

A dynamic property valuation decision support system

MJ Wentzel

 **orcid.org/0000-0002-3088-2353**

Dissertation accepted in partial fulfilment of the requirements for
the degree *Master of Science in Computer Science* at the
North-West University

Supervisor: Dr A van der Merwe

Graduation July 2021

24999369

ABSTRACT

A decision support system (DSS) is an information system that provides assistance towards solving problems that usually involve multiple courses of action to a specific problem. An expert system incorporates expert knowledge and inference procedures to solve more unconventional or complex problems in a certain field. The property valuation process involves performing certain procedures through the use of expert knowledge to determine the value of a property. The most common property valuation technique is known as the sales comparison method that involves identifying the most optimal property which closely resembles the property under consideration. Existing methods to execute the valuation process quickly and efficiently are known as automated valuation models or mass-appraisal. This study was conducted to determine if valuation mathematics used by expert valuers can be effectively integrated into a DSS that compensates for dynamic fluctuations in the economy in the South African residential property market. The aim was achieved through the development of a computerised demonstrator implementing the property valuation process through methods that were identified during the knowledge engineering process. The expert DSS was successfully applied to test scenarios of real-world residential properties. The system consists of a user interface for subject property attribute inputs, a decision-making module that incorporates the sales comparison method, a database of real-world properties and a knowledge base which uses property valuation mathematics to value the subject property. The expert DSS successfully calculates the value by selecting the most comparable property from the database according to an algorithm and compensating for attribute differences according to a mathematical model.

Keywords: Decision support systems, expert systems, knowledge engineering, mass-appraisal, property valuation, sales comparison method.

OPSOMMING

'n Besluitsteunstelsel (BOS) is 'n inligtingstelsel wat probleme oplos wat gewoonlik verskeie oplossings kan hê deur die beste oplossing te gebruik. 'n Deskundige stelsel bevat kennis vanaf deskundiges om ingewikkelde probleme op te los in sekere velde wat deskundige kennis benodig. Die eiendomswaardasie proses behels die uitvoering van sekere prosedures, met behulp van deskundige kennis, om die waarde van 'n eiendom te bepaal. Die mees algemene waardasie tegniek staan bekend as die vergelykbare verkoopsmetode wat 'n optimale eiendom identifiseer wat so vergelykbaar as moontlik is aan 'n eiendom wat waardeur word. Bestaande modelle en metodes wat gebruik word om die waardasie proses so vinnig en doeltreffend as moontlik uit te voer, staan bekend as outomatiese waarderingsmodelle of massa-waardering. Hierdie studie is uitgevoer om vas te stel of waarderingswiskunde wat deur kundige waardeerders gebruik word, effektief geïntegreer kan word in 'n BOS wat kompenseer vir dinamiese skommeling in die ekonomie in die Suid-Afrikaanse mark vir residensiële eiendomme. Die doel is bereik deur die ontwikkeling van 'n gerekenariseerde demonstrateur wat die eiendomswaardasie implementeer deur middel van metodes wat tydens die kennisingenieurswese proses identifiseer is. Die deskundige BOS is suksesvol toegepas op toetsgevalle van regte-wêreld residensiële eiendomme. Die stelsel bestaan uit 'n gebruikerskoppelvlak vir invoer van die eienskappe van die eiendom onder beskouing, 'n besluitnemingsmodule wat die vergelykbare verkoopsmetode inkorporeer, 'n databasis van werklike eiendomme en 'n kennisbasis wat die wiskunde vir waardasie van eiendomme gebruik om die eiendom onder beskouing te waardeer. Die kundige BOS bereken die waarde suksesvol deur die mees vergelykbare eiendom uit die databasis te kies volgens 'n algoritme en te kompenseer vir eienskapverskille volgens 'n wiskundige model.

Sleutelwoorde: Besluitsteunstelsels, deskundige stelsels, eiendomswaardasies, kennisingenieurswese, massa-waardasie, vergelykbare verkoopsmetode.

ACKNOWLEDGEMENTS

First off, I want to start by thanking my family for all of their love and support during this study. They assisted me to stay calm and focused on the harder parts of the study.

I would like to thank the Lord for keeping us all safe during these troubling times and for allowing me to complete this study.

I would also like to thank Wendy Barrow and the editors and authors of CTrans who reviewed the language within my dissertation. I hope it was as readable as possible!

And most of all, I would like to thank my supervisor Dr Annette van der Merwe. This study and its completion would not have been possible without her endless dedication, patience and support. I have a tendency to rush towards conclusion and there were several instances where I struggled to move forward. Without her continuous support, I would not have been able to make it this far!

11 March 2021

EDITING CERTIFICATE**Mr Thys Wentzel,**

This certificate serves to confirm that I am a qualified and certified editor and translator. I confirm that the mini-dissertation with the title *A dynamic property valuation decision support system* has undergone a professional language edit (including the checking of spelling, grammar, register and punctuation and if the reference list meets the requirements of APA 7th edition). The onus rests on the client to work through the proposed changes after the edit and accept or reject these changes.

Yours faithfully

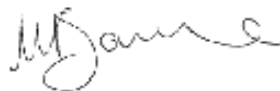
**Wendy Barrow**

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION AND PROBLEM STATEMENT	1
1.1 Introduction	1
1.2 Problem statement	1
1.3 Research goals.....	2
1.4 Research methodology.....	3
1.5 Chapter outline	4
1.6 Chapter summary.....	5
 CHAPTER 2 PROPERTY VALUATION AND COMPUTER SYSTEMS	 6
2.1 Introduction	6
2.2 Property valuation.....	6
2.3 The sales comparison method	10
2.4 Property valuation automation.....	12
2.4.1 Automated valuation models.....	12
2.4.2 Mass-appraisal	13
2.4.3 Variable dependency using regression	16
2.5 DSSs in property valuation.....	17
2.5.1 Data-driven DSS	19
2.5.2 Knowledge-driven DSS.....	20
2.6 Chapter summary.....	21
 CHAPTER 3 AN EXPERT DSS IN PROPERTY VALUATION.....	 22
3.1 Introduction	22

3.2	Knowledge and data-driven DSSs	22
3.3	Components of an expert DSS in property valuation	23
3.3.1	The user interface	23
3.3.2	A property database.....	24
3.3.3	The knowledge base	26
3.3.4	The decision-making module	27
3.4	Chapter summary.....	27
CHAPTER 4 EXPERT DSS DEVELOPMENT		28
4.1	Introduction	28
4.2	Expert systems.....	28
4.3	Expert system development lifecycle	31
4.3.1	Planning.....	32
4.3.1.1	Task phasing	33
4.3.1.2	Feasibility assessment	33
4.3.1.3	Preliminary functional layout	35
4.3.1.4	High-level requirements identification.....	35
4.3.2	Knowledge definition	36
4.3.2.1	Knowledge source identification and selection	36
4.3.2.2	Knowledge acquisition	37
4.3.2.3	Interviews and observations	37
4.3.2.4	Document analysis.....	38
4.4	Chapter summary.....	39

CHAPTER 5 EXPERT DSS STRUCTURE	40
5.1 Introduction	40
5.2 Knowledge analysis	40
5.2.1 Identifying property attributes.....	40
5.2.2 The sales comparison method and mathematical model	41
5.3 Knowledge extraction	45
5.3.1 Functional layout.....	45
5.4 Chapter summary.....	46
 CHAPTER 6: DEVELOPMENT AND IMPLEMENTATION OF THE EXPERT DSS	 47
6.1 Introduction	47
6.2 Knowledge design.....	47
6.2.1 Knowledge classification.....	47
6.2.2 Detailed design: code-and-checkout	51
6.3 Calculation of market value.....	53
6.3.1 Structural attribute adjustments	54
6.3.2 Umbrella attribute adjustments.....	55
6.4 Case-based expert DSS demonstration.....	58
6.5 Chapter summary.....	61
 CHAPTER 7: SYSTEM EVALUATION.....	 62
7.1 Introduction	62
7.2 Knowledge verification	62
7.3 System evaluation.....	67
7.4 Recommendations	68

7.5	Research contributions	69
7.6	Chapter summary	69
CHAPTER 8: CONCLUSION.....		71
8.1	Introduction	71
8.2	Objectives of the study	71
8.3	Problems experienced	74
8.4	Further research.....	75
8.5	Conclusion.....	76
BIBLIOGRAPHY		77
APPENDIX A: PROPERTY VALUATION INPUT DOCUMENT FOR THE EXPERT VALUERS.....		83
APPENDIX B: PROPERTY VALUATION INPUT PAGE FOR THE MUNICIPAL VALUERS ROLL.....		84
APPENDIX C: RESULTS OF HISTORIC VALUATION TESTS.....		85
APPENDIX D: EXPERT VALUER QUESTIONNAIRE FEEDBACK		87
APPENDIX E: EXPERT DSS PROPERTY DATABASE.....		91
APPENDIX F: PROPERTY VALUATION DSS DEMONSTRATOR C# CODE		99

LIST OF TABLES

Table 2.1: Additional studies in the sales comparison method	12
Table 2.2: Recent studies in automated valuation models.....	13
Table 2.3: Recent studies in mass appraisal	16
Table 2.4: Studies in property valuation DSSs	19
Table 4.1: Task phases of the expert system development lifecycle	33
Table 6.1: Property attribute priority rankings.....	49
Table 6.2 The selection of comparable properties.....	52
Table 6.3: Database table representing the structural attribute adjustment values	54
Table 6.4: Example structural adjustments.....	55
Table 6.5: Average property values for different suburbs in Klerksdorp.....	56
Table 6.6: Change in the monthly CPI for South Africa (January to July)	57
Table 6.7: Possible adjustments for property condition	57
Table 6.8: Example umbrella adjustment table.....	58
Table 6.9: Sample property for demonstration purposes	59

LIST OF FIGURES

Figure 3.1: Expert property valuation DSS structure	23
Figure 4.1: Human behaviour when executing an action according to certain events and rules	29
Figure 4.2: The general structure of an expert system	30
Figure 4.3: The expert system development lifecycle as a linear model	32
Figure 5.1: Database structure for the expert DSS.....	42
Figure 6.1: Flow diagram demonstrating step sequence and functions of the expert DSS	50
Figure 6.2: Subject property input screen.....	51
Figure 6.3: An example list of comparable properties.....	52
Figure 6.4: Required attribute adjustments for the most comparable property	53
Figure 6.5: Properties comparable to the 22 Sample Street	59
Figure 6.6: Calculated market value of 22 Sample Street.....	60
Figure 7.1: Expert valuer results compared to the expert DSS's valuation results	65
Figure 7.2: Expert DSS valuations vs. those from the municipal roll.....	66
Figure 7.3: Expert DSS valuation vs. most recent sale value of an atypical property.....	66

CHAPTER 1 INTRODUCTION AND PROBLEM STATEMENT

1.1 Introduction

Property valuation provides an objective and non-biased appraisal of a property without unnecessary additional adjustments to the value of a property that certain players in the process might add for their own personal gain. Therefore, many organisations consider it an important part of property development. However, due to the length of the process and a few external factors, such as economy fluctuations, valuer personal bias and the limited experience of a new valuer, certain errors may arise during the valuation process that can have a major influence on its outcome (Amidu *et al.*, 2019). This is especially true in the residential property environment (Babtan, 2020), highlighting the need to optimise and automate the residential valuation process.

Many software solutions aim to automate property valuation, but few focus on optimising residential property valuation. Most software systems focus on developing projects and few compensate for market imbalances. Another problem is that most of these systems are centred around a specific environment that may be inapplicable when conducted in a different area.

Studies have also shown that many valuers tend to steer clear of software solutions due to the lack of compensation of certain external factors (Van der Vyfer, 2020). Moreover, many valuers, especially new and inexperienced ones, can benefit from existing solutions or additional decision support provided by an automated valuation system (Amidu *et al.*, 2019). Therefore, a decision support system (DSS) incorporating expert knowledge is proposed to solve the automated valuation problem of residential properties to provide valuation results. This in turn can assist expert valuers by providing a much needed second opinion on the outcome of their own valuations.

The purpose of this chapter is to provide a basic layout of this study by discussing the problem statement in Section 1.2, the research goals in Section 1.3 and the methodology that was used to conduct the research in Section 1.4. This is followed by an outline and short description of each chapter in Section 1.5.

1.2 Problem statement

The concept of property valuation describes the process of determining the value of a property by considering specific property attributes (French & Gabrielli, 2018). This process is performed by valuers who have expert knowledge regarding a predetermined set of rules towards determining the value of a property (Shapiro *et al.*, 2019). These rules incorporate factors such as property location, the condition and a few other key property factors (IAAO, 2003). A valuator

typically follows a structured approach when completing the valuation process by considering various factors and property attributes (Amidu & Boyd, 2018; Angelos *et al.*, 2013; Metzner & Kindt, 2018) and applying an appropriate valuation model (Gorbenko, 2019; Michaletz & Artemenkov, 2018). A report summarising the property value estimate is then compiled and submitted to the relevant client. The importance of the factors that are considered during valuation can vary due to, among others, fluctuations in the economy, the environment or the political status of a country (Shapiro *et al.*, 2019) thereby creating a demand for a real-time automated solution.

Computer systems are generally developed to assist in providing decision support when mathematical calculations are required in real-world domains where variables are subject to contextual change. A DSS is described as a computer system that is mostly used by businesses or organisations to assist with judgement, determination and courses of action (Power, 2002). Such a system can search a large amount of data to compile a comprehensive solution towards a problem (Segal, 2019).

An expert valuer incorporates existing knowledge and mathematical techniques to perform the valuation process. Various mathematical modelling approaches are used in the property valuation environment for specific purposes like calculating amortisation tables, calculating appreciation and depreciation rates and the internal rate of return (Ellerman, 2004). Different forms of regression analysis models can also be used to determine the value of a property, such as multiple linear regression, hedonic-regression and *k*-nearest neighbour regression (Yildirim, 2019). These techniques, however, do not effectively compensate for factors that can influence the value of a property dynamically over time, such as fluctuations in the economy (Boyle, 1984). The focus of this study is to determine if valuation mathematics used by expert valuers can be effectively integrated into a DSS that compensate for dynamic fluctuations in the economy in the South African residential property market.

1.3 Research goals

The aim of this study is to incorporate expert system technology and mathematical modelling approaches into a property valuation decision support system that can dynamically adapt to the evolving economic climate of the South African environment. The following secondary objectives need to be met in order to achieve this aim:

- *Conduct* an in-depth literature study on property valuation, DSSs, current automated valuation or mass-appraisal techniques and valuation techniques that are applicable towards the development of a DSS.

- *Perform* the required stages of knowledge engineering in order to present a theoretical layout of an expert DSS for property valuation in South Africa, using expert system principals.
- *Investigate* the use of mathematical modelling in property valuation by considering different models and *formulating* a relevant model for the South African property environment.
- *Develop* and *design* an interactive computerised demonstrator in the form of an expert DSS that utilises mathematical modelling techniques appropriate for the South African property environment.
- *Verify* and *validate* the implementation of the system using historical data sets.
- *Evaluate* the system in terms of the criteria identified in literature and experts that were consulted during the knowledge engineering process.
- *Reflect* on the possible contributions to the South African property environment and *recommend* sensible improvements towards wide-scale system implementation.

1.4 Research methodology

Research is conducted according to a collection of certain research principles, methods and thought patterns which is known as a research paradigm (Rehman & Alharthi, 2016). These methods allow researchers to evaluate and justify their research and hypotheses in the community. Some common research paradigms that were considered for this study are elaborated on next.

The interpretivistic paradigm focuses on the search for different solutions for a problem, while remaining objective (Saunders *et al.*, 2015). Interpretivism is described as the assumption that multiple realities exist and are perceived differently by individuals (Oates, 2006). Researchers in this paradigm should always remain objective by reflecting on the reasoning behind why they want to solve a specific problem or why they are conducting a specific study. Personal opinions and beliefs of a researcher must not influence the outcome of the study. Interpretivism focuses primarily on creating a theory, rather than testing whether a hypothesis is true.

Unlike interpretivism that is dependent on varying interpretations of a scenario, the positivistic paradigm revolves around the concept that everything in the natural world functions in a certain manner and is governed by certain rules (Randolph, 2009). This paradigm is not influenced by different opinions in the social context and can be proven through an active or experimental approach. This leads to the following two assumptions (Oates, 2006):

- The world is ordered and not considered to be random.

- The world can be investigated with an objective approach, meaning that certain facts can be proven to be true if the evidence is valid.

In the positivistic paradigm, research is generally conducted according to the design science approach in which observations lead to instilling a change and evaluating its effect by means of quantitative methods in a deductive approach (Rehman & Alharthi, 2016). These methods include experiments which involve testing the cause and effect in an environment.

In the context of this study, a literature study and interviews with experts served to obtain fundamental knowledge on the South African property valuation environment. An artefact in the form of an expert DSS was then developed and evaluated in an experiment on case studies and archives.

1.5 Chapter outline

This section briefly provides a layout and brief summary of this study's chapters.

Chapter 2 provides comprehensive insight into various property valuation approaches, as well as related research and contexts. Varying methodologies for automating property valuation are also elaborated on. The chapter concludes with insight into the use of DSSs in property valuation.

In Chapter 3 the software tools required for developing the expert DSS are identified and discussed. This is followed by a layout of the components planned for the system and a short description of each.

Development of the expert DSS was done according to the expert system development lifecycle discussed in Chapter 4. System planning and data collection as part of the knowledge definition phase are discussed.

Chapter 5 continues on the development by elaborating on the knowledge analysis and extraction phases. The data collected in the knowledge definition phase is analysed and interpreted, leading to the functional layout of the system being finalised.

The developed expert DSS is discussed in detail in Chapter 6. The main functions are described and accompanied by screenshots and examples.

Chapter 7 purposes to discuss system testing and evaluation. The chapter concludes with some recommendations for improving the system, as well as valuable contributions to the relevant fields of study.

Finally, Chapter 8 concludes the study by providing a brief summary of the outcomes, describing how the objectives have been met, and presenting some recommendations and considerations for future research.

1.6 Chapter summary

The purpose of Chapter 1 was to introduce the context and plan layout of the study. Section 1.2 stated the research question by stating the goal and the objectives towards achieving it. Section 1.4 identified the research paradigm used in the study. Then the chapter was concluded with a brief summary of the chapters to follow.

CHAPTER 2 PROPERTY VALUATION AND COMPUTER SYSTEMS

2.1 Introduction

The purpose of this chapter as a literature study is to identify how the property valuation process is executed, how the process can be improved by computer software techniques and automation, as well as how different researchers approached problems relating to the valuation process, creating the theoretical background of this study. The proposed software technique for the current study is a decision support system (DSS), therefore it is important to understand how a DSS functions and to determine how a DSS can be used by examining how other researchers have used similar systems in the property valuation environment.

This chapter discusses the focus area, or knowledge domain, of property valuation for the incorporation of a DSS in Sections 2.2 and 2.3. Predominant automated valuation techniques are discussed to establish an idea of how property valuation may be improved upon through automation and how a DSS can be incorporated in this context in Section 2.4. Finally, the basics of DSS techniques are discussed to provide a general understanding on how such a system can be used in Section 2.5.

2.2 Property valuation

The concept of property valuation can be described as the process of determining the value of a property by considering specific property attributes and environmental factors (Epley, 2017). This process is performed by valuers who have expert knowledge regarding a predetermined set of rules compiled by the government of a specific country towards determining the value of a property (Amidu & Boyd, 2018). These rules incorporate factors, such as property location, the condition of its buildings (if applicable) and equipment situated on the property (IAAO, 2003). Such rules are subject to change in accordance with transformation in property valuation laws as set by government officials. Valuations are required for various purposes, for example:

- to obtain the sales price of a property;
- the administration of a deceased estate;
- objections against the municipal valuation roll;
- for insurance purposes – to determine the replacement value of a property to ensure that the property is not under-insured;
- any legal dispute that may arise from a sales agreement, servitude dispute or problems arising from co-ownership;
- high court application for liquidation or sequestration of a person or company's estate;
- financial records of a company;

- expropriation; and
- restitution of land rights in terms of the constitution.

To complete the valuation process, a valuer has to follow a structured approach by considering various factors and the property's attributes (Amidu & Boyd, 2018; Angelos *et al.*, 2013; Metzner & Kindt, 2018) and apply an appropriate valuation model (Gorbenko, 2019; Michaletz & Artemenkov, 2018).

A valuation report is a summary of the valuer's findings during the valuation process (Lott, 2020). These findings include any information that can influence the value of a property, such as: whether the building is traditional to that country or not; the overall condition of the property; whether the building has been rebuilt; and how it compares to other buildings in the area. The actual contents of the report should contain at least the following information (Anon, 2018):

- the name of the instructor;
- the name of the property owner;
- the property street address as well as deeds description;
- date of the valuation;
- the purpose of the valuation;
- type of the property (like residential, business or agricultural);
- the property location;
- the overall structure of the property;
- the local neighbourhood and amenities (such as hospitals, services and supplies or entertainment);
- details of the property (buildings, fencing, equipment, or size in square meters to name a few, etcetera);
- number of building floors, rooms, bathrooms;
- the condition of the property;
- improvements that the property might require;
- renovation that has been done in the past few years;
- the age and an estimated future lifespan of the property;
- the overall maintenance level that the property requires (how much effort is required from the property owners);
- availability of a front yard, a back yard, and a garage;
- the calculated fair market value of the property; and
- pictures of the property, preferably in colour.

Important information in the report is highlighted, such as the estimated value and any potential defects. For example, a property originally estimated to be worth R600 000 can be valued for

R550 000 due to structural damage to the walls of the building. The importance of the factors that are considered during valuation can vary due to fluctuations in the economy, the environment, or the political status of a country (Amidu & Boyd, 2018). As such, rules and regulations have been implemented in different countries to ensure that the valuations remain valid and accurate.

