>calDefinedArea</td>
<td>Operation</td>
<td>Calculate the user defined area</td>
<td></td>
</tr>
<tr>
<td>AffectedArea</td>
<td>Class</td>
<td>Define airport noise affected area</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td>Date</td>
<td>The date of detection</td>
<td></td>
</tr>
<tr>
<td>Time</td>
<td>Time</td>
<td>The time of detection</td>
<td></td>
</tr>
<tr>
<td>NoiseLevel</td>
<td>Integer</td>
<td>The noise level used for detection</td>
<td></td>
</tr>
<tr>
<td>Landuse</td>
<td>String</td>
<td>The detected land use type</td>
<td></td>
</tr>
<tr>
<td>calAffeArea</td>
<td>Operation</td>
<td>File Detect noise affected area</td>
<td></td>
</tr>
<tr>
<td>CurrentAffeArea</td>
<td>Class</td>
<td>Define current noise affected area</td>
<td></td>
</tr>
<tr>
<td>Year</td>
<td>String</td>
<td>The year of data used</td>
<td></td>
</tr>
<tr>
<td>Season</td>
<td>String</td>
<td>In which season of data used</td>
<td></td>
</tr>
<tr>
<td>Metrics</td>
<td>NoiseMetrics</td>
<td>Which noise metric is used for calculation</td>
<td></td>
</tr>
<tr>
<td>FutureAffeArea</td>
<td>Class</td>
<td>Predict future noise affected area</td>
<td></td>
</tr>
<tr>
<td>NewRunway</td>
<td>Runway</td>
<td>Expand runway</td>
<td></td>
</tr>
<tr>
<td>LandChange</td>
<td>Landuse-Data</td>
<td>Land use change</td>
<td></td>
</tr>
<tr>
<td>NewDemo</td>
<td>PopuCensus</td>
<td>Demographic change</td>
<td></td>
</tr>
<tr>
<td>IncreasingOpe</td>
<td>Class</td>
<td>Increase aircraft operations</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>AircraftOpe</td>
<td>Operation type</td>
<td></td>
</tr>
<tr>
<td>NumatDaytime</td>
<td>Integer</td>
<td>Number of operations at day time</td>
<td></td>
</tr>
<tr>
<td>NumatNight</td>
<td>Integer</td>
<td>Number of operations at night time</td>
<td></td>
</tr>
<tr>
<td>LanduseData</td>
<td>Class</td>
<td>Define land use data source</td>
<td></td>
</tr>
<tr>
<td>DataType</td>
<td>String</td>
<td>The data type of the land use data</td>
<td></td>
</tr>
<tr>
<td>NameDaBase</td>
<td>String</td>
<td>The name of the personal geodatabase of the data</td>
<td></td>
</tr>
<tr>
<td><strong>NameFeaSet</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>The name of the feature dataset of the data</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------</td>
<td>------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td><strong>NameFeaCla</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>The name of the feature class of the data</td>
</tr>
<tr>
<td><strong>FeatType</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>The feature type of the data</td>
</tr>
<tr>
<td><strong>GeoType</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>The geometry type of the data</td>
</tr>
<tr>
<td><strong>ComCriteria</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Land use compatibility criteria</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>Name of the compatibility criteria</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>Geodatabase type</td>
</tr>
<tr>
<td><strong>ConflictArea</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Analyse land use compatibility to detect conflict areas</td>
</tr>
<tr>
<td><strong>Criteria</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>ComCriteria</strong></td>
<td>Land use compatibility criteria</td>
</tr>
<tr>
<td></td>
<td><strong>calConfliArea</strong></td>
<td><strong>Operation</strong></td>
<td>ConfliReport</td>
</tr>
<tr>
<td></td>
<td><strong>visConfliArea</strong></td>
<td><strong>Operation</strong></td>
<td>Layer</td>
</tr>
<tr>
<td></td>
<td><strong>calSenLocation</strong></td>
<td><strong>Operation</strong></td>
<td>SensiLocation</td>
</tr>
<tr>
<td><strong>SensiLocation</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Define noise sensitive locations used for conflict detection</td>
</tr>
<tr>
<td><strong>PointID</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>The ID of the location</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>The Type of the location</td>
</tr>
<tr>
<td><strong>Latitude</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Float</strong></td>
<td>The latitude of the location</td>
</tr>
<tr>
<td><strong>Longitude</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Float</strong></td>
<td>The longitude of the location</td>
</tr>
<tr>
<td><strong>NoiseValue</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Float</strong></td>
<td>Detect the sensitive location is under which noise level</td>
</tr>
<tr>
<td><strong>ConfliRes</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Boolean</strong></td>
<td>Show the result of detection</td>
</tr>
<tr>
<td><strong>ConfliReport</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Define the land use conflict report</td>
</tr>
<tr>
<td><strong>Name</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>Name of the report</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>Geodatabase type</td>