This study was conducted in the South African property environment and is, therefore, subject to a unique set of laws. Outlines of some laws currently in place in South Africa to ensure the correctness of property valuation (Ellenberger *et al.*, 2010; Wentzel, 2020), include the following:

- *The Property Valuation Act 17 of 2014*: This first act regulates the method of valuation of fixed property that is destined for expropriation by the state or for the restitution of land rights. This act, in accordance with section 25(3) of the constitution, determines that the current use of the property as well as any historical benefits granted by the state must also be taken into consideration when determining the value of the property.
- *The Municipal Property Rates Act 6 of 2004*: The regulation of the valuation of properties for the purpose of determining rates and taxes payable to the local municipalities is addressed. It also states the method that the valuer must use to determine value, what factors he must take into consideration and what aspects must not be included and also regulates the objection process against the municipal valuation roll.
- *The Property Valuers Profession Act 47 of 2000*: The qualification of valuers, the work of the South African Council for the Property Valuers Profession (SACPVP) as well as the conduct of valuers are controlled in this act. The purpose is therefore to guide and protect the ethics of the profession.

Apart from these statutes, there are various other acts of law dealing with valuations in a lesser extent. These include:

- the *Subdivision of Agricultural Land Act 70 of 1970*;
- municipal ordinances relating to the zoning of land, taxation and rendering of services to the property;
- *National Roads Act 54 of 1971* in respect of subdivision and use of certain parcels of land.
- *Property Practitioners Act 22 of 2019* that regulates estate agents;
- *Council for the Built Environment Act 42 of 2000* creating a council that regulates the building sector;
- *Restitution of Land Rights Act 22 of 1994* to be read in conjunction with both the Constitution and the *Property Valuation Act 2014*; and
- *Sectional Title Schemes Management Act 8 of 2011* relating to flats.

Apart from adhering to the laws relating to property valuation, a valuer must have the proper qualification and must be duly registered at the SACPVP. The valuer's task differs from that of an estate agent, as the valuer must supply an *objective* value of the property, while an estate agent will act in the interest of a client and the valuations may be subjective (Hendrikse, 2015).

In terms of the International Valuation Standards Council (IVSC), as adopted by the SACPVP, there are three acceptable methods of valuation of property (Ellenberger *et al.*, 2010):

- The *sales comparison method* whereby the sale prices of similar or substitute properties are compared with assumptions regarding facts, conditions or situations affecting the subject of, or approach to the valuation which may not be capable or worthy of verification. The valuer must then consider whether the properties are the same, and if not, make appropriate adjustments. The value of the most comparable property (property with the least number of differences in attributes) is used and the adjustments that must be made. The adjustment value of attributes is added to the value of a property when that specific attribute is not the same as that of the subject property. This method is most often utilised for residential property valuations.
- The *income capitalisation method* provides an indication of value by converting future cash flow to a single currency value. The value of the asset is determined by a reference to the value of income, cash flow, or cost savings generated by the asset. The idea of the income capitalisation method is that the value of an income generating property cannot be the same value as that of any other property. This method is deemed to be the most appropriate and accurate method of valuation for an investment property that can produce an income.
- The *cost method* provides an indication of value using the economic principal that a buyer would not pay more for an asset than the amount it will cost to obtain an asset of equal quality. This method is used to determine what a property or asset would be worth if it was built from the ground up, as if it was a brand-new asset. Three cost approaches can be implemented, namely: replacement cost, reproduction cost and the summation method.

The definition of market value must also be investigated. In terms of the IVSC, the definition of market value is the estimated amount for which an asset property would exchange on the date of valuation between a willing buyer and seller in an arms-length transaction. An arms-length transaction is a transaction where both parties, the buyer and the seller, are independent and have their own interests, after proper marketing wherein both parties had each acted prudently, knowledgeable and without compulsion (Hendrikse, 2015). The prices paid for comparable properties in the neighbourhood are the contributing evidence to the market value. In the case of *Minister of Water Affairs v Mostert* (1966) it was stated that the comparable transactions afford a sound basis for arriving at a satisfactory guide in determining market value. This is also echoed

in the case of *Estate Marks v Pretoria City Council* (1969) where it was stated that the sales comparison method is a sound valuation principle.

2.3 The sales comparison method

In the sales comparison approach, a valuer typically uses recent sales of properties with the same attributes as the one under consideration to determine an estimated value (IAAO, 2003). The sales comparison process involves two steps. The first step is to find comparable sales to that of the current property being valued. This is done by identifying properties in the region that share the same attributes as the current property and have been sold recently. Before any value is determined, a valuer is despatched to the property to establish its attributes. They may include the property size, number of rooms, number of storeys, building style, garages, available space of amenities, outdoor features and the condition of the building(s) (Lott, 2020). With these in mind, a valuer then searches for properties with similar attributes from external sources, mainly from the *Deeds Office* but also from sources such as websites, catalogues, articles, or government sources (Hendrikse, 2015).

The second step is to adjust the value according to any property attributes that differ from those of the compared properties (IAAO, 2003). Such attributes can include a larger garage space, a swimming pool, larger property size, or a different building style. Generally, comparison adjustments are mostly used in the sales comparison method of residential properties to calculate the market value of a property by using the price of the most comparable property and adding the number of adjustment values that is required for differences if property attributes (such as the attributes mentioned above). To determine the total adjustment value, the valuer must compare the subject property with the most comparable property by considering the attributes that differ from each other. The valuer must then calculate the value of each attribute adjustment by referring to building costs of these attributes and calculating its worth. These attribute adjustment values are then added or subtracted from the total value of the property.

According to Tsu-Chin and Ying-Chih (2010), the sales comparison method can also be described as the nearest-neighbour method of property valuation. The nearest-neighbour method or analysis, in most fields, is described as the method of determining the distance between nodes or points on a plain and calculating the closest nodes to a point of origin or starting point (Taylor, 2016). This method is mostly employed in geography where, for example, the analyst attempts to determine the closest point of interest on a map. An analyst would start by identifying all the possible points of interest on a map and a point of origin, representing the area where, for example, the analyst presently resides. Then the analyst in this example would calculate the distance from the point of origin to each point of interest and choose the point that is the closest to the point of origin. The reasoning behind why the sales comparison method is similar to the

nearest-neighbour method, is that the valuer attempts to identify a set of closest comparable properties to a subject property, not necessarily just by its location, but by how closely the comparable property resembles the subject property through its attributes. Tsu-Chin and Ying-Chih (2010) compared the nearest-neighbour method with regression analysis to determine whether the conventional method of sales comparison, or in this case, described as the nearest-neighbour method, is superior to regression.

Regression is the method that is used to determine whether a set of independent variables has an impact on a dependent variable. If no data redundancy is identified, meaning the data follows a certain pattern or the data records share certain characteristics with one another, it can then be used to predict a new dependent variable (Swanepoel *et al.*, 2011). In the case of property valuation, regression can assist in determining the impact that each property attribute has on the value of a property. Tsu-Chin and Ying-Chih (2010) proved that both the nearest-neighbour method and the traditional sales comparison method is more accurate, because regression must deal with additional problems like multicollinearity, heteroscedasticity and autocorrelation (discussed in Section 2.4) if the dataset is difficult to predict or show no patterns or relationships (Swanepoel *et al.*, 2011).

Another study incorporating the sales comparison method with regression was conducted by Farkas and Porumb (2020). They proposed a multi-attribute valuation model to determine the values of apartments, based on their attributes and comparing them to their actual previous selling prices. One feature of this model was that a valuer could evaluate each attribute and assign to it a score of importance to be used in the model. The scores were multiplied with certain values that had been set according to a dataset of comparable properties and added or some cases subtracted from the value of the property. Farkas and Porumb (2020) concluded that their model was more efficient than simple linear regression, given that the apartments were sold in a balanced market. Market imbalance is when the values of assets in an economy fluctuate due to events in a country, such as natural disasters. However, the model used in this study did not compensate for market imbalance. This means that if it was to be used on a dataset that provides data from different time periods with great price fluctuations it can lead to inaccurate estimations. This model was primarily used on apartments. Table 2.1 shows additional, but less relevant studies, that have been conducted utilising the sales comparison approach.

Table 2.1: Additional studies in the sales comparison method

Authors	Title	Research Theme
Popescu, Cerna Mladin, Boazu and Bienert (2009)	Methodology for real estate appraisal of green value	Identified a different sales comparison method for valuing green properties by looking at additional factors, such as energy consumed or saved.
Anuar and Nor Hana Asykin (2010)	Comparison method - Preference of adjustment techniques among valuers.	Discussed different attribute adjustment techniques for the sales comparison method in Malaysia.
Ellis (2011)	Sales comparison valuation of development and operating stage mineral properties	Identified a method of sales comparison to value mineral properties, based on net operating income.
Rahim and Razali (2018)	Comparing the performance of similarity measures in sales comparison method	Compared different techniques on how comparable properties are selected for the sales comparison method.

Although the sales comparison method has been shown to be more reliable than regression in determining the value of a property, regression can still be used to determine the impact that each property attribute has on the value of a property when patterns are found. Therefore, the need to develop a system that compensates for market imbalance, and incorporates the sales comparison method to select comparable properties and regression to identify the correct attributes, is identified.

2.4 Property valuation automation

Property valuation can be automated by implementing certain models and techniques. Software applications are developed in many fields to assist in problem-solving and task automation. Property valuation is no exception. Many software applications are implemented in different areas of the property valuation environment and they use various methods to solve specific problems. Most software applications require a model to execute certain mathematical tasks automatically, therefore it is important to identify how to incorporate a valuation model in software for task automation.

2.4.1 Automated valuation models

Property valuation is conducted in a specific manner and this must be translated into a specific procedure to be implemented in software. An automated valuation model (AVM) is incorporated in a computer program and is used to estimate the market value of a property, depending on the model (IAAO, 2003). An AVM is built using the following steps:

1. Specify the model by reviewing the variables and gathered data and selecting the model that would yield the best results for the current property. The baseline models of property valuation include the sales comparison, income capitalisation, and cost models.
2. Calibrate the model by changing the mathematical coefficients in the model and comparing them to the statistics of average property valuation models or by consulting expert valuers.
3. Test the model on historic valuations to check whether the results are accurate.
4. Make adjustments to the model.
5. Recalibrate the model.
6. Repeat steps 3 to 5 until the model results are approved by the user of the system.

According to Epley (2017), it is important to reference the development process of an AVM, because it provides some guidelines on how to develop a system for property valuation purposes. The most important requirement of an AVM is that it is accepted by the user of the system, because if the AVM is not correct according to the user, the system can produce inaccurate or unacceptable results (Epley, 2017). Table 2.2 shows some additional research regarding automated valuation models.

Table 2.2: Recent studies in automated valuation models

Authors	Title	Research Theme
d'Amato and Kauko (2017)	Advances in automated valuation modelling	Outlined current studies and projects related to automated valuation models.
Birkeland and D'Silva (2018)	Developing and evaluating an automated valuation model for residential real estate in Oslo.	Implemented a machine learning technique, known as stacked generalisation as an automated valuation model for residential properties in Oslo.
Gallin <i>et al.</i> (2018)	Measuring aggregate housing wealth: New insights from an automated valuation model.	Measured the wealth of houses in the US based on Zillow's automated valuation model.
Babtan (2020)	Automated valuation model for residential real estates	Discussed the current problems in property valuation and attempted to build an algorithm for an AVM to value dynamic properties in Romania.

AVMs might provide a baseline on how to incorporate the main property valuation methods for automated valuations, but is not the most common method to value a large number of properties. The common term for appraising large quantities of properties is known as mass-appraisal.

2.4.2 Mass-appraisal

The property valuation process can be deemed unnecessarily long and expensive if the client wants to determine the values of a large number of properties. To solve this problem, a set of

techniques can be implemented on a property dataset, given that the dataset is accurate, to procedurally predict new property values, given the attributes of a list of new or subject properties. This is known as mass-appraisal (Bencure *et al.*, 2019).

Mass-appraisal is not limited to one specific technique or model, but almost always relies on a database, depending on whether it stores a set of properties or provide methods on how to calculate a value (Kilpatrick, 2011). A database of properties in mass-appraisal must contain some way of distinguishing properties from one another. Each property is distinguishable by its attributes, therefore, having a database that describes the properties and their attributes is important for value calculation, because it is highly unlikely for any one property to be exactly the same as another (Zyga, 2018).

It is important to examine how each attribute influences the value of a property (Selim, 2011). The most common method to determine the relationship between variables is known as regression (Bozic *et al.*, 2013; Okoruwa, 2019). Regression is mostly applied in cases where a dependent and an independent variable are present. Given a specific dataset, regression is used to determine the impact that the independent variable has on a dependent variable, for example to test whether the colour of a house has an impact on its value. The value of the dependent variable changes as the value of the independent variable does without influencing any other value (Swanepoel *et al.*, 2011).

In the case of property valuation there is one dependent variable that is the value of a property influenced by many independent variables in the form of property attributes that can have an influence on the market value. Such independent variables include, but are not limited to, the number of bedrooms, the number of bathrooms, garage size or a swimming pool. In general, regression is performed to find the dependency between a single set of variables. Multiple regression analysis (MRA), however, is used in cases where many independent variables influence a single dependent variable (Bozic *et al.*, 2013).

In finance, the application of MRA is known as hedonic regression (Butsic *et al.*, 2011; Selim, 2011; Yildirim, 2019). Hedonic regression involves the division of an asset or certain goods into several different attributes, each of which is valued separately and then combined to form a final price estimate (Selim, 2011). In property valuation, the hedonic regression model is used on a dataset of properties to find a relationship between actual market values of the properties and their attributes. After hedonic regression is applied in such case, a new property value can be calculated by entering the attributes of the property and applying the coefficients of the regression results in an equation. This also facilitates valuers or researchers to determine the demand for such attributes in the housing market (Yildirim, 2018). However, a few major problems have been encountered when creating market assumptions, such as:

- the identification of supply and demand;
- market disequilibrium where a surpluses or shortages of goods occur (market balance or equilibrium is achieved when supply meets demand (Ellenberger *et al.*, 2010)); and
- market segmentation is the partitioning of a business market into different groups depending on their behaviour or interests (Hendrikse, 2015).

The problems in regression are due to the heterogeneity among clients and researchers, in other words, people will have different views on which property attributes are important (Gorbenko, 2019). The following is an example of hedonic regression, where a study was conducted on the impact of climate change on ski-resorts in North America, using data gathered in Western North America and Western Canada (Butsic *et al.*, 2011). The researchers applied a hedonic regression model on two different datasets. The first set consisted of data published on census tracts – an online database recording property sale from ten states in North America. The second dataset consisted of property sales information in four different market regions, namely: Whistler and Fernie in Canada; Washoe County in Nevada; and El Dorado County in California. By using snowfall statistics gathered through online research and news outlets, Butsic *et al.* (2011) concluded that the overall property prices might decrease in regions where snowfall decreases.

An advantage of regression analysis is that it can usually be incorporated into almost any dataset to find some form of relationship between the variables applicable to the specific model. Results of a regression prediction can be inaccurate if any of the following is encountered (Swanepoel *et al.*, 2011):

- Multicollinearity happens when one or more independent variable in a dataset can be predicted using the remaining independent variables. This is a problem, because the independent variables must have an impact on a dependent variable and not independent variables. These variables are then called statistically insignificant.
- Heteroscedasticity refers to when there is no linear pattern in a dataset. The term linear regression refers to when a dataset follows a certain pattern that makes it easy to predict.
- Autocorrelation occurs when the dataset is non-linear. Regression is considered linear when a dataset of dependent and independent variables follows a certain or specific pattern, allowing a new value for the dependent variable to be predicted. When the dataset is non-linear, an analyst cannot logically predict a new value for a variable in the dataset. An example of autocorrelation in property valuation or the real estate market is when market value fluctuations are caused by certain events occurring over specific periods of time. A property is worth a certain amount at one point in time, then an event occurs causing the value of the property to either increase or decrease. When using a dataset that has property values determined in different time periods, regression can yield

incorrect results unless an accurate compensation measure is put into place (Arcuri *et al.*, 2020).

Even though property value prediction through regression is the most common method of mass-appraisal, there are other techniques with which to calculate the values of a set of properties. Table 2.3 shows some additional mass-appraisal related studies.

Table 2.3: Recent studies in mass appraisal

Authors	Title	Research Theme
Doszyń (2020)	Econometric support of a mass valuation process	Discussed the implementation of an econometric model for mass-appraisal and identified the advantages and disadvantages of using such a model.
Gnat (2020)	Impact of the regularisation of regression models on the results of the mass valuation of real estate	Analysed whether regularised regression can reduce the errors of property valuation.
Miotti and Loch (2020)	Property value map updating by mass-appraisal method – a case in the city of Pato Branco, state Paraná	Used a mass-appraisal method in updating a valuation map for Brazilian municipalities.
Sevgen and Aliefendioğlu (2020)	Mass-appraisal with a machine learning algorithm: random forest regression	Implemented random forest regression to value a large set of flats as a mass-appraisal technique.

In developing a dynamic automated property valuation software for the South African environment, regression will be used to identify dependencies between property attributes and the market value of a property. The next section discusses the use of regression in determining the relationship between independent and dependent variables.

2.4.3 Variable dependency using regression

Regression can be used to effectively determine the association between different types of variables in a specific model (Swanepoel *et al.*, 2011). In the case of property valuation, different types of attributes of a subject property can directly influence its market value. This technique has a wide range of applications but an important contribution in the context of property valuation is prediction. In dependant variable prediction, a prediction value, or p -value associated with each independent variable signifies how well each independent variable can be used to predict a dependant variable (Montgomery & Peck, 1992). In multiple regression analysis, p -values are generated for each independent variable in a dataset. A p -value of less than or equal to 0.05, indicates that an independent variable has an impact on a dependant variable (Swanepoel *et al.*, 2011). The lower the p -values, the greater their impact on the dependant variable. When using regression for prediction in a specific dataset, any variable with a p -value greater than 0.05 is

discarded. Regression analysis is then repeated on the remaining variables and updated until only the most significant independent variables remain (Montgomery & Peck, 1992).

No two properties are identical, which means a program will need a method to distinguish property attributes from each other and identify which attributes to consider when finding properties that are comparable to one another. Regression is considered to be a good technique to determine the relationship between property attributes and market value (Abidoye & Chan, 2016; Amidu *et al.*, 2019; Bozic *et al.*, 2013). In this study, it will be used to determine the impact that each attribute may have on the market value of a property.

Although the use of regression enables the prediction of property market values, these may be estimates and additional decision support in selecting comparable properties can only increase the accuracy of the resulting valuation (Lowies *et al.*, 2016). A system that can assist in providing additional support in such cases, is known as a decision support system (DSS) (Taylor, 2016).

2.5 DSSs in property valuation

A DSS is an information system, mostly incorporated within organisations, to provide options or judgement towards a problem that can contain many solutions (Segal, 2019). Such systems can either solve problems or provide intelligence towards making more informed decisions.

DSSs can filter through large datasets to compile comprehensive solutions for specific problems (Segal, 2020). One of the applications of a DSS is the optimisation of processes in a system (Power, 2002). Consider, for example, an organisation that develops two products. Each product has a specific production cost, requires a certain amount of labour hours for manufacturing and generates a certain amount of income, if sold. The organisation has a certain number of labour hours available and a set budget. It wants to know the maximum income that can be generated. Typically, a DSS can calculate how much of each product the company must produce to maximise its income while adhering to the budget and time constraints.

DSSs allow users to make more informed decisions in different sections of an organisation or for different types of users. The five most common types of DSSs follow (Power, 2002).

- A communication-driven DSS allows several users in an organisation to collaborate, share information and support decision-making without the need for physical interaction. Users are generally divided into their own specific workgroups to focus on their own assigned tasks. These DSSs are often web-based with instant messaging software and have online bulletin boards or project management software. They are also known as group DSSs.

- Data-driven DSSs allow specific information extraction for use in decision-making. A model and a set of inputs allow users to specify certain conditions and, in most cases, a goal according to which data must be extracted towards solving a specific problem.
- The document-driven DSS is one of the most widely used DSSs in the current working environment. It allows users to retrieve and analyse sets of documents to retrieve information regarding a certain subject.
- A knowledge-driven DSS uses expert knowledge to provide decision support or courses of action relating to how a certain task must be accomplished. This type of DSS requires a knowledge base which provides the system with specific expertise in a problem domain. A knowledge base typically consists of a set of rules that stipulate which actions to perform based on the available data. Such rules are generally provided by people who have expert knowledge in a specific domain. Such a system is very similar to an expert system in the field of artificial intelligence.
- In a model-driven DSS, the system must be developed using a specific model relating to the problem domain. Models require certain input variables from the user and generate results that can be interpreted accordingly. With the use of computer software, models can be automated to produce better or more accurate results and reduce processing time. Such a model is highly dependent on the specific problem domain, meaning that you cannot use the same model from one problem domain to solve a different problem. Most model-driven DSSs contain collections of models applicable to specific problem domains.

In order to incorporate a DSS in the sales comparison method of property valuation, it is important to note how a DSS can optimise the sales comparison process. Part of this process is to select a set of comparable properties, therefore a data-driven element can be used to store comparable properties and select the properties most comparable to the input provided (Zyga, 2018).

The other part of the sales comparison method is to adjust the value of the most comparable property according to certain values. A knowledge-driven element can be used to store the property selection criteria as well as the value adjustment rules. The optimal solution for implementing the sales comparison method for property valuation in South Africa will, therefore, be a hybrid DSS consisting of a knowledge-driven and data-driven element. Table 2.4 represents additional research regarding the incorporation of a DSS in property valuation.

Table 2.4: Studies in property valuation DSSs

Authors	Title	Research Theme
Aziz, Aziz and Harris (2012)	Intellectual property valuation decision support system for university research output: a conceptual model.	Discussed how a DSS can assist in selecting the appropriate valuation model for valuing a certain intellectual property (IP) by predicting the outcome of each valuation method and selecting the best one for the current IP.
Keim, Kisilevich and Rokach (2013)	A GIS-based decision support system for hotel room rate estimation and temporal price prediction: the hotel brokers' context.	Identified how a GIS-based DSS can assist in selecting appropriate hotel rooms based on their location and their prices for tourists and travel agencies.
Lausberg and Dust (2017)	Reducing the appraisal bias in manual valuations with decision support systems.	Developed a DSS to provide valuers with additional calculation methods instead of using one method when conducting property valuation.
Evans, How and Lausberg (2019)	Reducing the property appraisal bias with decision support systems: an experimental investigation in the South African property market	Studied whether a decision support tool can assist in reducing behavioural property adjustments when calculating a new value.

2.5.1 Data-driven DSS

A data-driven DSS can assist in selecting the correct set of data from a database, according to boundaries or constraints set by a user, for decision-making purposes (Power, 2002). Examples of data-driven DSSs in property valuation include spatial planning on how to use the space in a property (Christopoulou & Haklay, 2005; Syful Hoque, 2019), selection of the correct comparable properties from a database of properties (Zyga, 2018), or selecting the correct attributes for use in property value calculation (Kokot & Gnat, 2019).

Information regarding a property can be obtained through several means. The most common method is for the valuer to personally inspect the property. Another method is by obtaining information regarding the property from a client. This may include photos, property size measurements, the address, additional assets to sell, or the condition of the property to name a few (Hendrikse, 2015). Technology exists that is readily available to the public that contains large databases of geographic information that can also be utilised in the information gathering process. Such technology is known as geographic information systems (GIS) (Sue, 2017).

The main function of a GIS is to store geographic data, analyse such sets and then visually represent them to a potential user (GISGeography, 2020; Zeng & Zhou, 2001). The data used in such a system can be represented as geographic positions on the surface of the earth. By using the geographical context, a system can better analyse trends, patterns or spatial relationships

(McClusky *et al.*, 1997). GISs allow people to retrieve different types of information such as (Koller *et al.*, 1995):

- retail store or shopping mall locations;
- areas experiencing power outages;
- incidents and analyses of crime;
- route navigation; or
- weather forecasting.

One of the applications of GIS technology in property valuation is in residential properties where a system can determine the closest comparable properties to a property under consideration. Expert valuers usually select properties that are in within a radius of one kilometre of the subject property but need to make major adjustments if comparable properties are situated farther away (Abidoye & Chan, 2016). The GIS can search for properties from a database according to their addresses, calculate the distance to each property and assist in selecting the closest one.

The biggest application of GIS software in property valuation is the provision of assistance in spatial planning (Arcuri *et al.*, 2020; Christopoulou & Haklay, 2005; Zeng & Zhou, 2001) and effective resource management on a geographic plain (McClusky *et al.*, 1997; Zhu *et al.*, 1998). Such programs are commonly known as geo-spatial DSSs (Christopoulou & Haklay, 2005) that uses geographic data stored online or locally and plots optimal areas of use in the given geographic data.

One such application is by Zeng and Zhou (2001) who used a data-driven DSS in combination with GIS technology in property valuation for development planning on an open land. The spatial data was supplied by the user and stored in a database. The GIS mapped the available space and determined the optimal usage of the space. The system was also dubbed a geo-spatial DSS but was primarily a data-driven DSS, considering that the spatial data was stored in a database and used for decision-making.

2.5.2 Knowledge-driven DSS

Sometimes managers in organisations require expert knowledge to solve specific problems. Most tasks can be automated in computer systems, but without the correct method, knowledge, and skills certain tasks cannot be completed. A solution to such problems is known as a knowledge-driven DSS.

Part of the property valuation process is determining which attributes to adjust and how, during selection and calculation. A knowledge-driven DSS can store expert knowledge in the form of rules which can be used to adjust the attribute values of a property during the calculation of its

value (Kilpatrick, 2011). Furthermore, it can also help to determine which attributes of a property to consider when selecting comparable properties (Amidu *et al.*, 2019; Boyle, 1984).

Boyle (1984) provided an example application of a system that incorporates knowledge to conduct a valuation of a property. The system incorporates a database of properties and attributes. Regression is used on the database of properties to determine the importance of each property attribute when calculating a new value through prediction. The system also incorporates a knowledge base that stores certain rules, along with the attribute importance of each attribute, to determine how to adjust a certain attribute during calculation, such as a house with a certain style is worth more than other houses, meaning that an additional adjustment value must be added to the equation. During execution of the valuation task, the user inputs certain attributes of a new property in a question-and-answer format. These inputs are then applied to the value prediction equation, along with the regression coefficient of each attribute and the additional adjustment value if a certain attribute is a certain value. It then calculates a new property value using that applied equation. Boyle (1984) proved that this system provided usable results for an expert valuer, but the system is highly dependent on a consistent dataset of properties. If an inconsistent property dataset is used, multicollinearity becomes a major problem. Boyle (1984) stated, however, that the system can assist in providing a second opinion to an expert valuer and does provide an error term, representing the accuracy of the results if the valuer would consider using this outcome.

2.6 Chapter summary

This chapter introduced the concept of property valuation and continued with discussions on prevalent methods used in valuating different types of properties. Recent research in property valuation methods were considered to provide broader insight into the implementation of the sales comparison and the mass-appraisal methods. The benefit of regression analysis in property valuation was outlined and briefly followed by an introduction to DSSs. Specifically, data-driven and knowledge-driven DSSs, which will be essential sub-components for a DSS that incorporates human expert knowledge with mathematical valuation models, were discussed.

Chapter 2 motivated how a DSS can be used to improve residential property valuation in South Africa. Chapter 3 will purpose to discuss how such a DSS will function and which components are required to perform these functions.

CHAPTER 3 AN EXPERT DSS IN PROPERTY VALUATION

3.1 Introduction

The structure of a decision support system (DSS) depends on the sources of information to consult towards developing the system (Krukeja, 2013). The knowledge domain for the South African residential property environment encompasses the sales comparison method discussed in Chapter 2 and rules as specified by human experts.

In this chapter a hybrid DSS will be constructed in the context of this study in Section 3.2 to include features from both knowledge and data-driven types. This is followed by a layout of the hybrid expert DSS and a discussion on each individual component in Section 3.3.

3.2 Knowledge and data-driven DSSs

Property and property attribute data can be stored efficiently in a database. According to the sales comparison method, appropriate data can be extracted from the database, processed and implemented in a relevant model to determine the current market value of a specific property. Such a system will require a dedicated data storage component from which data *selection* can be made, as well as a knowledge base that will show how to *calculate* a property market value. A hybrid DSS that contains both a data-driven and knowledge-driven element is required for this type of functionality.

DSSs are frequently designed to assist in solving specific problems or automating existing systems and this is also true for the field of property valuation. The valuation process can be lengthy or drawn-out which may lead to either invalid or inaccurate value estimates. In computing the value of a property, a valuer relies on various sources of information like expert knowledge, different mathematical equations, personal intuition, environmental factors, and more. A DSS can effectively combine such information sources in a knowledge base for property valuation. The type of DSS that incorporates the use of knowledge in optimisation, is known as knowledge-driven (Power, 2002). In this study, the functionality of a knowledge-driven system is combined with that of a data-driven system to create the expert DSS as illustrated in Figure 3.1.

The combination of the knowledge-driven element and the data-driven element requires four main components, namely: the *user interface*, *database*, *knowledge base*, and *decision-making module*.

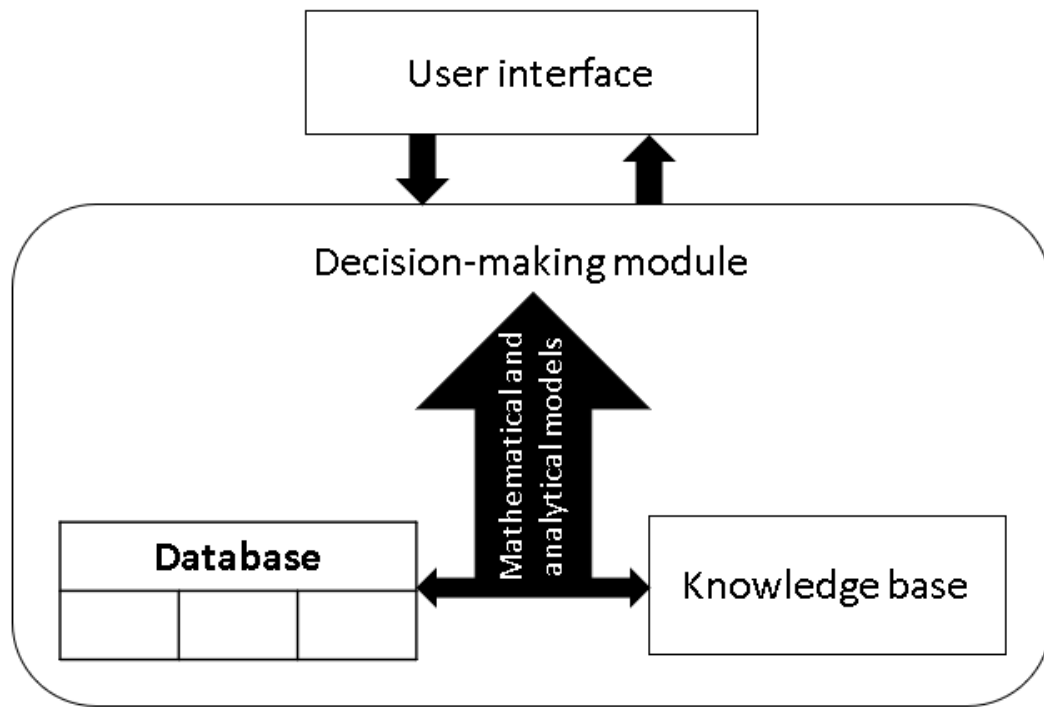


Figure 3.1: Expert property valuation DSS structure

Each of these components is a computer structure in its own right. In the next section, they are discussed in more detail and how to combine them into an expert DSS in the South African property context is presented.

3.3 Components of an expert DSS in property valuation

The aim of this section is to elaborate on the user interface, database, knowledge base, and decision-making module of the property valuation DSS. Each of the components is discussed briefly and rationalised for inclusion into the system.

3.3.1 The user interface

Computer systems can execute complex activities or automate tedious tasks and most require a tool through which users can interact with it. These interactions can include powering the system, displaying information or commencing task execution (Pinard *et al.*, 2013). The component of a computer system which allows interaction is known as a user interface (UI).

In computer development, a UI is the link between the user and the system through which the user can input data and view outputs or results that the system generates after executing a task (Deitel & Deitel, 2012). The very basic UI of most computer systems consists of a command line that prompts a user to input data. Non-developer users of such a system may find this difficult to use when interacting with a system, therefore an alternative tool for interaction must be developed.

The type of UI that contains fields, labels of text, images or navigation pages is known as a graphical user interface (GUI) (Pinard *et al.*, 2013). When designing a GUI, the developer must be aware of and consider the requirements set by potential users as well as experts in the field. The developer must be familiar with the system's required functionality and structures and present them so that they are understandable by the user. In this scenario, a user-friendly design must be incorporated for the system to be used correctly and efficiently. The following factors must be considered when evaluating the *user-friendliness* of software (Whitten & Bentley, 2007):

- *The software is easy to install and implement.* The software must be accompanied by proper information and instructions on the installation process.
- *The software is intuitive,* meaning the GUI is well designed, follows a proper structure, uses a strong format, and is easy to understand.
- *The software functionality is efficient.* It is important for the software to work well with the existing operating system. Tasks must be executed quickly, and the software should not be sluggish. This is ensured through software optimisation.
- *The software contains proper error-handling procedures.* The program must be able to check for errors during installation and execution and implement relevant solutions. This can range from forgetting to input data into certain fields to checking for more complex errors that were encountered during testing and implementing a work-around.

Generally, input fields are presented to users for the values required to solve a problem and results are presented as text, tables, charts or images (Krukeja, 2013). In the property valuation environment, a GUI should typically contain fields to input attributes of a property under consideration, such as the size, available amenities, date of sale or number of bedrooms. The selection fields are generated dynamically from tables in a property database. The system will, therefore, be dependent on a database and any values that are added to it must reflect on the GUI. Apart from dynamic field generation, the database will also contain other important functions which are discussed in the next section.

3.3.2 A property database

In the decision-making module the data to be used for analysis must be structured and connected which may result in a very large dataset. This will significantly increase the resource cost and processing time. There is, therefore, a need for structured property data storage and presentation. In Computer Science, such a structure is known as a database (Coronel *et al.*, 2013).

Data can be defined as collections of facts such as numbers, words or measurements that have been converted into a specific format for use in computer systems (Vaughan, 2017). Apart from

Computer Science, data are used in almost every field, with or without the user's knowledge. Any aspect of life that is perceived by a user can be considered to be data.

One of the reasons why data is important to companies, is that they can use it to analyse trends, forecast sales, track potential customers, or identify competitors (Vaughan, 2017). Data allows companies to improve the overall decision-making process and output better results through data analysis. In the property valuation domain, a user can determine:

- whether more expensive residential properties have marketable views;
- if specific mainstream properties share a certain style;
- whether the suburb has a large impact on the value of a property;
- the impact of the previous date of sale on the property's value; and
- how much the condition of a property would influence its selling price.

A database is a structured data storage formation that can be organised and presented for users (Coronel *et al.*, 2013). The general composition of a database is a collection of tables that contain records regarding a certain object, relationship, or action. Each record consists of a list of attributes that contain additional information about the object of the table. For example, a table containing a list of users would generally consist of a username, e-mail address, password, or date of birth. In addition, each table in a database contains an additional attribute known as an index or primary key to uniquely identify each record.

Manually performing data analysis on large datasets can be difficult and time-consuming. Specific software applications allow users to automate such tasks. Database functionality includes task automation, storing and retrieving records for later use, presenting data in a more readable manner, and automating data analysis, amongst others (Coronel *et al.*, 2013). The database in a data-driven DSS typically contains information needed to solve a specific problem that has been mined from organisational datasets. In many cases, databases are used to predict or forecast values from historic data. Forecast values are validated and stored, thereby continually expanding the database so as to improve the accuracy of future estimates. When working with multiple large sets of data, it is important to always maintain a certain table and data structure according to which data are to be organised, or else certain analysis results might be incorrect, or data might become hard to retrieve and analyse. A database with multiple interrelated and linked tables is known as a relational database (Coronel *et al.*, 2013).

A relational database in the property valuation environment will contain data on properties, comparable based on their attributes. In the DSS for this study, each attribute of a property is represented in a separate table. An *attribute table* represents a different type of attribute that relates to a record in the list of properties. Such a relational database can be used to classify

properties according to type, for example, an open piece of land, flat, or normal residential property. The database can also store additional information about each attribute, such as the average price increase based on a combination of different factors.

The DSS in this study uses a database to store comparable properties with their corresponding attributes, weights for attribute selection, and the adjustment values that must be respected if an attribute does not correspond to that of the property under consideration. Although property data can be efficiently stored in a database, a knowledge base is required to manage the rules that determine how to extract, combine and use it in the decision-making module. The property valuation knowledge base is discussed in the next section.

3.3.3 The knowledge base

Certain techniques and expertise are required to conduct a valid and accurate property valuation. These techniques must be translated into a certain format for use in a DSS (Liao, 2005; Power, 2002). For this study, a knowledge base was designed and implemented to incorporate these techniques.

A knowledge base is defined as a repository that contains procedures and rules that are used to make conclusions based on specific inputs and existing datasets (Giarratano & Riley, 2005). The rules to be implemented depend on the specific context of the problem. In this study, the knowledge base contains rules on how to extract the appropriate comparable properties (selection method) and how to calculate the value of a subject property (calculation process).

The database will be queried during the selection process, but it is highly unlikely to find exact comparable properties. It must, therefore, modify the query by removing attributes until a suitable record can be retrieved. When attributes are removed, relevant adjustments need to be communicated to the decision-making module so that appropriate models are implemented towards compensating for them. Regression is utilised to determine the impact that each attribute will have on the value of a property. Mathematical and analytical models are used for the calculation of a new property value and are discussed in more detail in Chapter 4.

The central component that manages user input and consults the database and knowledge base in order to retrieve a list of comparable properties is the decision-making module. The purpose and function of the decision-making module are described next.

3.3.4 The decision-making module

The decision-making module may be regarded as the brain that controls how the other components interact. It manages and conducts task execution based on user input, database and knowledge base consultation, and model application.

The decision-making module must prompt for user input specific to the property under valuation. It will then identify how to select the appropriate comparable properties by consulting the knowledge base. The database is traversed to find the most analogous dataset, while minding the acquired knowledge on required value adjustments. The decision-making module must function much like a working memory, or reasoning element in an expert system. Upon completion of the selection process, appropriate models are utilised with the resulting dataset of comparable properties, along with the required value adjustments, to calculate the estimated value of a property.

3.4 Chapter summary

In this chapter the functionality of the planned expert DSS was discussed and the type used in this study was identified. The incorporation of a knowledge-driven element in combination with a data-driven DSS into an expert property valuation DSS was first briefly discussed. Then the components of an expert property valuation DSS, including a GUI, database, knowledge base, and decision-making module were elaborated on.

The composition of the discussed components grants the DSS to function like the combination of components in an expert system. This, combined with the expert knowledge required in the knowledge base, makes the expert system software development life cycle the ideal approach for developing the expert DSS in property valuation. The detailed design phases and the implementation of each are unpacked in the next chapter.

CHAPTER 4 EXPERT DSS DEVELOPMENT

4.1 Introduction

Property valuation is generally performed by expert valuers guided by their training and own experiences. Therefore, the involvement of multiple experts during development is crucial to ensure that the property valuation expert decision support system (DSS) is valid. In order to build a system that will accurately emulate and enhance the valuation process a software development lifecycle that will accommodate consultation with experts must be selected. This chapter discusses the functionality of an expert system in Section 4.2 and describes the expert system development lifecycle in Section 4.3.

The lifecycle can be divided into four main subsections which are covered in the following chapters as follows: *preparation* describes the planning and knowledge definition stages and is covered in Chapter 4; *structuring* refers to knowledge analysis and extraction which is discussed in Chapter 5; *implementation* in Chapter 6 is the process of knowledge design, and validation; verification and analysis is covered in Chapter 7 as part of *evaluation*.

4.2 Expert systems

In software development, dynamically changing problems can become so complex that normal algorithmic approaches cannot solve them. Also, when all required information is not readily available leading developers have to rely on intuition and experience to make decisions as is the case in property valuation.

An expert system can be described as a computer program that uses expert knowledge to solve problems that require advanced domain-specific knowledge from a human expert (Maylawati *et al*, 2017). Such knowledge simulates human judgement or behaviour. The knowledge is stored in the form of rules in a knowledge base and is accessed when necessary to find possible solutions based on certain input data.

Considering that an expert system attempts to simulate human behaviour to provide expert solutions for a problem, it is important to develop an expert system in collaboration with a human who is an expert in the specific knowledge domain. The knowledge domain consists of the problem-solving techniques that are available to the system, provided by human experts. One such example is the sales comparison method in the property valuation environment (Giarratano & Riley, 2005). The involvement of a human expert will also ensure that the system is valid, and that performance can be effectively verified (Amidu & Boyd, 2018). Such a system must be developed to follow the same problem-solving behaviour of an expert. The developer must

understand the knowledge domain to correctly transfer the problem-solving capabilities into a computerised system. This is generally achieved by studying real-world problems in the form of case studies (Liao, 2005).

The general representation of knowledge in an expert system is according to an *if-then* rule format. This means that an expert system would receive a fact and derive a conclusion or execute an action according to the corresponding rule in the knowledge base. A *fact* is a statement provided by the user, stating that a certain aspect is true or that a certain event has occurred. A *rule* is a statement stored in the knowledge base that instructs the system on what to do for each fact it receives. A *conclusion* or *result* is a statement on what must happen when certain conditions are met and is determined by the system by implementing rules stored in the knowledge base on a fact from a user. The described system is known as a rule-based expert system (Giarratano & Riley, 2005) and is illustrated in Figure 4.1.

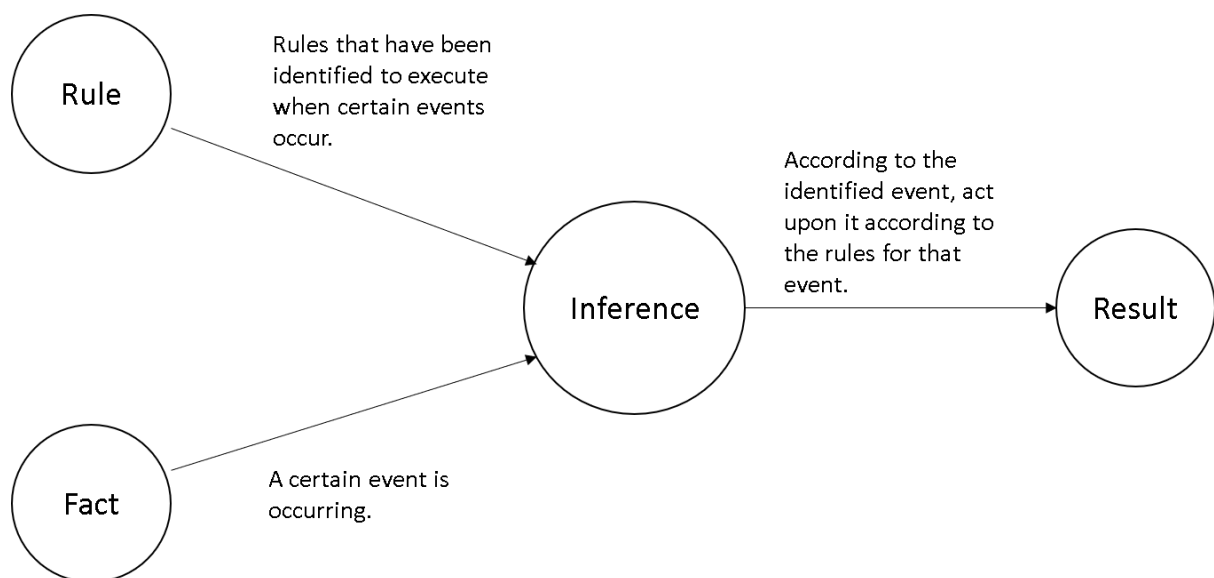


Figure 4.1: Human behaviour when executing an action according to certain events and rules

A human has knowledge on how to act upon certain events when they occur. This process is referred to as logical inference. The expert DSS developed in this study is constructed with a knowledge base and working memory functionality that resemble those of an expert system. Figure 4.2 illustrates the structure of an expert system.