</tr>
<tr>
<td><strong>PopuCensus</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Define population census information</td>
</tr>
<tr>
<td><strong>CountyCode</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>String</strong></td>
<td>The code of the county to which Census block belongs</td>
</tr>
<tr>
<td><strong>Block</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Integer</strong></td>
<td>Census block code</td>
</tr>
<tr>
<td><strong>LandArea</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Float</strong></td>
<td>Area of the Census block</td>
</tr>
<tr>
<td><strong>Latitude</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Float</strong></td>
<td>Latitude of the centroid of the Census block</td>
</tr>
<tr>
<td><strong>Longitude</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Float</strong></td>
<td>Longitude of the centroid of the Census block</td>
</tr>
<tr>
<td><strong>PopCount</strong></td>
<td><strong>Attribute</strong></td>
<td><strong>Integer</strong></td>
<td>Number of people living in the Census block</td>
</tr>
<tr>
<td></td>
<td><strong>calAffeBlock</strong></td>
<td><strong>Operation</strong></td>
<td>String</td>
</tr>
<tr>
<td></td>
<td><strong>calAffeNumber</strong></td>
<td><strong>Operation</strong></td>
<td>Integer</td>
</tr>
<tr>
<td></td>
<td><strong>reportInfo</strong></td>
<td><strong>Operation</strong></td>
<td>CensusReport</td>
</tr>
<tr>
<td><strong>CensusReport</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Define the Census report</td>
</tr>
<tr>
<td>AffeBlock</td>
<td>Attribute</td>
<td>Integer</td>
<td>The affected block code</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>---------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>AffeNumber</td>
<td>Attribute</td>
<td>Integer</td>
<td>The number of affected people in the Census block</td>
</tr>
<tr>
<td>DetailedInfo</td>
<td>Attribute</td>
<td>File</td>
<td>The detailed information about this Census block</td>
</tr>
<tr>
<td><strong>OrthoPhoto</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Define orthophoto for noise visualization</td>
</tr>
<tr>
<td>DataType</td>
<td>Attribute</td>
<td>String</td>
<td>The data type of the orthophoto, the initial value is Raster.</td>
</tr>
<tr>
<td>Source</td>
<td>Attribute</td>
<td>String</td>
<td>The source folder of the data</td>
</tr>
<tr>
<td>Name</td>
<td>Attribute</td>
<td>String</td>
<td>The name of the data file</td>
</tr>
<tr>
<td><strong>NoiseChange</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Defines and store noise change within given time interval.</td>
</tr>
<tr>
<td>FromDate</td>
<td>Attribute</td>
<td>Date</td>
<td>The beginning date for calculating noise change</td>
</tr>
<tr>
<td>ToDate</td>
<td>Attribute</td>
<td>Date</td>
<td>The end date for calculating noise change</td>
</tr>
<tr>
<td>Contour</td>
<td>Attribute</td>
<td>NoiseContour</td>
<td>Which noise contours are involved</td>
</tr>
<tr>
<td>calNoiseChange</td>
<td>Operation</td>
<td>Layer</td>
<td>Calculate noise change between different time period</td>
</tr>
<tr>
<td>visNoiseChange</td>
<td>Operation</td>
<td>Layer</td>
<td>Visualize noise change</td>
</tr>
<tr>
<td><strong>Render</strong></td>
<td><strong>Class</strong></td>
<td></td>
<td>Defines render for visualization</td>
</tr>
<tr>
<td>NoiseAnimation</td>
<td>Attribute</td>
<td>Layer</td>
<td>Defines render for visualization of noise animation</td>
</tr>
<tr>
<td>ConflicAreas</td>
<td>Attribute</td>
<td>Layer</td>
<td>Defines render for visualization of noise conflict areas</td>
</tr>
<tr>
<td>NoiseImpact</td>
<td>Attribute</td>
<td>Layer</td>
<td>Defines render for visualization of noise impact</td>
</tr>
</tbody>
</table>
Appendix B: Use case diagram

This page should be substituted by A3 print!
Appendix C: Class diagram

This page should be substituted by A3 print!
Appendix B

The use case diagram of Split ANIS

System Manager

Measure and Report Noise

- Calibrate Noise Model
  - Noise Grid
  - Noise Contour

- Correlate Noise event with aircraft

- Get Noise Distribution

- Visualize Noise
  - Noise distribution change
  - Noise conflicted areas
  - Noise impact Vs. Census

- Get user defined noise free area

- Detect affected area

- Measure and Report Noise

- Get number of affected people

- Get Population Census Information
  - Detailed information of affected people

- Single Event Report

- Report Periodic noise distribution
  - Weekly
  - Monthly
  - Yearly

- Future Affected
  - caused by increasing operations
  - Caused by increasing runways

- System Clock
  - Noise monitor should be calibrated Every month automatically by system if no manually calibration is executed by system manager

- Sensitive Locations
  - in different years, seasons
Appendix C: The class diagram of Split ANIS