An expert system contains five main components, namely: the user interface, the knowledge base, the inference engine, the agenda and the working memory (Giarratano & Riley, 2005). All of these components work in conjunction to solve a problem.

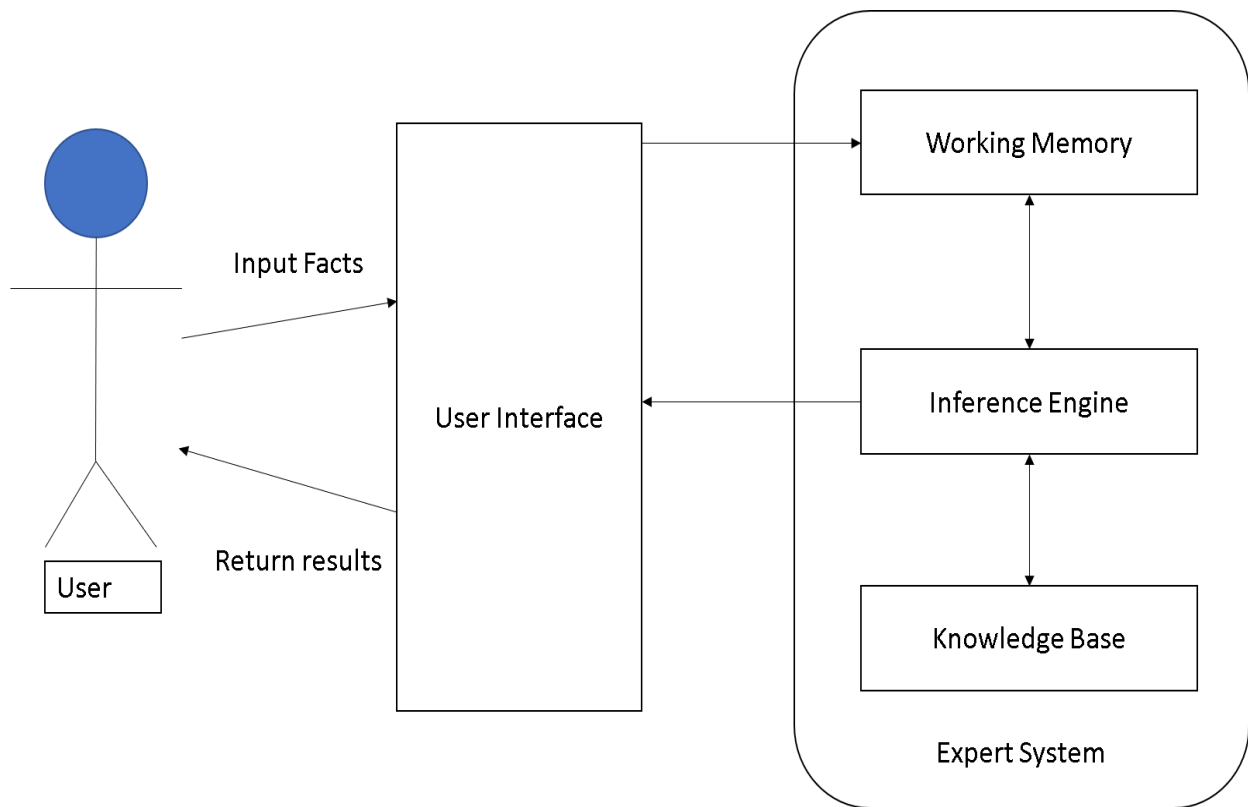


Figure 4.2: The general structure of an expert system

- The *knowledge base* contains expert knowledge procedures that are required to carry out the tasks, and rules that are used to make conclusions through the inference engine (Liao, 2005). The knowledge contained in the knowledge base is gathered through external sources and human experts in the specific application field.
- An *inference engine* works in conjunction with the knowledge base and working memory to derive conclusions based on input data provided by the user. Known facts and observations are used to derive conclusions in a specific field (Surber, 2014).
- The *agenda* contains a list of rules that has been derived by the inference engine. These rules have been selected to satisfy the facts stored in the knowledge base and is executed in a certain order. Each rule is assigned a priority, depending on how important each rule is towards solving the problem or deriving a conclusion.
- The *working memory* contains data that is relevant to the current problem that is being solved (Liao, 2005). Such data differs for every unique problem and determine how the inference engine obtains conclusions by referencing the knowledge base.
- The *user interface* is the tool through which the user interacts with the system. This component is connected to the working memory to provide facts, and the inference engine to receive a result of behaviour on how to act on the facts.

In this study, the knowledge base contains rules on how to select comparable properties according to the subject property, along with the adjustment values of each attribute that must be

considered in order to calculate the value of a new property. Instead of an inference engine, a decision-making module will be implemented to determine which attributes of the subject property to use for comparable property selection and which attributes need to be adjusted according to the sales comparison method, before calculating the estimated property value. Due to the similarities between the planned system and expert systems in general, the expert DSS in property valuation is developed according to the expert system development lifecycle.

4.3 Expert system development lifecycle

There are several available development lifecycle models that can be implemented to develop an expert system. The most common development lifecycles include the waterfall, code-and-fix, incremental and linear models.

- *The waterfall model*, like the linear model, is divided into sequential steps allowing the developer to either revisit prior steps or continue with a current one while the next step commences (Vhutshilo, 2019). Each step is dependent on the previous step and the developer must ensure that the work done in each step is evaluated before continuing.
- *The incremental model* is used when the system is divided into sections and each section is developed, tested and evaluated incrementally (Vhutshilo, 2019). This means that each time a new section of the software is developed, the previous section has been fully evaluated and tested. For example, developing the user interface first, testing it and evaluating it before developing any other components.
- *The code-and-fix model* involves developing the entire system in sections, allowing users to test and then fix the errors as they are found in the system (Giarratano & Riley, 2005). This method is not preferred, because when a system is developed from start to finish without proper evaluation of each developed component, it leaves the system in an incredibly unfinished state and requires rigorous testing of each component before launch. The user or client of the system then executes the system testing phase.
- *The linear model* is a very common expert system development lifecycle that contains all of the necessary steps to develop a functional expert system (Giarratano & Riley, 2005). Every step in the linear model contains sub-tasks to accomplish to ensure the completeness of development. Figure 4.3 illustrates the steps in the linear model.

4.3.1.1 Task phasing

Task phasing involves identifying the required tasks of the development lifecycle in each phase. The phases and tasks identified for the development of the expert DSS in property valuation are shown in Table 4.1.

Table 4.1: Task phases of the expert system development lifecycle

Stage	Tasks
Planning	• Task identification • Feasibility assessment • Preliminary functional layout • High-level requirements
Knowledge definition	• Knowledge source identification and selection • Knowledge acquisition, analysis and extraction
Knowledge design	• Knowledge classification • Detailed design
Code and checkout	• Program development and coding • System tests • System evaluation

4.3.1.2 Feasibility assessment

The feasibility assessment phase is used to determine the validity of the study by answering certain expert related questions. Before developing a system, it is important for the developer to know how to find the optimal solution to a problem. The following questions must be answered.

1. *Can the problem be solved by means of conventional programming?* The knowledge domain must be discussed in detail. Considering that expert systems are used to solve unconventional problems, the developer must identify whether existing systems are unable to solve such a problem. In the property valuation environment, other software applications exist that solve the valuation problems by different means with varying accuracy. However, considering that alternative systems use conventional valuation approaches that do not compensate for environmental and economic factors, it is difficult to value residential properties in the fluctuating South African property environment by means of conventional programming. It is for this reason that the current system is being developed.
2. *Is the knowledge domain of the problem well-bounded?* In the context of property valuation, the valuer's knowledge involves how to conduct, for instance, the sales comparison method on a residential property. The developer must identify the boundaries of the knowledge domain, for example, not to use the income capitalisation method when the sales comparison method is appropriate. The identified knowledge domain includes South African residential property valuation, using the sales comparison method.

3. *Is there a need for an expert system in the current domain and setting?* The developer must identify through research whether users require an expert system to solve the valuation challenges in the South African property market. Usually, experts in the knowledge domain will be able to identify whether an expert system will solve a problem. An expert DSS can assist a valuer in selecting the most comparable property and applying applicable models as determined by problem-specific inputs and appropriate knowledge. Therefore, the need for an expert DSS in property valuation exists.
4. *Is there at least one human expert who is willing to provide the necessary knowledge to the system developer?* Experts in the knowledge domain can provide a more practical approach for the developer to understand. In this study, human experts have been identified who were willing to assist with the knowledge engineering process. The knowledge engineering process involved gathering the necessary knowledge to be incorporated in the system (Giarratano & Riley, 2005). The experts consulted in this knowledge domain are seniors in the field with many years of personal experience in residential property valuation. The consulted experts in this study included Nathen Theron, Chris van der Vyfer, Alet Wentzel and Tersia Prinsloo.
5. *Can the expert explain the knowledge in such a fashion that the developer can convert it into an expert DSS?* It is important that the experts can explain their approaches to property valuation in such a manner that the developer can automate, not only the mathematical models, but also the intuitive solution techniques that are generally used. The expert must also be willing to answer any questions that the developer may have. In this study, the identified property valuation experts were thorough and open to discussions. They were willing to share their knowledge and intuitive approaches to the developer.
6. *Is the knowledge used for problem solving in the knowledge domain mainly heuristic and uncertain?* An expert system can function more efficiently when knowledge is used that have been tested and proven through practice. The knowledge to be used in the property valuation knowledge domain of the sales comparison method have been used thoroughly by the consulted expert valuers. These valuers focus primarily on residential properties and have provided many examples on how to conduct the sales comparison method through past property valuations. This method is heuristic in nature, meaning that the calculated property value might not be the most optimal method, but is sufficient enough for the valuation process.

After the feasibility assessment is completed, the preliminary functional layout is built. Here, the developer set-up a base design for the expert system and identifies a few main components to be used.

4.3.1.3 Preliminary functional layout

The purpose of this expert DSS is to assist valuers in conducting the valuation process, specifically when determining the value of a residential property in South Africa. This was accomplished by utilising the sales comparison method and a set of selected comparable properties.

The expert DSS uses an interface to enable the user to select the attributes of the property under consideration (the subject property) and then retrieve comparable properties from the database. The attributes that need to be included in the database, therefore, need to be identified beforehand. Having many attributes for comparison will increase the accuracy of the valuation results but will limit data availability of comparable properties significantly, because there are very few sources that provide a full description of attributes of a property without a physical inspection. Most sources merely provide the sale price and a few other redundant descriptions of a property, such as its owner.

It is highly unlikely that two properties will be one hundred percent comparable to each other, therefore, the system will compare the attributes of a property to those from similar properties in the database. The basic function of the expert DSS is to list the comparable properties, select the property with the most comparable attributes, determine the value adjustments that need to be made and implement them in the calculation model, and output the estimated market value to the user. The next step of the development lifecycle is to identify the high-level requirements for the system to be developed and execute its tasks.

4.3.1.4 High-level requirements identification

Before developing an expert DSS, the developer must identify which technologies would be required to solve a problem in the knowledge domain. This describes the high-level requirements and includes identifying technologies such as the programming language, the development environment, or the hardware requirements, and suchlike.

The expert DSS will require a platform for the user to interact with (the user interface) to input the attributes of a subject property and receive results, a programming language that executes tasks, and a database tool to store comparable properties and attribute adjustment values. The user interface and any corresponding task execution functionality were developed using the windows presentation foundation (WPF) desktop application tools. They include the extensible application mark-up language (XAML) and C# to provide the user with an interactive platform with which to input data and carry out property value calculation and comparable property selection. Database engines, such as OLEDB for C# is used to communicate with the database written in Microsoft

Access. Additional regression functionality to determine the attribute weights was implemented through a program written in the Python coding language, using a library known as *Statsmodels*.

In the planning phase, task phasing was completed, the feasibility of an expert DSS for property valuation was discussed, the basic layout for the DSS was presented, and the technology that is required to develop the expert DSS was identified. The next section continues the discussion on the development lifecycle, referring specifically to how knowledge would be obtained and how to execute tasks in the sales comparison method.

4.3.2 Knowledge definition

The knowledge definition phase involves the identification of the rules that form the knowledge base of the system. This is achieved through two main tasks, namely: knowledge source identification and selection, and knowledge acquisition, analysis and extraction.

4.3.2.1 Knowledge source identification and selection

The developer must identify the sources from which the knowledge will be acquired, identify its availability and list them in order of importance. Afterwards the developer must select the sources appropriate for use in the system.

The sources that were used for the expert DSS presented in this study include interviews with experts, journal articles of previous research in the field, electronic and online sources and textbooks. Journal articles are considered to be one the most reliable sources when it comes to research because they provide extensive proof of research through numerous citations and references, as well as the fact that they have been extensively reviewed to ensure validity. Most sources on property valuation provide insight into the incorporation of a DSS in some valuation field (Bencure *et al.*, 2019). Research on property valuation systems more commonly utilise neural networks, multiple linear regression, and Bayesian trees (Nejad *et al.*, 2007; Yildiri, 2019). Online and electronic sources provided some information on obscure terminology and helped identify possible production software. Textbooks were used to define topical terminology.

Expert knowledge is one of the most reliable sources of knowledge because experts can provide practical feedback in a certain field, along with their own opinions on the relevant knowledge domain (Giarratano & Riley, 2005). This type of knowledge engineering can be performed by means of conducting interviews or surveys. Both of these options were implemented in this study. The interview was conducted with a local expert valuer after which questionnaires were compiled, validated by the expert with whom the interview was conducted and sent to expert valuers living and practicing in different towns. The nature of the knowledge required for the expert DSS renders the interviews and questionnaires to be the most important source of knowledge.

4.3.2.2 Knowledge acquisition

The knowledge engineer needs to comprehensively understand the functions performed by practicing property valuers before building an endorsed knowledge base. The knowledge acquisition phase was used to determine the best method for the developer to incorporate the system in the knowledge domain. The information that was used to build the system, was acquired as follows:

- Interviews were conducted with local professional property valuers.
- Questionnaires were compiled in collaboration with expert valuers and returned to them for comments.
- Journal articles and textbooks were studied to gain comprehensive insight into the variety of approaches and methods, as well as ongoing research in property valuation and DSSs.

The knowledge that was acquired during the practices described above was used to compile the problem-solving rules that form the knowledge base of the expert DSS. Each of these is discussed in the following paragraphs.

4.3.2.3 Interviews and observations

The purpose of the study and the concept of a DSS was explained to the experts and a request was made to conduct an interview with them. The wide range of valuation techniques and developmental research in property valuation, means that no single model was identified for use in the planned system at the time of the interviews. Accordingly, the interviews focused on the optimal approach to property valuation in the South African context.

The boundaries of the identified knowledge domain include property valuation, the residential environment, the country of South Africa, and the sales comparison method. It is important to understand how the expert DSS can be incorporated in the property valuation environment and contributions it can make. The following information was gathered (Theron, 2020; Van der Vyfer, 2020; Wentzel, 2020):

1. From the identified methods and models in property valuation, the *sales comparison method* seemed to be most commonly used to determine the value of a residential property. For business properties, the income capitalisation method is typically used.
2. The valuers consulted, focused more on the valuation of *residential properties*. One of the reasons for this is that the market range of residential valuations is larger than the other sections of property valuation, meaning valuers get more work opportunities through the valuation of residential properties.

3. The value of a property may not be *increased*, or the property may not be modified in any way to increase its value during the valuation process. This issue was addressed by implementing a model-based DSS in the property valuation environment, because a typical DSS aims to optimise a process by either maximising or minimising a specific objective.
4. The most common use of DSSs in property valuation is in the land management or township development section. In such cases, valuers determine the values of properties as if a building project or land management project was to be performed on that property. In such an instance the expert valuer will determine the best courses of action for the leader of the project to incorporate in order to maximise the outcome of the project. Projects can include the effective use of natural resources, optimised placement of certain equipment, or effective planning of districts. The use of a DSS for project optimisation is not relevant to this study and will therefore not be considered further.
5. During the valuation of a residential property through the sales comparison method, a valuer needs to decide on which properties to be used for comparison. A DSS presents the ideal platform for providing decision support in selecting the most comparable properties.

Upon completion of the interviews, the sales comparison method was selected for implementation in the expert DSS in the residential property valuation environment.

4.3.2.4 Document analysis

To ensure that the information gathered was varied and supported by the insights of the expert valuers, additional research articles were studied to confirm the application of the software and the steps of the valuation process. Common valuation approaches were identified in Chapter 2. When comparing properties, the valuer must make the necessary adjustments to compensate for the differences between the subject property and the comparable property. In this study, the goal is to minimise the number of adjustments that is required when selecting the comparable properties and calculating a value estimate for the subject property. Additional sources such as electronic documents and articles (Amidu *et al.*, 2019; Arcuri *et al.*, 2020; Birkeland & D'Silva, 2018; Ellerman, 2004) were also studied to determine the amount that the program needs to adjust the value with.

A database of property attributes for a region needs to be constructed for comparative purposes. The required attributes must be identified and, using valuation reports provided by expert valuers and online sources, the database must be populated with the properties and their corresponding attributes.

This section described how knowledge was acquired. With the knowledge gathering process concluded, the next step is to analyse the knowledge and construct a basic functional layout for the expert DSS. This is covered in Chapter 5.

4.4 Chapter summary

The involvement of expert valuers during development of an expert DSS for property valuation motivated the implementation of the expert system development lifecycle. This chapter commenced by briefly discussing the concept of expert systems, along with the expert system development lifecycle. Next followed discussions and execution of the planning and knowledge definition stages.

The next chapter continues with the expert system development lifecycle by analysing the gathered knowledge and constructing a functional layout for the expert DSS. Upon completion of knowledge engineering, development of a prototype could commence.

CHAPTER 5 EXPERT DSS STRUCTURE

5.1 Introduction

Once the problem-solving procedure that needs to be implemented in the expert decision support system (DSS) has been established, its structure can be designed. This involves identifying the necessary tasks and phases to perform as part of the sales comparison method and developing the mathematical model necessary to incorporate expert knowledge and accommodate dynamic changes that may occur.

In this chapter, the design of the structure of the expert DSS is discussed in the knowledge analysis and extraction phases in Section 5.2 and 5.3. The knowledge analysis phase describes the tasks required to perform the sales comparison method including the most important attributes of a property to consider, how to select comparable properties, and how to calculate a new market value. In the knowledge extraction phase, the order in which identified tasks need to be completed to correctly implement the acquired knowledge, is defined.

5.2 Knowledge analysis

The knowledge analysis phase is necessary to identify and structure the knowledge essential for correct system performance. Before a property can be compared to others the attributes useful for comparison must be identified. This process is discussed next.

5.2.1 Identifying property attributes

Various factors can influence the value of a property. Each property has a set of unique attributes that contributes to estimating its value. Identifying these attributes are an important part of property valuation, regardless of the valuation method that will be used to determine final value. These attributes are usually identified by a valuer when doing a physical inspection of the property.

It is important to determine the most marketable attributes of a property when developing automatic valuation models for automated valuation for accurate value estimations. Property information that is available to the public is usually limited to the location of the property, the size of the property, and the value of previous sale of the property. According to the South African Council for the Property Valuers Profession, the most important attributes of any residential property that must be identified, are the following (Ellenberger *et al.*, 2010; Mills, 2019):

- number of garages;
- presence of flat(s);

- number of bedrooms;
- number of bathrooms;
- number and purpose of out building(s);
- amenities; and
- condition of the building(s).

These attributes enable valuers to identify properties comparable to the subject property. Other important attributes that may contribute to selection accuracy include:

- the dates of sale of comparable properties;
- the estimated values of comparable properties;
- the neighbourhood or suburb of the subject property; and
- the distances between the comparable properties and the subject property.

The property database will be used to store comparable properties with their corresponding attributes, as well as adjustment values for the different types of attributes to be used by the knowledge base. This adjustment value has been retrieved along with the population of the database from external sources, such as *Property24*, the National Deeds Office and Expert Valuers. Each attribute used for valuation is represented by a unique table in the database, along with an adjustment value which will be utilised by the knowledge base in calculating a new property value. These tables are discussed in detail in Chapter 6. Figure 5.1 represents the database structure designed for the expert DSS.

The attributes of a property play an important role in the calculation of a new property value. With the most important property attributes identified, the next step is to identify a relevant model to determine how the expert DSS would calculate the value of a new property. The next section briefly discusses the sales comparison method in the context of the expert DSS, along with its relevant mathematical model.

5.2.2 The sales comparison method and mathematical model

The method used to determine market value in residential properties is the sales comparison method. The expert DSS selects the best comparable dataset from the property database, while minimising the number of attribute differences between that and the subject property. For demonstration and validation purposes, the DSS was developed for implementation in the town of Klerksdorp, South Africa. Accordingly, the property database contained suburb and property datasets from this area. The sales comparison method can be expressed as a procedural process and is provided as follows.

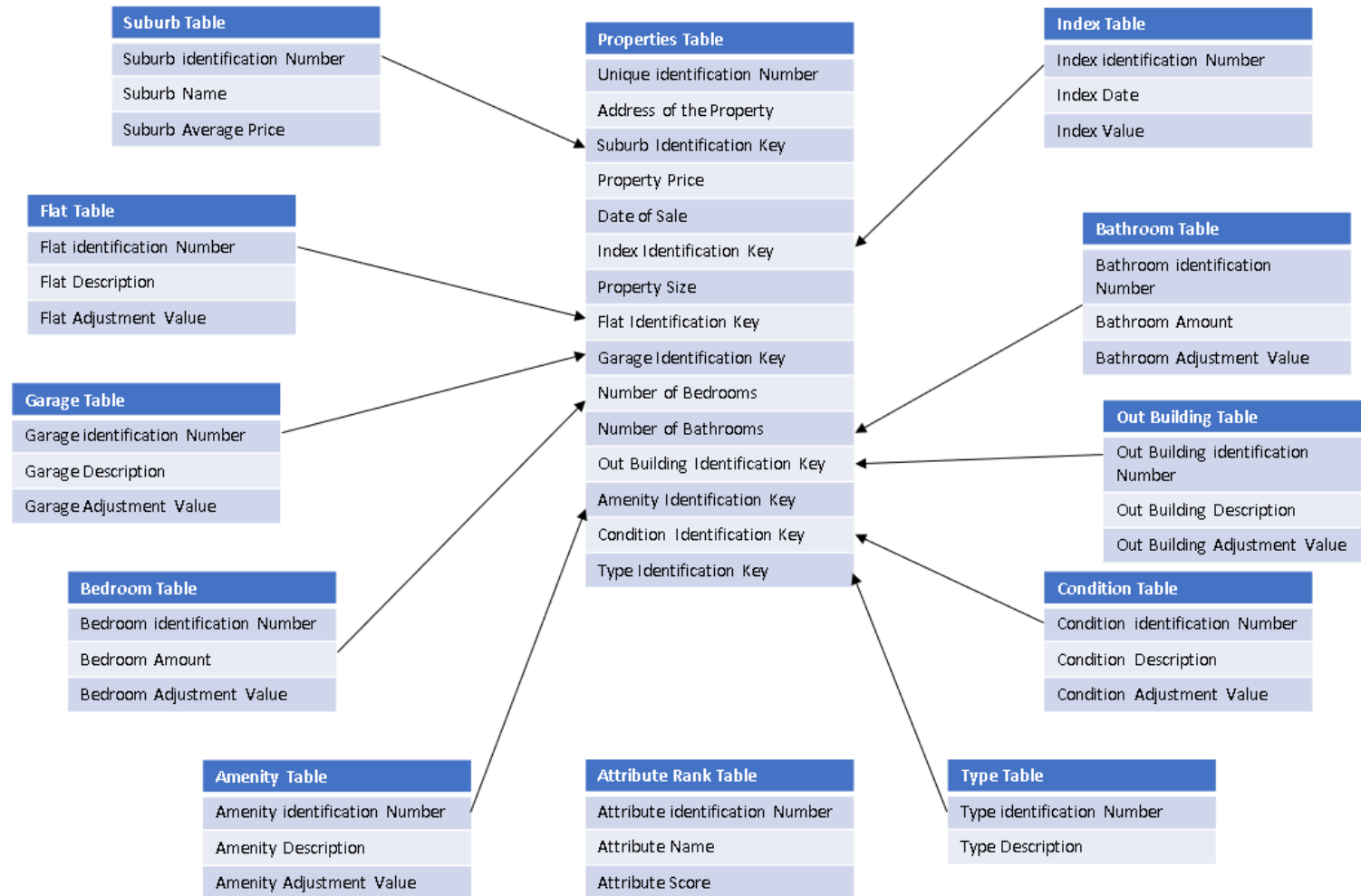


Figure 5.1: Database structure for the expert DSS

Step 1: Identify the subject property.

Before valuation can be carried out, the subject property must be identified, along with its corresponding attributes. In this step, the valuer conducts what is known as a physical inspection of the property after agreeing on a date of inspection with the owner of the property.

Step 2: Obtain comparable properties.

The valuer must then obtain a list of properties comparable to the subject property. In South Africa, valuers have access to the national Deeds Office to request property reports or online databases that store property information, such as *Lightstone property* (Hendrikse, 2015).

When a list of comparable properties have been compiled, the valuer must inspect each property on the list to determine whether all of its attributes are listed. The most common inspection methods are physical inspections and online inspections by means of GISs.

A physical attribute inspection is usually performed along with the identification visit (step 1) and can prove to be an incredibly lengthy process. Sometimes comparable properties must also be physically inspected if the data provided from external sources do not describe the comparable property's attributes. Online inspection entails the use of GIS technology like Google Maps with which valuers can identify the necessary attributes of each comparable property, after retrieving a list of such properties with their associated addresses and market values from an external source (Ellenberger *et al.*, 2010). This approach is risky because not all internet sources are verified, and some value adjustments can then be made incorrectly. In such cases, valuers need to use a credited verification process to confirm the validity of the acquired information. Valuers use this approach when a physical inspection is not possible due to the lack of transportation, dangerous or hazardous circumstances, or lack of time. After the list of comparable properties have been identified the valuer compares them with the subject property and selects the three with the least differences in attributes.

Step 3: Calculate the value of the property.

From the three identified properties, the one that is the most similar to the subject property is selected. Alternatively, the property with the latest sale date or located closest to the subject property is selected. The market value of the subject property (mv_c) is expressed mathematically as

$$mv_c = p + \sum_{i=1}^m x_i, \quad (5-1)$$

where p represents the sale price of the most comparable property that has been identified; x_i is the value adjustment for the i^{th} attribute; and m is the number of attributes to be adjusted for.

The above-mentioned steps identified how an expert valuer would typically execute the sales comparison valuation process. Algorithm 1 expresses this process in terms of how a computer system would execute it.

Algorithm 1: Execution of the sales comparison method	
Input:	x_1 represents an array of all the attribute values of the subject property.
Output:	mv_c represents the calculated market value of the subject property, according to the sales comparison method.
<pre> 1. Declare x_1; 2. Declare $x_2 = \text{empty}$; 3. Declare $x_c = \text{empty}$; 4. Declare $mv_c = \text{empty}$; 5. Declare $qry = \text{selection query commands}$; 6. For $attr = 1$ to n do 7. $qry = qry + x_1[attr]$; 8. Search for comparable properties[qry, x_2]; 9. If $x_2 > 0$ and $x_2 \leq 3$ then 10. For $compProp = 1$ to n do 11. For $attr = 1$ to n do 12. If $x_1[attr]$ not $x_2[compProp][attr]$ then 13. $x_2[compProp].reqAdj += 1$; 14. $x_2.orderBy(reqAdj)$; 15. $x_c = x_2[1]$; 16. If x_c not empty 17. $mv_c = x_c.marketVal$; 18. Declare $x_i = \text{empty}$; 19. For $attr = 1$ to n do 20. If $x_c[attr]$ not x_1 then 21. $x_i.add(x_1[attr].attrValue - x_c[attr].attrValue)$; 22. For $attr = 1$ to n do 23. $mv_c = mv_c + x_i[attr]$; 24. Output (mv_c) </pre>	

The variable x_1 represents a list of attribute values of the subject property. x_2 represents the list of comparable properties where each comparable property is represented as its own list. x_c is the variable representation of the most comparable property that the system uses for the

final property value calculation (mv_c). The next variable qry is used to select comparable properties from the database, that is inserted into the variable x_2 . qry consists of the attributes that have been identified in the variable x_1 . After a set of at least three comparable properties (x_2) have been found by the database query qry , each property in x_2 is compared with the subject property x_1 to identify the number of required adjustments for each attribute that is different. The comparable property from the list variable x_2 with the least number of attribute differences is then set into the variable x_c . If the variable x_c contains a comparable property, the final calculation is executed where the value of the most comparable property x_c is adjusted for each attribute that is different than the subject property. Then, this final value is set into the variable mv_c and displayed to the user.

Now that an algorithm and a mathematical model has been identified for how a new property value is to be calculated, all of the previously described tasks in Section 5.2 must be combined into a functional layout describing how the expert DSS will execute the sales comparison method, which is described in the next section.

5.3 Knowledge extraction

Knowledge extraction describes the process in which the functional layout of a system is set, and the phases required to perform the functions are described. It also identifies the requirements according to which the system will be evaluated.

5.3.1 Functional layout

The expert DSS should provide decision support to a property valuer in the South African environment. Upon completion of the knowledge analysis phase, the functions that the system should perform have been identified.

Using the expert DSS to calculate an estimated market value for a property will start by allowing the user to enter an abstraction of the subject property. This will require the user to provide inputs regarding the subject property as accurately as possible. To ensure user-friendliness, property attributes will be pre-listed allowing the user to select them from a structure referred to as a drop-down list. Each item in the drop-down list will be obtained from the property database for a specific area.

When an abstraction of the subject property has been generated, the DSS will consult the knowledge base and select comparable properties from the database using a query as described in Section 5.2.1. During selection, the query is continuously adapted, by removing

attributes with a high weighted value (discussed in Chapter 6). The selection stops when at least three properties have been retrieved.

The DSS then identifies the number of adjustments that are required and sort the comparable properties according to that number. The property that is the most comparable to the subject property is used for calculating the market value. This value, along with all the adjustments that were made, is compiled into a report and displayed to the user for explanation purposes.

The steps required for performing the functions of the DSS include:

1. Prompt the user to input the attributes of the subject property.
2. Select a set of comparable properties from the property database.
3. Calculate the number of adjustments that are required for each comparable property.
4. Display the comparable properties and the adjustments that are required to calculate the new value to the user for confirmation and exception handling.
5. Calculate and display the new value of the subject property based on the finalised parameters.

This section marks the end of the knowledge analysis and extraction phases of the system development lifecycle. With the functional layout and design in mind, development of the expert DSS can commence.

5.4 Chapter summary

The purpose of this chapter was to explain how the structure for the expert DSS was designed. This was achieved through the knowledge analysis and extraction phases. Knowledge analysis comprised identifying the property attributes relevant in the sale comparison method as well as how the sales comparison method will be executed in the context of an expert DSS. As part of knowledge extraction, the functional layout of the DSS was established.

In Chapter 6, the development of the expert DSS in the form of a computerised demonstrator is discussed. The development is achieved through knowledge classification, detailed design, and code-and-checkout phases.

CHAPTER 6: DEVELOPMENT AND IMPLEMENTATION OF THE EXPERT DSS

6.1 Introduction

The preliminary functional layout of the expert decision support system (DSS) presented in Chapter 4 was modified and finalised in Chapter 5 upon completion of the knowledge definition, analysis and extraction phases. This means that the knowledge base that will be utilised to search for the optimal calculation considerations can now be developed. Consequently, knowledge design, and code-and-checkout can be completed.

In this chapter, system task flow and mathematical considerations are discussed during the knowledge design phase and implemented in the expert DSS in Section 6.2. This includes relevant value adjustments leading up to formulation of an overarching mathematical model for calculation of a property value in the South African environment, discussed in Section 6.3. Verification of the functional performance of the expert DSS is done throughout this chapter and concluded in Section 6.4 by applying case-based data collected during the initial stages of this study.

6.2 Knowledge design

During knowledge design a detailed model of the system must be built. This is achieved by means of knowledge classification and detailed design. Before these tasks are executed, it is important to define how the knowledge will be represented in the *knowledge classification task*. The basic user interface must be conceptualised according to the structure developed in the previous phases. Sample data and dynamic scenarios are utilised for evaluating the system.

The *detailed design task* results in a comprehensive report describing system structure, implementation, and evaluation requirements. The layout of the development chapters of this mini-dissertation is in the form of a cumulative design report meaning that once knowledge classification is completed, code-and-checkout can commence. In the development of a DSS, the knowledge classification phase defines how the general structure of the program will function, while detailed design focuses on implementation.

6.2.1 Knowledge classification

Coding and implementation of the components established for this DSS means that the development platform selected should be able to integrate advanced programming with a

database structure allowing for implementation of mathematical modelling techniques. The C# development platform provides object-oriented and functional design as well as specific database integration for use in the property valuation domain (Wagner, 2021). Property data were constructed as objects, allowing data access without complex coding structures. The universal technology base, *Microsoft*, linking *C#* to *Access* means integration of these two development programs is the ideal solution for coding the expert DSS enabling compilation of database queries in structured query language (SQL) (Heller, 2019; Landry, 2019). The combination of these tools used in conjunction with specialised modules in the Python programming language forms the development platform for the expert DSS. The database will be used to store attributes and property data for use in calculating property values. Property attributes are used to build a virtual representation, or model, of the property (subject property) for which an up-to-date market value is required.

Once the attributes of a subject property are known, a comparable property with a recent and known market value must be found from the database. The value of this property, known as the *most comparable* property, will be used as a base for calculating the value of the subject property. All properties in the database must be examined to find those with attributes most similar to the subject property. It is highly unlikely that a property identical to the subject property will be found. Therefore, the query used to search the database will be iteratively updated according to the rules in the knowledge base. The initial query used will represent an exact abstract model of the subject property. When no comparable property is found, an attribute is removed from the query and the database is searched again. This process is repeated until at least three comparable properties are found. Because each type of attribute can have a different impact on the market value, regression analysis was performed to calculate a priority ranking for them. This was achieved by means of *Statsmodels* which is a module in the Python programming language that provides estimation capabilities of many and various statistical models (Perktold *et al.*, 2020). Regression analysis can be performed every time the database is populated with a dataset from a different region. During every next iteration of database query modification, the attribute with the next lowest priority is removed from the query. However, for each attribute removed from the query a value adjustment must be made according to a specific mathematical model.

Two types of attribute adjustments can be made, namely: fixed structural attribute adjustments and what is referred to as umbrella attribute adjustments. Structural attributes are represented by fixed-cost values that will be added or subtracted directly from the value of the property, subject to certain conditions. The values for these structural attributes were obtained from

external sources as discussed in Chapter 5 (Brümmer, 2019; Wentzel, 2020). The structural attributes that can be adjusted for (if applicable) are determined by the:

- number and size of garages;
- number, type, and size of any out-buildings on the property;
- number of bedrooms;
- number of bathrooms; and
- number, type, and size of amenities.

Umbrella attributes refer to indirect property factors that impact its market value. The adjustment ratio for each attribute is calculated according to its own unique model and multiplied with the property value. Umbrella attributes include:

- neighbourhood;
- date of last valuation; and
- general property condition.

Structural and umbrella attributes play a large role in the selection of comparable properties when applying the sales comparison method. Priority ranking of the attributes is shown in Table 6.1.

Table 6.1: Property attribute priority rankings

Attribute name	Score
Suburb	0.000067
Amenities	0.003496
Condition	0.013823
Number of bathrooms	0.133004
Garage	0.565373
Flat	0.690482
Domestic quarters	0.788372
Index	0.808116
Number of bedrooms	0.91753

Although umbrella attributes have a more significant impact on the market value of a property, both types of attributes must be considered to ensure system integrity. Adjusting for both types of attributes is discussed in more detail later in Section 6.3. The functions of the expert DSS are presented in Figure 6.1.

Most of the inputs required from the user are constructed as drop-down lists, each of which is populated with data from the property database. The reasoning behind this is that different adjustment approaches are used for each attribute.

Although implementation of the expert DSS can extend to all qualifying properties, three are selected for the purpose of comparing the results with real cases. The reasoning for this is that expert valuers narrow down their search of comparable properties to only three and this will enable accurate verification of the DSS (Ellenberger *et al.*, 2010).

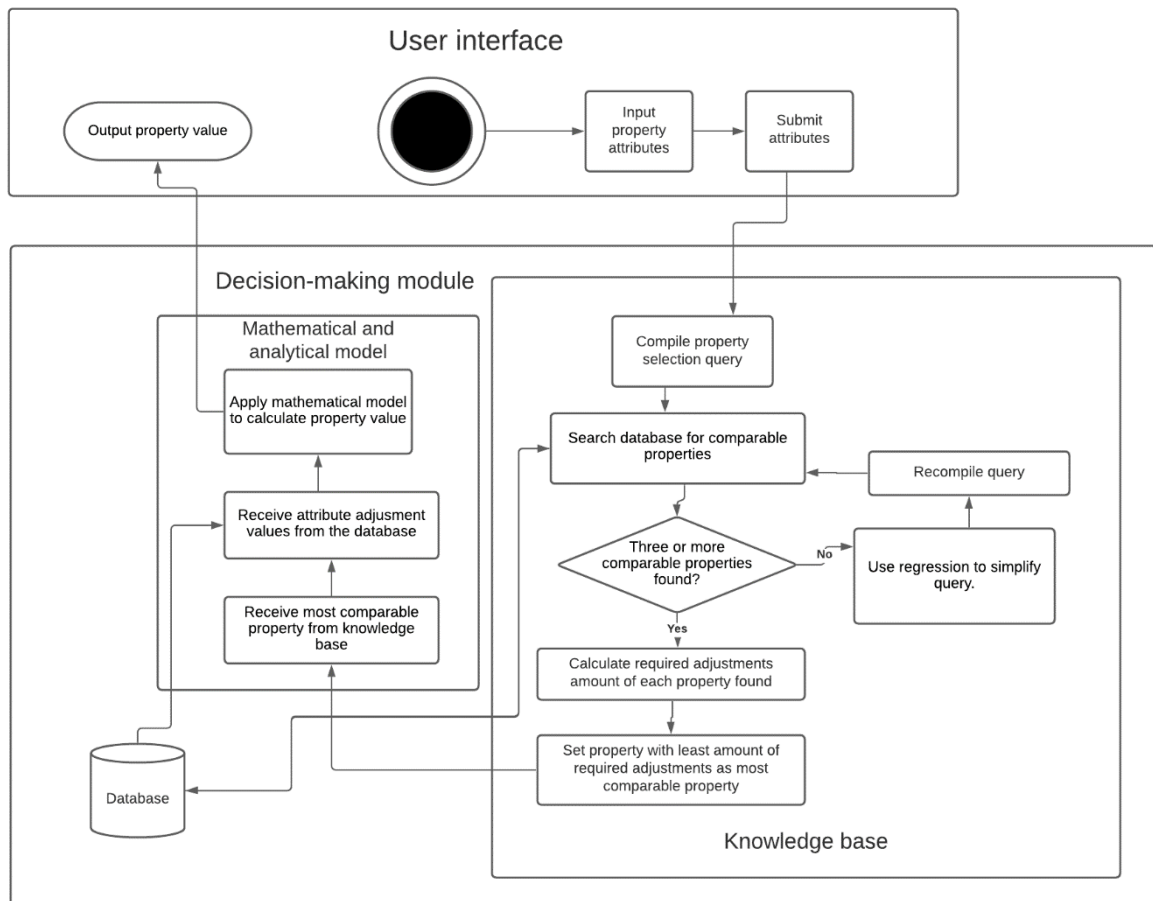


Figure 6.1: Flow diagram demonstrating step sequence and functions of the expert DSS

Next, the decision-making module determines the number of adjustments that are required by means of the mathematical model relevant to each attribute. Subsequently, the property with the least number of required adjustments is selected and used to calculate the market value for the subject property. If more than one property exists with the same number of required adjustments, the expert DSS selects the property with the latest date of sale which is the one umbrella attribute not included in the database query. Expert valuers agree that the property with the most recent date of sale or valuation can be used as the ultimate filter in selecting the most comparable property because of external influences, such as natural disasters, failing or flourishing economy, and more (Anuar & Nor Hana Asykin, 2010; Hendrikse, 2015).

The knowledge classification phase is concluded with the completion of conceptualisation of the system task flow. The next section continues on knowledge design by discussing the detailed implementation and coding of the expert DSS.

6.2.2 Detailed design: code-and-checkout

Upon completion of knowledge classification, the structural layout of the expert DSS has been finalised and the detailed design in which the components are constructed and their accompanying code are created, can commence.

This section describes design and development of the components required to perform the identified functions of the expert DSS. Example inputs are used to illustrate the DSS operation.

The first step is to obtain input from the user. When a new property valuation is initiated the user is prompted for information through several input fields that represent the attributes of a property – such an input screen is shown in Figure 6.2.

Property valuation for the town of Klerksdorp

Please complete the following fields:

Property address	Sample Address	Latest index according to database: 117,3 for the month of Dec of the year 2020 Check latest index online
Property suburb	Wilkoppies	
Date of valuation	2020/08/07 15	
Property type	Residential dwelling	
Property size(m ²)	1200	
Property flat description	No flat	
Property garage description	Single garage	
Number of bedrooms	3	
Number of bathrooms	2	
Out buildings description	Standard/domestic quarters	
Amenities description	No amenities	Calculate new value Back
Condition description	Habitable and properly maintained	

Figure 6.2: Subject property input screen

Default values for the required input fields are extracted from the property database. The database utilised for system validation purposes was populated with property data selected from the town Klerksdorp and linked to the DSS through the *suburb* table. This table contains a list of suburbs, along with an average property price for each as obtained from *Property24.co.ca* (Property24, 2020), local estate agents (Theron, 2020; Van der Vyver, 2020; Wentzel, 2020) as well as the *Deeds Office Pretoria database* (Windeed, 2020) as on 30 July

2020. The property database utilised for queries can be updated at any time and expanded for any region in South Africa, subject to data availability. The remaining attribute input fields contain values that were pre-set by an expert valuer and can be changed by the user according to features best describing the subject property. Comparable property selection is done according to the process described in Section 6.2.1 and summarised in Table 6.2.

Table 6.2 The selection of comparable properties

Selection of comparable properties	
Step1:	Receive the subject property's attributes from the user interface as input.
Step2:	Retrieve the attribute scores from the database.
Step3:	Compile a selection query using the attribute values that have been identified in step 1 and place them in an order of importance according to their rank, identified in step 2.
Step4:	Execute the selection query on the database to retrieve comparable properties.
Step5:	If not at least three properties have been identified, remove an attribute from the selection query from the last position of the attributes list.
Step6:	Recompile and re-execute the query.
Step7:	Repeat steps 5 and 6 until at least three comparable properties have been retrieved.

The DSS dynamically extracts the current consumer price index (CPI) for South Africa from Moody's Analytics (2020) which is the official auditing authority for the government. The date of valuation is compared to those of the comparable properties and value adjustments are made accordingly, if applicable. The expert DSS's database contains CPI values from 1 January 2010 up until the current time frame. When new CPI values become available, the expert DSS can search online on the above-mentioned source for the latest CPI values and then update the database.

The retrieved comparable properties are presented to the user upon completion of the selection process. Figure 6.3 presents an example of a list of comparable properties determined during a developmental trial. The addresses of the properties were changed to test addresses to hide the identity of the actual properties.

According to the information provided, the following properties are the most comparable:

Adjustments	Property Address	Suburb	Property Price	Date of Sale	CPI Identification Number	Property Size	Flat
2	Property A	Wilkoppies	2080000	4/23/2018 12:00:00 AM	107	575	No flat
3	Property B	Wilkoppies	900000	8/13/2019 12:00:00 AM	113.1	1448	Standard open plan o
3	Property C	Wilkoppies	1300000	5/20/2019 12:00:00 AM	112	1318	Standard open plan o

Figure 6.3: An example list of comparable properties

The *most comparable property* is presented to the user with information on the type of value adjustments that will be required for any attributes that differ between it and the subject

property. The user has the option to save information on all of the listed comparable properties in a text file for reporting purposes. This is presented in Figure 6.4.

The most comparable property is:

Property A

Attributes that differ between the most comparable property and subject property:

Date of sale
Garage

Next

Save Comparables

Figure 6.4: Required attribute adjustments for the most comparable property

Once the most comparable property has been identified, the market value for the subject property is calculated and displayed to the user. Market value calculation is discussed in the next section.

6.3 Calculation of market value

The expert DSS uses an adjusted sales comparison model to calculate the value of a subject property. This model can be expressed as:

$$mv_s = \prod u_i (mv_c + \sum s_j) \quad (6-1)$$

with $i \in \{suburb, date, condition\}$,

$j \in \{garage, outbuilding, bedroom, bathroom, amenity\}$,

where mv_s is the market value of the subject property, u_i is the value adjustment for umbrella attribute i , mv_c represents the latest value of the most comparable property, and s_j is the value adjustment for structural attribute j .

Adjustments for structural attribute differences are added to the value of the most comparable property and for umbrella attribute differences are multiplied with it to determine the market value of a subject property. These adjustment scenarios are discussed in Sections 6.3.1 and 6.3.2.

6.3.1 Structural attribute adjustments

Structural attributes represent the physical attributes that describe the structure(s) built on a property. Each attribute is assigned a fixed value that can be directly added to or subtracted from that being calculated, depending on the differences between the subject property and the most comparable property. Structural adjustments can be made for garages, out-buildings, the number of bedrooms, the number of bathrooms and the amenities. The adjustment value for each attribute is predetermined by an expert valuer and external sources (Brümmer, 2019; Wentzel, 2020) that is stored in a database, but may be subject to change in the future.

The following tables from the database, combined into Table 6.3, contain the value of each field of each structural attribute in the database. The number of bedrooms and bathrooms are multiplied according to a predetermined value and is then subtracted or added. Out-buildings consist of the flat contained within a property, as well as whether the property has servants or domestic quarters and is stored within the database as two different tables.

Table 6.3: Database table representing the structural attribute adjustment values

Structural attribute tables	
Description	Value
Garage Table	
Car port only	R15,000.00
Single garage	R50,000.00
Double garage	R85,000.00
More than 2 garage/garage and carport	R95,000.00
Flat Table	
Standard open plan one bedroom	R150,000.00
Open plan 2 bedrooms	R200,000.00
Domestic quarters Table	
Standard/domestic quarters	R50,000.00
Bedrooms Table	
Adjustment per bedroom	R100,000.00
Bathrooms Table	
Adjustment per bathroom	R45,000.00
Amenity Table	
Swimming pool	R100,000.00
Built-in barbeque/playroom/bar	R150,000.00
Swimming pool plus other amenity	R250,000.00

All of the aforementioned structural attributes are added or subtracted from each other, depending on the difference between the subject property and the most comparable property. A structural attribute value is added when the subject property has an attribute that the most comparable property does not have or if the subject property has a better version of that attribute, such as having more bedrooms. On the other hand, a structural attribute value is subtracted from the value if the opposite of the above scenario occurs. To use an example

equation, the following attribute adjustments must be made for the subject property, compared to the comparable property (Table 6.4):

Table 6.4: Example structural adjustments

Structural attribute adjustments required	
Adjustment value	Description
$s_{outbuilding} = (500000 - 400000) = 100000$	The comparable property has a bigger flat compared to subject property.
$s_{garage} = (0 - 50000) = -50000$	The subject property does not have a garage, while the comparable property has a single garage.
$s_{amenity} = (0 - 100000) = -100000$	Comparable property has a swimming pool whereas the subject property does not.

All the other structural attributes remain the same, therefore they have no impact on the value calculation. The total structural attribute adjustment for the subject property using the most comparable property in the example will be -R50,000.00.

In addition to structural adjustments, indirect attribute differences must also be considered when comparing property values. Such cases are referred to as umbrella attributes. They have a greater influence on the market value of a property than structural attributes and the model for these adjustments is discussed in the next section.

6.3.2 Umbrella attribute adjustments

When determining the value of a property using an adjusted sales comparison model, an expert also considers what is referred to as umbrella attributes of the subject and comparable properties (Hendrikse, 2015; Theron, 2020; Van der Vyfer, 2020; Wentzel, 2020). To construct a mathematical model that addresses regional and overarching influence consistently, the distance between the subject property and possible comparable properties had to be constrained. Experts concur that properties within a one-kilometre radius of the subject property qualify for selection. However, should the number of properties that qualify be less than three, alternatives may be selected on condition that value adjustments based on differences in umbrella attributes are implemented.

It was established during knowledge analysis that when calculating the value of a property, an expert valuer seriously considers the average recent prices for properties in the same suburb as the subject property. Properties from different suburbs are not generally in the same price range. According to an expert valuer, considering the same suburb as the subject property can still be efficient in determining an accurate property value, due to many properties sharing

more or less the same market value range with a property under consideration (Ellenberger *et al.*, 2010; Wentzel, 2020).

Considering that completely comparable properties are rarely found in the same neighbourhood or suburb, the DSS uses a suburb table to make adjustments for such differences. Such a suburb table for the town of Klerksdorp is shown in Table 6.5.

Table 6.5: Average property values for different suburbs in Klerksdorp

Suburbs: Klerksdorp	
Suburb name	Average price (R)
Wilkoppies	R1,787,000.00
Flamwood	R1,200,000.00
Doringkruin	R1,098,500.00
La Hoff	R888,200.00
Flimieda	R870,000.00
Neserhof	R850,000.00
Meiringspark	R800,000.00
Ellaton	R650,000.00
Adamay view	R672,833.00
Roosheuwel	R540,833.00
Elandsheuwel	R550,000.00

Value adjustment of the subject property based on its suburb is calculated as the ratio of the latest average price of properties in the same suburb and those in the suburb of the comparable property. This is achieved by dividing the suburb value of the subject property with that of the most comparable property. The resulting value is referred to as the *suburb coefficient* (u_{suburb}) and will be multiplied by the structurally adjusted comparable value ($mv_c + \sum s_j$).

Consider, for example, a property in Wilkoppies (the subject property) and a property in Flimieda (the comparable property). The average price of a property in Wilkoppies is R1,787,000.00, while in Flimieda the price is R870,000.00. The resulting suburb coefficient for this example will have a value of $u_{suburb} = 1787000/870000 = 2.05$.

Another factor that has a notable influence on property values, is the national CPI. In financial terms, the CPI measures monthly changes in prices for a range of consumer products and records the up-to-date inflation rate of a country (Moody's Analytics, 2020). The property database contains monthly CPI values dating back to the start of the year 2010 up until the current date. The CPI of a country changes on a monthly basis (Moody's Analytics, 2020), therefore, the system can dynamically update the database on request of the user.

Table 6.6 shows the change in the CPI for South Africa over the time period between January and July 2020.

Table 6.6: Change in the monthly CPI for South Africa (January to July)

Index table	
Index year	Index value
01/01/2020	114,1
02/01/2020	115,2
03/01/2020	115,6
04/01/2020	115
05/01/2020	114,3
06/01/2020	114,9
07/01/2020	116,4

In order to accurately determine the value of a subject property when compared to properties from a different time period, an index adjustment relating to the date of the latest valuation of a comparable property needs to be implemented. Consequently, an *index coefficient* needs to be calculated which represents the change in property value since the latest month of sale or valuation. The index coefficient (u_{index}) is calculated similarly to the suburb coefficient as the ratio of current CPI and that of the latest date of sale or valuation of the comparable property.

Consider, for example, a subject property that is valuated in July 2020 with a valid index value of 116.4, while the comparable property was sold in February 2020, with an index value of 115.2. The resulting index coefficient will be $u_{index} = 116.4/115.2 = 1.01$.

Another factor that can dynamically change the value of a property is its condition. Experts generally rate the condition of a property according to one of three levels (Ellenberger *et al.*, 2010). For the purpose of demonstrating its influence on the property value and calculating a *condition coefficient* ($u_{condition}$), relevant adjustment values are assigned. This is shown in Table 6.7.

Table 6.7: Possible adjustments for property condition

Condition of the property	
Description	Adjustment value
Habitable and properly maintained	0
Standard maintenance required	5
Structural maintenance required	20

The *condition coefficient* ($u_{condition}$) is expressed as the inverse percentage of the difference in condition rating between the subject property and the comparable property. It is expressed as:

$$u_{condition} = 1 - \left(\frac{s_{condition} - c_{condition}}{100} \right) \quad (6-2)$$

A subject property that, for example, requires structural maintenance and a comparable property that is habitable and properly maintained will result in a condition coefficient of

$$u_{condition} = 1 - \left(\frac{20 - 0}{100} \right) = 0.8.$$

The umbrella attribute adjustments must then all be multiplied with each other. Any umbrella attribute that does not require an adjustment, receives an adjustment value of one. The following is an example of how the umbrella attributes are used for adjustment (Table 6.8):

Table 6.8: Example umbrella adjustment table

Umbrella attribute adjustments required	
Adjustment value	Description
$u_{suburb} = (1787000/870000) = 2.05$	The subject property is in the Wilkoppies suburb, while the comparable property is in Flimieda.
$u_{index} = (116.4/115.2) = 1.01$	The subject property was valued in July 2020, while the comparable property was valued in February 2020.
$u_{condition} = \left(1 - \left(\frac{20 - 0}{100} \right) \right) = 0.8$	The subject property requires maintenance, while the comparable property is habitable.

The total umbrella attribute adjustment is a value of $2.05 \times 1.01 \times 0.8 = 1.66$ which will be multiplied with the sum of the market value of the most comparable property and the relevant structural attribute adjustments.

All of the attribute adjustment values discussed throughout section 6.3.1 and 6.3.2 play an important role in the calculation of a new property value, because they can adjust the value for major attribute infrequencies. The next section discusses the total property valuation process with an example valuation.

6.4 Case-based expert DSS demonstration

This section shows how the system would execute the valuation process using the sales comparison method with an example property to be valued. The property and its corresponding attributes shown in Table 6.9 will be used.

To preserve owner privacy an anonymous address (22 Sample Street) is used. The first step in calculating the market value is to identify its most comparable properties. Implementation of the designed components into an electronic expert DSS makes it possible for the system to use all of the properties that compare to the subject property in its calculation. However, for the purpose of demonstration and validating system design the three most comparable properties are selected as is the case in an actual expert valuer-performed valuation.

Table 6.9: Sample property for demonstration purposes

22 Sample Street, Klerksdorp	
Attribute description	Value
Suburb	La Hoff
Date of previous valuation	7 August 2020
Size	800 m ²
Flat	Standard open plan one bedroom
Garage	Double garage
Number of bedrooms	3
Number of bathrooms	2
Domestic quarters description (out-building)	Standard
Amenities	No amenities
Condition	Habitable

For 22 Sample Street, the list of comparable properties is shown in Figure 6.5. To protect the rights of the property owners, the street addresses have been renamed to property A, B and C. These are not the actual addresses that are displayed by the expert DSS.

According to the information provided, the following properties are the most comparable:

Adjustments	Property Address	Suburb	Property Price	Date of Sale	CPI Identification Number	Property Size	Flat
2	Property A	La_Hoff	R800,000.00	11/4/2019	113.5	1360 m ²	No flat
2	Property B	La_Hoff	R720,000.00	10/22/2018	109.4	1456 m ²	No flat
2	Property C	La_Hoff	R850,000.00	8/2/2018	108.4	1467 m ²	Standard open p
< >							

The most comparable property is:

Property A

Attributes that differ between the most comparable property and subject property:

Flat
Date of sale

Next

Save Comparables

Figure 6.5: Properties comparable to the 22 Sample Street

The adjustments column in Figure 6.5 indicates the number of adjustments that would be required if the relevant property was to be selected as the most comparable. For this example, each property listed will require a total number of two adjustments, which means the expert DSS needs to adapt the selection of the most comparable property. To accomplish this, the comparable properties are sorted according to their dates of sale/latest valuation. If a situation is encountered as in the example where the number of required adjustments is the same for

all comparable properties, an expert valuer would perform the sales comparison method on the most recently valued property when calculating the market value of a subject property (Hendrikse, 2015).

This criterion causes property A to be the most comparable. The attributes that need to be adjusted for are the flat (the subject property has an open plan single-bedroom flat and the comparable property does not) and the date of sale (the subject property was valued in August 2020, while the comparable property was sold in November 2019).

According to the information accumulated, the market value of the subject property can be calculated. This is done as follows:

- The base price of the subject property will be R800,000.00.
- The total structural adjustment is R150,000.00.
- The total umbrella adjustment is 1.03.

The calculations and calculated market value of the subject property are shown on screen. An example of such output can be seen in Figure 6.6.

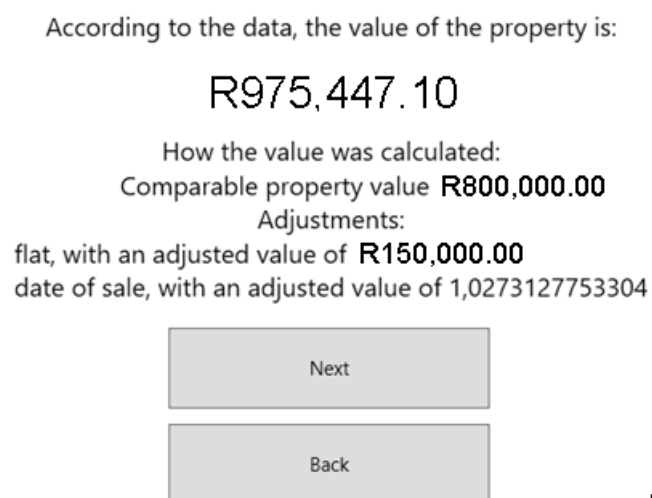


Figure 6.6: Calculated market value of 22 Sample Street

Now that a working demonstrator of the expert DSS has been developed, the code-and-checkout phase of the development lifecycle has been concluded. The next step is to evaluate the expert DSS by using historic data and determining whether the system shows positive application in a real-world scenario. This will be achieved in the next section of the development lifecycle, namely the system evaluation phase.

6.5 Chapter summary

The goal of Chapter 6 was to describe the development and implementation of the expert DSS as designed in preceding chapters. The task flow and mathematical considerations of the expert DSS were discussed. Inclusion of structural and umbrella adjustments in an overarching mathematical model for the calculation of a property value were provided. The functional performance of the expert DSS was verified throughout this chapter by utilisation of case-based data collected during the onset of this study.

In Chapter 7 the expert DSS will be validated by means of formal evaluations. These evaluations will include property valuation data sets generated by human experts and from the official municipal valuation roll.

CHAPTER 7: SYSTEM EVALUATION

7.1 Introduction

The purpose of Chapter 7 is to evaluate the expert decision support system (DSS) that has been developed in this study. This is achieved through formal tests which were performed to substantiate the validity of the system in a real-world scenario.

Sections 7.2 starts by discussing how the system's validity is evaluated through formal tests and includes a summary of the expert DSS's results. This is then followed with a brief system evaluation summary in Section 7.3. Next, a few recommendations are made towards possible future implementations for the system in Section 7.4 and is afterwards concluded with a brief description of possible contributions to the relevant fields of this study in Section 7.5.

7.2 Knowledge verification

In the knowledge verification phase the expert DSS is formally tested using historic data to validate its function and performance. The results are documented and analysed. This phase constitutes validation of the expert DSS.

The first step involved was obtaining historic property valuations. Datasets for validation purposes were attained by means of input documents distributed among expert valuers who assisted in knowledge engineering during the initial stages of development of the expert DSS. An example of an input document in which the sales comparison method was implemented can be found in Appendix A and an example of the mass-appraisal technique in Appendix B. Both documents also provide the estimated market value of the properties. See Appendix C for all the property valuation input data that were completed by the expert valuers and municipal valuers roll, summarised from the documents into a tabular format.

System validation was done by generating valuations on 22 properties in the town of Klerksdorp and comparing them to historic valuations performed previously. Half of the historic valuations were performed using the sales comparison method and the other half by means of the mass-appraisal technique. The town consists of 11 main suburbs and the valuations were conducted on at least one property from each suburb.

The addresses of the properties used for system verification are referred to numerically so as to protect the rights of the owners. Table 7.1 contains an example of a dataset received from the expert valuers.

Tabel 7.1: Property valuation test input

STREET ADDRESS	Property 2 (M2 for example)
VALUER'S VALUE	620000
SUBURB	ROOSHEUWEL
PROPERTY TYPE	RESIDENTIAL
FLAT DESCRIPTION	NONE
GARAGE/CARPORT	CARPORT
NUMBER OF BEDROOMS	3
NUMBER OF BATHROOMS	2
OUT-BUILDINGS	STANDARD: X NONE:
AMENITIES	POOL
CONDITION	GOOD
PROGRAM RESULT	XXXXXXXXXXXXXXXXXX

Valuation results generated by the expert DSS were compared to that of the expert valuers and municipal valuation roll. To determine the accuracy of the valuations calculated by the expert DSS their correlation with the two input datasets were calculated.

The term correlation refers to the relationship between two variables or two sets of data (Fernando, 2020). This is usually represented through a coefficient value that is calculated using a specific equation. In this case, the Pearson correlation coefficient equation is used and is expressed as:

$$r = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_j - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2 \sum_{i=1}^n (y_j - \bar{y})^2}}, \quad (7-1)$$

where x_i represents the value of the i^{th} item in the first dataset (in this case, the result produced by the expert valuers), $\bar{x}$ represents the average value of the first dataset, y_j represents the value of the j^{th} item in the second dataset (the results calculated by the expert DSS) and the average value of the second dataset, represented by the variable $\bar{y}$. Correlation is usually expressed as a value between one and minus one. If the value is not within this range, it means an error was encountered during calculation. A positive correlation is closer to one while a negative value is closer to minus one (Fernando, 2020).

Additional to correlation between the valuations generated by the expert DSS and the case studies, each individual valuation set was also compared. This is expressed as a similarity index, represented as a percentage value. To calculate the similarity index, the two types of valuation results, namely the expert DSS's results and that of the historic valuations, are divided by each other. The smaller valuation result is divided by the larger valuation result so that the resulting similarity index value remains lower than one. The closer the similarity index is to one, the more accurately the expert DSS calculated a new property value. The formal test results are presented in tabular format.

In Tables 7.2 and 7.3 valuations generated by the expert DSS are compared to those from the expert valuers and the municipal valuation roll, respectively. Each table also shows a correlation value (r) for the specific dataset.

Tabel 7.2: Valuation results from the expert DSS compared to the expert valuers

Property identifying value	Suburb	Valuation from expert valuer (R)	Valuation produced by DSS (R)	Similarity index
E1	Elandsheuwel	1235000	1124812	0.91
E2	Wilkoppies	2200000	2175225	0.99
E3	Roosheuwel	500000	521956,1	0.96
E4	La Hoff	1350000	1116243	0.83
E5	Doringkruin	1300000	1241297	0.95
E6	Meiringspark	600000	812986,9	0.74
E7	Neserhof	710000	633482,1	0.89
E8	Adamayview	320000	616049,2	0.52
E9	Flamwood	1450000	1400000	0.97
E10	Flimieda	1200000	1000000	0.83
E11	Ellaton	460000	472500	0.97
Correlation (r)	0.96			

Tabel 7.3: Valuation results from the expert DSS compared to the Municipal Valuation Roll

Property number	Suburb	Valuation valuation (R)	Roll	Valuation produced by DSS (R)	Similarity index
M1	Flamwood	1200000		2119684	0.57
M2	Roosheuwel	620000		644028,8	0.96
M3	Doringkruin	1060000		1300879	0.81
M4	Flimieda	850000		860154,4	0.99
M5	Meiringspark	720000		633888,9	0.88
M6	Adamayview	620000		608600,2	0.98
M7	Ellaton	640000		492001	0.77
M8	Elandsheuwel	720000		847329,3	0.85
M9	Wilkoppies	1750000		4024692	0.43
M10	Neserhof	780000		789073,3	0.99
M11	La Hoff	780000		714428,7	0.92
Correlation (r)	0.98				

A strong positive correlation is observed between the valuations generated by the expert DSS and both those of the expert valuers and municipal valuation roll. Furthermore, the similarity indices indicate the ratio by which the valuations calculated by the expert DSS are similar to those from the expert valuers. An interesting observation that can be made is that although the sales comparison method and the mass-appraisal technique are fundamentally different, the valuations performed by the expert DSS shows a positive correlation with results from both of these approaches.

This is illustrated in Figures 7.1 and 7.2, respectively. The blue line represents the valuation results from the historical data and the orange line represents the valuation results produced

by the expert DSS. The x-axis represents the number of each property while the y-axis represents the value that was produced by the expert DSS and historical datasets. The closer the lines are to each other, the more accurately the expert DSS's results are compared to the historical datasets.

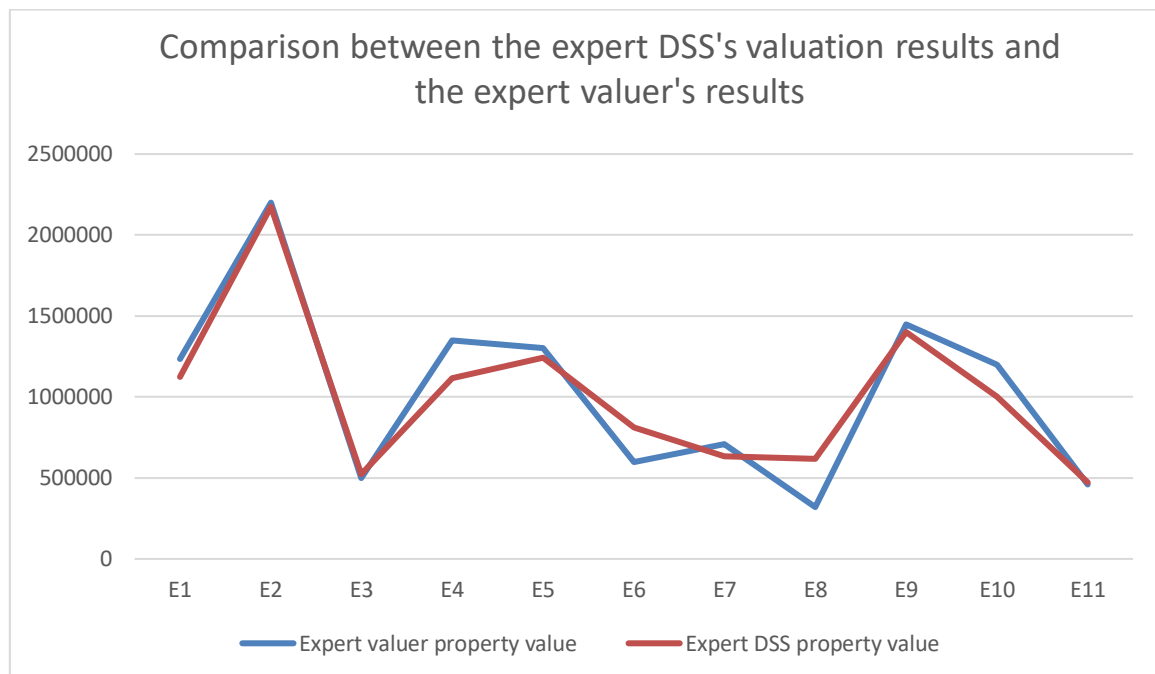


Figure 7.1: Expert valuer results compared to the expert DSS's valuation results

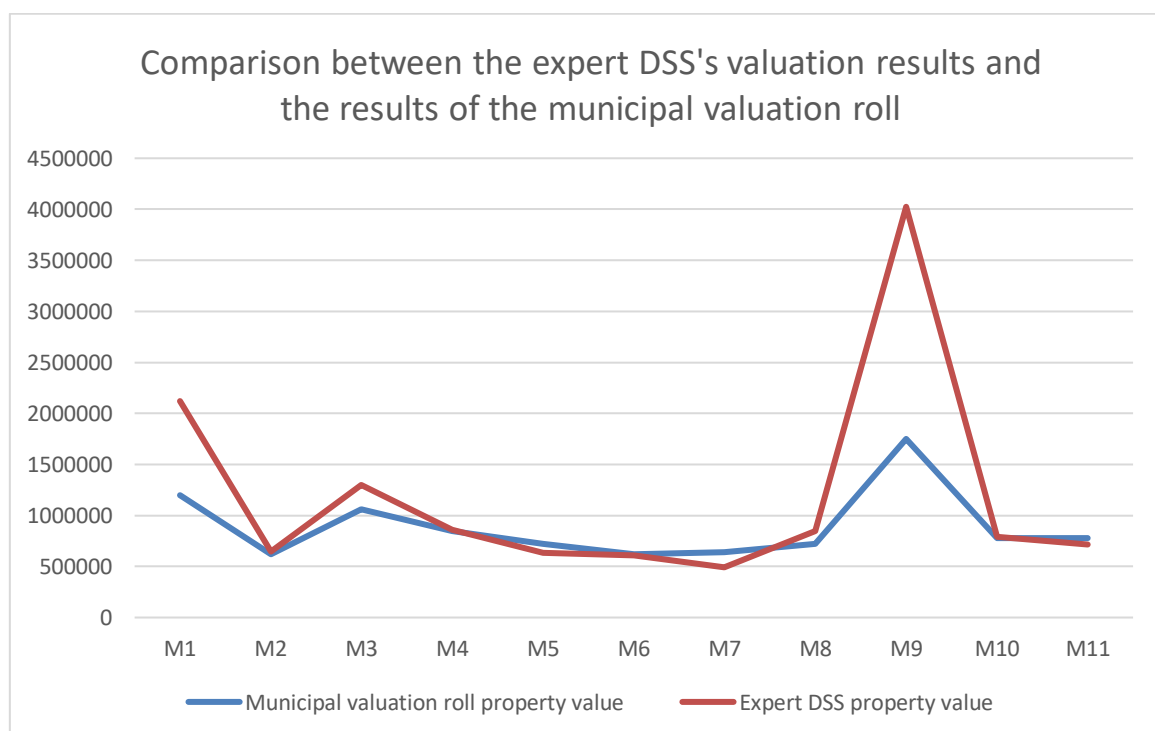


Figure 7.2: Expert DSS valuations vs. those from the municipal roll

Although in the majority of cases the valuations are close to one another, exceptions were observed in the properties numbered E8, M1 and M9. In these cases, the most recent market values of the comparable property selected differed from the average property value for the suburb.

The expert DSS prioritises the selection of comparable properties from the same suburb as the subject property but sometimes the most recent sale price of the comparable property is an atypical representation of the average property value in the suburb. This can be the case if a comparable property was sold at a much higher or lower value than it is worth (Ellenberger *et al.*, 2010; Mohale *et al.*, 2016; Wier, 2020). The cases where this type of exception was observed are shown in Figure 7.3.

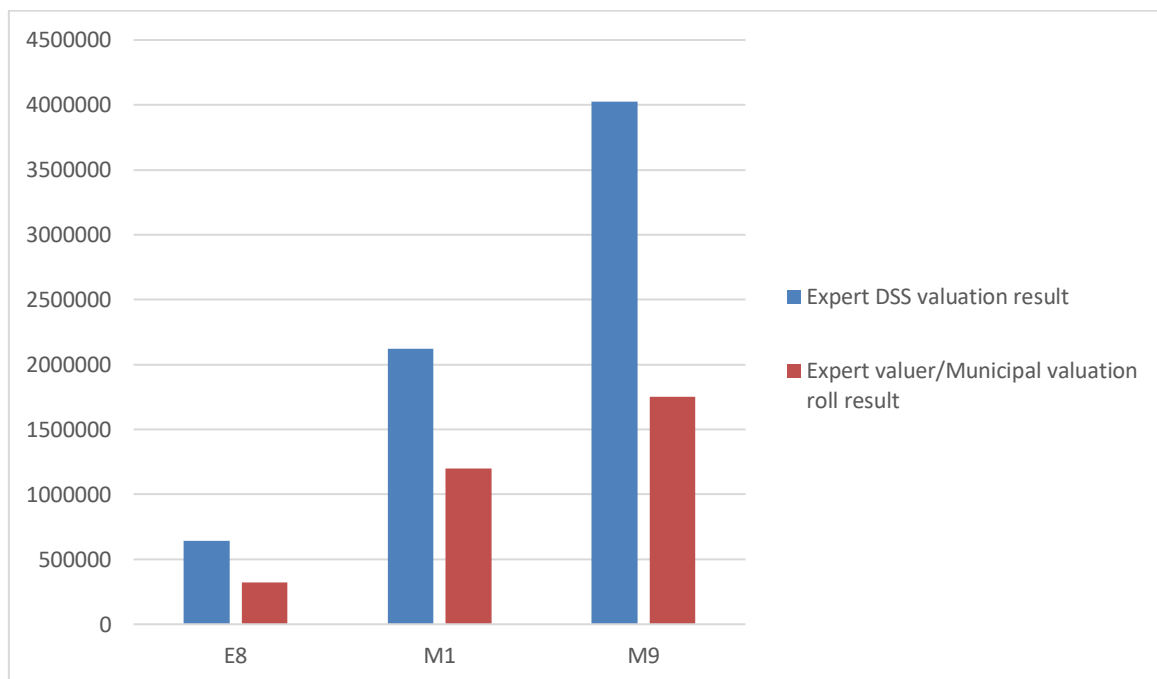


Figure 7.3: Expert DSS valuation vs. most recent sale value of an atypical property

Considering that the expert DSS is designed not to know what the actual value of the property under consideration is, it is considered to add an additional feature that will enable the DSS to remove atypical properties that adhere to specific criteria from a search database when constructing the search query.

The expert DSS successfully compensates for the difference in date of most recent valuation of a comparable property by using the consumer price index linked to their associated dates of sale or valuation. The relevant cases are shown in Table 7.4.

Tabel 7.4: Similarity indices of properties using comparable properties from different dates

Property number	Suburb	Date valuation	of Date of sale of most comparable property	Similarity index
E7	Neserhof	02/24/2014	05/23/2019	0.89
E3	Roosheuvel	06/04/2015	03/07/2019	0.96
E6	Meiringspark	01/14/2016	04/10/2018	0.74

With the identified results in mind from the knowledge verification phase, it is concluded that expert DSS provides valid property valuation results. The next step is to evaluate the system to determine whether the expert DSS achieved the research goals that were identified in the literature.

7.3 System evaluation

The objective of this study was to evaluate the integration of property valuation mathematical models and expert valuer knowledge into an automated valuation model as an expert DSS to provide valuable valuation results. This must be achieved while compensating for dynamic fluctuations in the South African environment and avoiding pattern recognition problems that are identified in other automated valuation models or systems (Bozic *et al.*, 2013; Gorbenko, 2019; Gnat, 2020), as identified in the literature.

The study focused on the residential properties. A DSS focuses on optimisation problems and has been implemented previously in other areas of property valuation like spatial planning (Christopoulou & Haklay, 2005), but not in residential valuations in the South African environment. Residential valuation relies on factual data that may not be manipulated to optimise the value of a property (Van der Vyfer, 2020; Wentzel, 2020), therefore, a DSS is not commonly implemented in residential valuations. The development of an expert DSS was considered ideal for applying the sales comparison method where a property value is determined by comparing its attributes to that of other properties (Theron, 2020; Wentzel, 2020). Therefore, the sales comparison property valuation method was used in this study.

An interactive computerised demonstrator, in the form of an expert property valuation DSS was developed. The knowledge engineering process was followed to develop the expert DSS by consulting expert valuers in the field and studying published documentation. This allowed the development of all the necessary models and components to be used in the expert DSS.

Upon property attribute input received from a user, the expert DSS would compile a database search query according to specific rules for this purpose in the knowledge base. A selection

of three comparable properties was acquired in an iterative approach. The property with the least number of attribute differences and most recent date of valuation was identified as the most comparable property and used to calculate the market value of a property under consideration. Structural and/or umbrella adjustments were made for the relevant attributes that differed from those of the subject property.

The validity of the expert DSS was confirmed by means of the formal tests discussed in this chapter. It was substantiated that the selected mathematical model together with the expert rules in the knowledge base is effectively utilised by the decision-making module to compile a search query and find a comparable property in the database to use for the valuation of a subject property. System validation also showed that valuations calculated by the expert DSS strongly correlates with valuations done by human experts using the sales comparison method and the municipal roll determined by the mass-appraisal technique.

The expert DSS accomplishes the goal of compensating for time differences relating to the most recent date of valuation of a comparable property by making relevant adjustments based on the consumer price index.

7.4 Recommendations

The goals of the expert DSS has been achieved, but additional features and recommendation have been identified during the course of this study that may be useful for further implementation. The following recommendations have currently been identified:

- Currently, the system is based on the residential property valuation section. The system can still be expanded towards different sections of property valuation, such as industrial properties, given proper models.
- The implementation of GIS software to determine the distance between the subject property and comparable properties is recommended. In order to apply the one-kilometre rule a large properties database will be required.
- The implementation to gather properties from external online sources. External sources, such as *Property24*, provide property data of many regions in South Africa. This type of data include property selling price and one or two property attributes. This can be a very useful addition if the source is willing to cooperate and share a larger dataset.
- Use a property attribute infrequency error term that has been calculated from the associated dataset of comparable properties to adjust the calculation of a new property value. This error term would be a description on how much the attributes

influence the value of a property. Given a large enough dataset, the expert DSS would be able to identify infrequent selected properties based on their value compared to attributes and adjust the estimated market value accordingly.

7.5 Research contributions

This study determined the validity of combining property valuation mathematical models and expert system and DSS principles into an automated valuation model in the form of an expert DSS in the South African residential property environment, while compensating for time difference and avoiding pattern recognition faults of mass-appraisal techniques. As a result, the following contributions can be considered for the fields of property valuation, decision support systems and expert systems:

- The study concluded that a decision support system can be used in the residential property environment of property valuations to provide feasible results.
- A knowledge base representing the selection and calculation methods of the process can be successfully used in conjunction with a decision support system to calculate a new property value.
- If property data is available, the expert DSS can be expanded towards different regions.
- The identified attributes that are used for comparable property selection in this study can be used to accurately calculate new property values.
- The expert DSS compensates for time differences in properties that, as identified in the literature, has been overlooked by most conventional automated valuation systems.
- The decision module selects the most optimal comparable properties for use in the calculation of a new property.

7.6 Chapter summary

Chapter 7 provided a summary of the results from the conducted formal tests and system evaluation phase. Section 7.2 discussed how the validity of the system was tested and included a summary of the results from the formal tests. This was followed by a brief evaluation of the expert DSS in Section 7.3 by looking back at certain objectives identified in the literature in, to be achieved. Afterwards a few recommendations towards future implementations were discussed in Section 7.4 and the chapter concluded with a few possible contributions to the fields that were relevant to the study in Section 7.5.

The next chapter concludes this study through a brief summary of what was achieved during the study. The goals that were identified at the start of the study are discussed, along with possible contributions to the field. A few encountered problems are also discussed.

CHAPTER 8: CONCLUSION

8.1 Introduction

Chapter 8 discusses concluding remarks on this study. The objectives are firstly evaluated and discussed on how they were achieved in Section 8.2. Next, a brief summary of a few problems that had been encountered during the study are discussed in Section 8.3. Afterwards, a number of new possible research topics will be discussed that had been identified during the course of the study in Section 8.4.

8.2 Objectives of the study

In Chapter 1 the goal of this study was stated as to incorporate expert system technology and mathematical modelling approaches into a property valuation decision support system (DSS) that can dynamically adapt to the evolving economic climate of the South African environment. The secondary objectives that had to be reached towards achieving this goal included:

- Conduct an in-depth literature study on property valuation, DSSs, current automated valuation or mass-appraisal techniques and valuation techniques.
- Investigate the use of mathematical modelling in property valuation and formulate a model relevant for the South African property environment.
- Perform the required stages of knowledge engineering in order to develop a theoretical layout of an expert DSS for property valuation in South Africa.
- Develop and design an interactive computerised demonstrator in the form of an expert DSS that implements the formulated model for the South African property environment.
- Verify and validate the developed expert DSS.
- Evaluate the system in terms of the criteria identified in literature and experts that were consulted.
- Reflect on the contributions to the South African property environment and recommend future research.

A summary of how these objectives were achieved and addressed follows.

Conduct an in-depth literature study on property valuation, decision support systems (DSSs), current automated valuation or mass-appraisal techniques and valuation techniques.

This objective was achieved in Chapter 2 which discussed the property valuation environment in detail by referring to relevant and recent literature. The sales comparison method was identified and discussed to be used in a DSS. Research on automated valuation approaches

in the sales comparison method, that is currently being used in the property valuation environment, was considered. The concept of a DSS was discussed, along with the relevant types that were considered for this study. Chapter 3 contains a functional layout of the expert DSS. The incorporation of a knowledge-driven element in combination with a data-driven DSS into an expert property valuation DSS was briefly discussed. The user interface, database, knowledge base, and decision-making module were described for implementation in the DSS. The composition of the components in the DSS motivated utilisation of the expert system software development life cycle for developing the expert DSS.

Investigate the use of mathematical modelling in property valuation and formulate a model relevant for the South African property environment.

Two different mathematical modelling techniques that are often used in property valuation and which are relevant to this study were discussed in Chapter 2. This was accompanied by brief examples of implementation of the sales comparison and mass-appraisal methods from previous research studies. Availability of residential property datasets motivated the implementation of the sales comparison method in the expert DSS developed in this study. Factors that needed to be considered in developing a mathematical model for the South African environment were discussed and incorporated. Implementation of the model in the expert DSS was discussed in more detail in Chapter 5.

Perform the required stages of knowledge engineering in order to develop a theoretical layout of an expert DSS for property valuation in South Africa.

In Chapter 4, the motivation for the use of the expert system development lifecycle was explained by discussing the involvement of expert valuers during system development. Consequently, the concept of expert systems was briefly described along with the related development lifecycle. This was followed by the planning, definition, analysis and extraction of the relevant knowledge resulting in a structural design for the expert DSS which was covered in chapter 5. Knowledge analysis comprised identification of the property attributes relevant in the sales comparison method construction of the rules for the expert DSS. The functional layout of the expert DSS resulted from knowledge extraction.

Develop and design an interactive computerised demonstrator in the form of an expert DSS that implements the formulated model for the South African property environment.

The development of the expert DSS was outlined in Chapter 6. This chapter started by discussing the software development tools that the expert DSS was built upon and how the expert DSS executes the valuation process in Section 6.2. This also included a flowchart,

depicting the flow of the expert DSS's task execution, from the start where the data is inputted into the system to where a new property value is calculated. Afterwards, the chapter continued by discussing the most important components of the expert DSS with the assistance of figures. The next section discusses how the property attributes are used to adjust the value of a property if needed. Finally, the chapter is concluded with an example valuation conducted through the expert DSS.

Verify and validate the developed expert DSS.

System validation was executed in Chapter 6 and verification in Chapter 7. System validation involved assessing the expert DSS's current functionality on how it executes the property valuation process through the sales comparison method. This was handled in Section 6.2.2. System verification involved testing the system's functionality against case data or historic valuations as described in Section 7.2. This involved identifying real valuations that were conducted by expert valuers and using the same data to execute the valuation process by using the expert DSS. Two different types of historic valuation datasets were used to broaden the expert DSS's validity towards how it handles property valuation automation. The types of valuation datasets consisted of valuations performed using the sales comparison method (valuations from expert valuers) and valuation performed through mass-appraisal (valuations performed by the municipal valuation roll). The expert DSS's produced valuation results showed positive outcomes towards both datasets, concluding that the system provides valid property valuation results.

Evaluate the system in terms of the criteria identified in literature and experts that were consulted.

The identified criteria expressed by the literature and by expert valuers, involved developing an expert DSS incorporating the sales comparison mathematical model to provide feasible valuation results in the South African environment, while compensating for the shortcomings of the most common mass-appraisal techniques, namely the dynamic South African economic environment and not using some form of pattern recognition in a dataset. To compensate for economic imbalance, the expert DSS uses a CPI value to adjust the value of a property according to differences in date of sale or valuation. The expert DSS also completely ignores pattern recognition because of how it executes the sales comparison method. Considering that pattern recognition in the case of property valuation attempts to identify how the attributes of a property influence the value of a property, the expert DSS does not incorporate this at all. It merely searches for comparable properties of the most comparable attributes to the subject property as possible, regardless of how much the attributes influence the value of a property.

Finally, according to the positive results from the formal tests, the expert DSS is proven to be valid for residential valuations in the South African environment, which was discussed in Chapter 7.

Reflect on the contributions to the South African property environment and recommend future research.

This final research goal was achieved in Chapter 7, where possible recommendations for the system were identified in Section 7.4, followed by a brief summary in Section 7.5 on how the expert DSS contributes to the relevant fields associated with this study.

To summarise, all objectives as set forth were achieved. Based on the results presented in Chapter 7, it was concluded that:

- It is feasible to combine expert system techniques with property valuation mathematical modelling techniques into an expert DSS that can execute the valuation process, while compensating for dynamic fluctuations in the South African property valuation environment.
- The expert DSS selected the most optimal comparable properties from the database.
- The expert DSS successfully adjusted attributes that were different from the comparable property.
- The expert DSS meets the requirements set out by the expert valuers by executing the sales comparison method properly.
- The expert DSS correctly compensated for market imbalance by adjusting the final value of the property with the index value, if needed.
- The expert DSS performed well against historic valuations in the formal testing phase of the development lifecycle. Using two datasets that contained calculated property values from two different property valuation methods, it is concluded that the system performs very well as an automated valuation method.

8.3 Problems experienced

Property attributes provide a detailed description of a property, as well as a good indication towards its value. Having a small number of available property attributes, an accurate estimated property value cannot be derived. The more available attributes there are describing a property, the more accurately an expert valuer can estimate the value of a property. The same can be said for automated valuation models and systems.

On the other hand, the data that the system used for comparable property selection was, and still is difficult to obtain. Very few external sources provide in-depth property descriptions of attributes and mostly require a physical inspection of the property to identify the appropriate attributes. This may not have been a problem if external sources provided better attribute descriptions. The expert DSS is very reliant on a thorough description of property attributes and if external sources do not provide a thorough description, comparable properties must be physically inspected. Luckily, expert valuers were able to provide enough comparable properties for the expert DSS's property database for testing purposes.

8.4 Further research

The expert DSS was developed to optimise the process of residential property valuation in the fluctuating South African environment. Although it executes the sales comparison method feasibly for residential properties, it can be expanded towards different sections of property valuation such as industrial properties, given enough methods and models of valuation.

Another topic for further research is to expand the expert DSS's selection capabilities using GIS technology. Expert valuers tend to limit their search to properties located within a one-kilometre radius of the property under consideration. Given a much larger available dataset of comparable properties, the expert DSS can increase the sales comparison complexity even further with GIS technology by searching for properties within a one-kilometre radius of the subject property. The expert DSS currently prioritizes properties from the same suburb, but suburbs can be much larger than a one-kilometre radius.

The implementation of property selection from an external source instead of a local property database is also being identified. As discussed in the previous section, online sources such as *Property24* do not provide a thorough description of a property's attributes but are dynamically updated. This type of information from different sources might also not be completely available to the public free of charge or at an affordable rate. If property data with enough attributes for a proper valuation were readily available, it can be implemented as another dynamic feature of the system.

The expert DSS can also benefit from a property infrequency error term to adjust the value of a property, based on how property attributes influence the property value. This is described as an adjustment value that is derived from the property database as an indication towards how the property attributes accurately describe the value of a property. If the attributes show a proper influence on the value of a property, then this error term value would be close to zero. However, this adjustment value would increase if the attributes indicated no proper description

on how they influence the value of a property. Currently, the expert DSS selects comparable properties based on how closely relatable they are to the subject property, using property attributes. This is to avoid inaccurately estimated values if the property attributes barely have any effect on the value of a property. A few property value infrequencies were noted during testing, due to infrequent property values, so a property infrequency error term can assist to more accurately adjust the value of a property.

8.5 Conclusion

Chapter 8 is the final chapter of this mini-dissertation. A summary of the goal of the study and the secondary objectives were briefly outlined, followed by how they were achieved and addressed. In conclusion, the problems experienced, and possible future research opportunities were outlined.

BIBLIOGRAPHY

- Abidoye, R. B., & Chan, A. P. C. (2016). A survey of property valuation approaches in Nigeria. *Property Management*, 34(5), 364-380.
- Amidu, A.-R., & Boyd, D. (2018). Expert problem solving practice in commercial property valuation: an exploratory study. *Journal of Property Investment and Finance*, 36(4), 366-382.
- Amidu, A., Boyd, D., & Agboola, A. (2019). The role of knowledge in valuation practice: expert valuers' perceptions. *Journal of Property Investment and Finance*, 37(6), 524-540.
- Angelos, M., Antonis, R., & Marianthi, S. (2013). Property valuation with artificial neural network: the case of Athens. *Journal of Property Research*, 30(2), 128-143.
- Anon. (2018, August 13). *What you should know about property valuation report* [Blog post]. <https://www.valuationsnsw.com.au/blog/things-you-should-know-about-property-valuation-report/>
- Anuar, A., & Nor Hana Asykin, N. H. (2010). Comparison method - preference of adjustment techniques among valuers. *Journal of Design and Built Environment*, 7(1), 33-42.
- Arcuri, N., De Ruggiero, M., Salvo, F., & Zinno, R. (2020). Automated valuation methods through the cost approach in a BIM and GIS integration framework for smart city appraisals. *Sustainability*, 12(18), 7546. doi: 10.3390/su12187546
- Aziz, K. A., H. Harris & N. A. Ab. Aziz, 2012. Intellectual property valuation decision support system for university research output: a conceptual model. *Communications of the IBIMA*, 2012(1), 1-17.
- Babtan, S.I. 2020. *Automated Valuation Model for Residential Real Estates* (No. 2780). EasyChair.
- Bencure, J. C., Tripathi, N. K., Miyazaki, H., Ninsawat, S., & Kim, S. M. (2019). Development of an Innovative Land Valuation Model (iLVM) for mass-appraisal application in sub-urban areas using AHP: an integration of theoretical and practical approaches. *Sustainability*, 11(13), 3731.
- Birkeland, K., & D'Silva, A. D. (2018). *Developing and evaluating an automated valuation model for residential real estate in Oslo*. (Unpublished master's dissertation). NTNU, Norway.
- Boyle, C. (1984). An expert system for valuation of residential properties. *Journal of Valuation*, 2(3), 1-15.
- Bozic, B., Milicevic, D., Pejic, M., Marosan, S. (2013). The use of multiple linear regression in property valuation. *Geonauka*, 1(1), 41-45.

- Brümmer, G. (2019). *AECOM Africa Property & Construction Cost Guide 2019/20* (30th ed.). AECOM SA.
- Butsic, V., Hanak, E., & Valletta, R. G. (2011). Climate change and housing prices: hedonic estimates for ski resorts in western north America. *Land Economics*, 87(1), 75-91.
- Christopoulou, K., & Haklay, M. (2005). A Spatial Decision Support System for Property Valuation. In *GIS Research UK 2005* (Proceedings). Glasgow, UK.
- Coronel, C., Morris, S., Rob, P., & Crocket, K. (2013). *Database principles: fundamentals of design, implementation and management* (2nd ed.). Andrew Ashwin.
- d'Amato, M., & Kauko, T. (2017). Advances in automated valuation modeling. *Springer International Publishing AG*, 10(1), 978-983.
- Deitel, P., & Deitel, H. (2012). *Java, how to program* (9th ed.). Pearson Education Limited.
- Doszyń, M. (2020). Econometric support of a mass valuation process. *Folia Oeconomica Stetinensia*, 20(1), 81-94.
- Ellenberger, E. L., Smal, J. D. C., & Margolius, J. L. (2010). *The valuer's manual*. (3rd ed). LexisNexis Butterworths.
- Ellerman, D. (2004). The mathematics of real estate appraisal. *Social Science Research Network Journal*, 1(1), 1-43.
- Ellis, T. R. (2011). Sales comparison valuation of development and operating stage mineral properties. SME Annual Meeting Feb. 27-Mar. 02, 2011, Denver, CO.
- Epley, D. (2017). The need to reference automatic valuation models to the valuation process. *Journal of Real Estate Literature*, 25(1), 237-251.
- Estate Marks v Pretoria City Council*, 3 SA 227 (A) 253 H-254 (B) (1969).
- Evans, K. M. Lausberg, C., & How, J. S. S. (2019). Reducing the property appraisal bias with decision support systems: an experimental investigation in the South African property market. *Journal of African Real Estate Research*, 4(1), 108-138.
- Farkas, A., & Porumb, B. (2020). A multi-attribute sales comparison method for real estate valuation. *Periodica Polytechnica Social and Management Sciences*, 28(1), 1-11.
- Fernando, J. 2020. *Correlation Coefficient*. Investopedia. Retrieved August 12, 2020, from <https://www.investopedia.com/terms/c/correlationcoefficient.asp>
- French, N., & Gabrielli, L. 2018. Pricing to market property valuation revisited: the heirarchy of valuation approaches, methods and models. *Journal of Property Investment and Finance*, 36(4), 391-396.
- Gallin, J., Molloy, R., Nielsen, E. R., Smith, P. A., & Sommer, K. (2018). Measuring aggregate housing wealth: New insights from an automated valuation model. *Finance and Economics Discussion Series 2018-064*. Washington: Board of Governors of the Federal Reserve System, <https://doi.org/10.17016/FEDS.2018.064>.

- Giarratano, J. C., & Riley, G. D. (2005). *Expert systems: principles and programming* (4th ed.). Thomson Course Technology.
- GISGeography. (2020). *What is Geographic Information Systems (GIS)?* GISGeography. Retrieved on April 27, 2020, from <https://gisgeography.com/what-gis-geographic-information-systems/>
- Gnat, S. (2020). Impact of the regularization of regression models on the results of the mass valuation of real estate. *Folia Oeconomica Stetinensia*, 20(1), 163-174
- Gorbenko, A. S. (2019). How do valuations impact outcomes of asset sales with heterogeneous bidders? *Journal of Financial Economics*, 131(1), 88-117.
- Heller, M. (2019). *What is SQL? The lingua franca of data analysis*. <https://www.infoworld.com/article/3219795/what-is-sql-the-lingua-franca-of-data-analysis.html>
- Hendrikse, J. W. (2015). *Valuations handbook*. Genesis Corporate Services.
- IAAO. (2003). Standard on automated valuation models (AVMs). *Assessment Journal*, 10(4), 109–154.
- Kilpatrick, J. (2011). Expert systems and mass-appraisal. *Journal of Property Investment and Finance*, 29(4/5), 529-550.
- Kisilevich, S., Keim, D., & Rokach, L. (2013). A GIS-based decision support system for hotel room rate estimation and temporal price prediction: The hotel brokers' context. *Decision Support Systems*, 54(2), 1119-1133.
- Kokot, S., & Gnat, S. (2019). Identification and classification of real estate features for the purpose of an algorithm-based valuation – case study in Szczecin. *Folia Oeconomica Stetinensia*, 19(2), 38-55. doi: 10.2478/fofi-2019-0012.
- Koller, D., Lindstrom, P., Ribarsky, W., Hodges, L. F., Faust, N., & Turner, G. (1995). *Virtual GIS: a real-time 3D geographic information system*. College of Computing and Graphics, Visualization & Usability Center Georgia Institute of Technology, Georgia.
- Krukeja, S. (2013). *Components of Decision Support Systems (DSS)*. HGManagement. <https://www.managementstudyhq.com/components-of-decision-support-systems.html#:~:text=A%20decision%20support%20systems%20consists,mined%20form%20the%20Internet%2C%20etc>
- Landry, M. (2019, September 11). *Key Benefits of Microsoft Access Database*. [Blog post]. <https://training.nhlearninggroup.com/blog/key-benefits-of-the-microsoft-access-database>.
- Lausberg, C., & Dust, A. (2017). Reducing the Appraisal Bias in Manual Valuations with Decision Support Systems. *Advances in Automated Valuation Modeling*, 86(1), 331-343

- Liao, S-H. (2005). Expert system methodologies and applications - a decade review from 1995 to 2004. *Expert Systems with Applications*, 28(1), 93-103.
- Lott, M. (2020). *What is a valuation report?* Compare my move. Retrieved 18 March, 2020, from <https://www.comparemymove.com/advice/surveying/valuation-survey>
- Lowies, G. A., Hall, J. H., & Cloete, C. E. (2016). Heuristic-driven bias in property investment decision-making in South Africa. *Journal of Property Investment & Finance*, 34(1), 51-67.
- Maylawati, D. S., Darmalaksana, W., & Ramdhani, M. A. (2018). Systematic design of expert system using unified modelling language. *IOP Conference Series: Materials Science and Engineering*, 288(1), 012047
- Metzner, S., & Kindt, A. (2018). Determination of the parameters of automated valuation models for the hedonic property valuation of residential properties. *International Journal of Housing Markets and Analysis*, 11(1), 73-100.
- Michaletz, V. B., & Artemenkov, A. (2018). The transactional assets pricing approach and income capitilization models in professional valuation: towards a "quick" income capitalization format. *Real Estate Management and Valuation*, 26(1), 89-107.
- Mills, A. C. (2019). Residential perspective on data collection and property description in the valuation process. *The Appraisal Journal*, 87(1), 42-53.
- Miotti, L. A., & Loch, C. (2020). Property value map updating by mass appraisal method—a case in the city of Pato branco, state Paraná. *Acta Scientiarum. Technology*. doi: 10.4025/actascitechnol.v43i1.48912.
- Mohale, M., Geyer, H. S. Jr., & Geyer, H. S. (2016). Undervaluation of real estate properties in disadvantaged areas in the city of Johannesburg. *South African Journal of Economics*, 84(2), 217-229.
- Moody's Analytics. (2020). *South Africa - Consumer Price Index (CPI)*. Moody's. Retrieved July 28, 2020 from <https://www.economy.com/south-africa/consumer-price-index-cpi>
- Montgomery, D. C., & Peck, E. A. (1992). *Introduction to linear regression analysis* (2nd ed.). Wiley.
- Minister of Water Affairs v Mostert*, 4 SA 890 (A) (1966).
- Nejad, M. Z., Lu, J., & Behbood, V. (2007). Applying dynamic Bayesian tree in property sales price estimation. 12th International Conference on Intelligent Systems and Knowledge Engineering. University Technology Sydney, Australia.
- Oates, B. J. (2006). *Researching information systems and computing*. SAGE Publications Ltd.
- Okoruwa, A. A. (2017). *Regression analysis* (14th ed.). Appraisal Institute.

- Perktold, J., Seabold, S., & Taylor, J. (2020). *Statsmodels*. Retrieved July 12, 2020, from <https://www.statsmodels.org/stable/index.html>
- Pinard, S., & Quasney, J. (2013). *Introduction to programming*. Cengage Learning EMEA.
- Popescu, D., Cerna Mladin, E., Boazu, R., & Bienert, S. (2009). Methodology for real estate appraisal of green value. *Environmental Engineering and Management Journal*, 8(3), 601-606.
- Power, D. J. (2002). *Decision support systems: concepts and resources for managers*. Quorum Books.
- Property24. (2020). *Property for Sale in Klerksdorp*. Retrieved July 20, 2020, from <https://www.property24.com/for-sale/klerksdorp/north-west/128>
- Rahim, F. N. F., & Razali, M. F. (2018). Comparing the performance of similarity measures in sales comparison method. Proceedings of the ASEAN Post Graduate Conference (APGC) 2018, University of Malaya, Kuala Lumpur, 15 November 2018.
- Randolph, J. J. (2009). A guide to writing the dissertation literature review. *Practical Assessment, Research & Evaluation*, 14(13), 1-13.
- Rehman, A. A., & Alharthi, K. (2016). An introduction to research paradigms. *International Journal of Educational Investigations*, 3(8), 51-59.
- Saunders, M., Lewis, P., & Thornhill, A. (2015). *Research Methods for Business Students*. (7th ed.). Pearson.
- Segal, T. (2019). *Decision support system-DSS*. Investopedia. Retrieved September 6, 2019, from <https://www.investopedia.com/terms/d/decision-support-system.asp>
- Selim, S. (2011). Determinants of house prices in turkey: a hedonic regression model. *Doğuş Üniversitesi Dergisi*, 9(1), 65-76.
- Sevgen, S. C., & Aliefendioğlu, Y. (2020). Mass appraisal with a machine learning algorithm: random forest regression. *Bilişim Teknolojileri Dergisi*, 13(3), 301-311.
- Shapiro, E., Mackmin, D., & Sams, G. (2019). *Modern methods of valuation*. (12th ed). London: Routledge.
- Sue, C. (2017). *GIS (Geographic Information System)*. Retrieved September 30, 2020, from <https://www.nationalgeographic.org/encyclopedia/geographic-information-system-gis/>
- Surber, K. 2014. *What is Inference? - How to Infer Intended Meaning*. Retrieved August 31, 2020, from <https://study.com/academy/lesson/what-is-inference-how-to-infer-intended-meaning.html>
- Swanepoel, J. W. H., Swanepoel, C. J., Van Graan, F. C., Allison, J. S., & Santana, L. (2011). *Elementêre statistiese metodes* (4de Uitg.). Noordwes-Universiteit.
- Syful Hoque, M. (2019). Addressing property taxation challenges for developing countries: an application of a spatially based, weighted parametric valuation model. *Journal of Property Tax Assessment & Administration*, 16(2), 15-41.

- Taylor, B. W. (2016). *Introduction to management sciences* (12th ed.). Pearson Education Limited.
- Theron, N. (2020). Decision support systems in property valuation for South Africa [Questionnaire]. 16 Jul., Klerksdorp.
- Tsu-Chin, L., & Ying-Chih, L. (2010). Further empirical examination of an improved sales comparison approach. *Nordic Journal of Surveying and Real Estate Research*, 7(2), 51-66.
- Van der Vyfer, C. (2020). Decision support systems in property valuation for South Africa [Questionnaire]. 21 Jul., Klerksdorp.
- Vaughan, J. (2017). *Data*. Retrieved August 4, 2020 from <https://searchdatamanagement.techtarget.com/definition/data>
- Vhutshilo, M. (2019). *Unbounded rule-based expert system for selecting software development methodologies*. (Dissertation – MSc.). University of Venda.
- Wagner, B. (2021). *A tour of the C# language*. Retrieved January 30, 2021, from <https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/>
- Wentzel, A. (2020). Decision support systems in property valuation for South Africa [Personal interview]. 29 Jul., Klerksdorp.
- Whitten, J. L., & Bentley, L. D. (2007). *System analysis and design for the global enterprise* (7th ed.). McGraw-Hill/Irwin
- Wier, L. 2020. Tax-motivated transfer mispricing in South Africa: Direct evidence using transaction data. *Journal of Public Economics*. 148(1): 1-16.
- Windeed, 2020. *Deeds office property search*. Retrieved July 24, 2020, from <https://search.windeed.co.za/DeedsOffice/Property?ReturnUrl=%2f>
- Yildirim, H. (2019). Property value assessment using artificial neural networks, hedonic regression and nearest-neighbors regression methods. *Selçuk University Journal of Engineering, Science and Technology*, 7(2), 387-404.
- Zeng, T. Q., & Zhou, Q. M. (2001). Optimal spatial decision-making using GIS: A prototype of a real estate geographical information system (REGIS). *International Journal of Geographic and Information Sciences*, 15(1), 307-321.
- Zhu, X., Healey, R. G., & Aspinall, R. J. (1998). A Knowledge-based systems approach to design of spatial decision support systems for environmental management. *Journal of environmental Management*, 22(1), 35-48.
- Zyga, J. (2018). Object and objective of property appraisal and their effects on valuation methods and databases. *GEODETSKI VESTNIK*, 63(1), 92-103.

APPENDIX A: PROPERTY VALUATION INPUT DOCUMENT FOR THE EXPERT VALUERS

Valuation results of the expert DSS in comparison with private valuations performed in the past 12 months in the Klerksdorp area, by expert valuers:

Expert valuer estimated value: R1,235,000.00

Expert DSS resulting value: R1,124,812.00

PROPERTY ATTRIBUTES:

Property number: E1

Effective date of valuation: 23 May 2019

Suburb: Elandsheuwel

Property type: Residential

Flat/garden flat: 2 Bedroom flat

Garage/carport: More than 2/ Garage and a carport

No of bedrooms: 3

No of bathrooms: 2

Domestic quarters description: Standard

Amenities eg swimming pool: Pool

Condition: Good

APPENDIX B: PROPERTY VALUATION INPUT PAGE FOR THE MUNICIPAL VALUERS ROLL

RESULTS OF COMPARISON WITH MUNICIPAL VALUATION ROLL OF CITY OF MATLOSANA (KLERKSDORP)

PROPERTY NUMBER:	M1
MUNICIPAL VALUE:	R1,200,000.00
SUBURB:	Flamwood
FLAT DESCRIPTION:	Single bedroom
GARAGE/CARPORT:	More than 2/ garage and a carport
NO BEDROOMS:	3
NO BATHROOMS:	2
DOMESTIC QUARTERS:	STANDARD: X EXTRA:
AMENITIES:	Pool
CONDITION:	Good
EXPERT DSS RESULT:	R2,119,684.00

Table B.1: Excerpt of a single property valuation, completed by the municipal valuer's roll in the city of Matlosana (Klerksdorp)

APPENDIX C: RESULTS OF HISTORIC VALUATION TESTS

Expert valuer's property valuation results

	E1	E2	E3	E4	E5
Market value (Expert valuer)	R1,235,000.00	R2,200,000.00	R500,000	R1,350,000.00	R1,300,000.00
Market Value (Expert DSS)	R1,124,812.00	R2,175,225.00	R521,956.10	R1,116,243.00	R1,241,297.00
Suburb	Elandsheuwel	Wilkoppies	Roosheuwel	La Hoff	Doringkruin
Date of valuation	23 May 2019	18 May 2020	4 June 2015	6 March 2020	6 January 2020
Flat description	2 Bedroom	None	None	Single bedroom	None
Garage/carpport	More than 2	More than 2	Double	More than 2	Double
Number of bedrooms	3	4	3	4	3
Number of bathrooms	2	3	2	3	2
Domestic quarters description	Standard	Standard	Standard	Standard	Standard
Amenities	Pool	None	None	Pool	Pool
Condition	Good	Good	Good	Good	Good

Table C.1: First set of property attributes and valuation results of the expert valuers, compared to the expert DSS

E6	E7	E8	E9	E10	E11
R600,000.00	R710,000.00	R320,000.00	R1,450,000.00	R1,200,000.00	R460,000.00
R812,986.90	R633,482.00	R616,049.20	R1,400,000.00	R1,000,000.00	R472,500.00
Meiringspark	Neserhof	Adamayview	Flamwood	Flimieda	Ellaton
14 January 2016	24 February 2014	12 January 2019	2 July 2020	28 July 2019	14 August 2018
None	None	None	2 Bedroom	Single bedroom	None
Double	Double	Double	More than 2	Single	More than 2
3	4	3	4	3	3
2	3	2	3	2	2
Standard	Standard	Standard	Standard	Standard	Standard
Pool	Pool	None	None	None	Pool
Mid	Good	Good	Good	Good	Good

Table C.2: Second set of property attributes and valuation results of the expert valuers, compared to the expert DSS

Municipal valuer's roll property valuation results (all properties were valued 31 January 2020)

	M1	M2	M3	M4	M5
Market value (Municipal valuer's roll)	R1,200,000.00	R620,000.00	R1,060,000.00	R850,000.00	R720,000.00
Market Value (Expert DSS)	R2,119,684.00	R644,028.80	R1,300,879.00	R860,154.40	R633,888.90
Suburb	Flamwood	Roosheuvel	Doringkruin	Flimieda	Meiringspark
Flat description	Single bedroom	None	None	Single bedroom	Single bedroom
Garage/carport	More than 2	Carport	More than 2	Double	Carport
Number of bedrooms	3	3	3	3	3
Number of bathrooms	2	2	2	2	2
Domestic quarters description	Standard	Standard	Standard	Standard	Standard
Amenities	Pool	Pool	Built in bar/playroom	None	Pool
Condition	Good	Mid	Good	Good	Good

Table C.3: First set of property attributes and valuation results of the municipal valuer's roll, compared to the expert DSS

M6	M7	M8	M9	M10	M11
R620,000.00	R640,000.00	R720,000.00	R1,750,000.00	R780,000.00	R780,000.00
R608,600.20	R492,001.00	R847,329.00	R4,024,692.00	R789,073.00	R714,428.70
Adamayview	Ellaton	Ellandsheuvel	Wilkoppies	Neserhof	La Hoff
None	None	2 Bedroom	Single bedroom	None	None
Single	Single	Carport	More than 2	More than 2	Single
3	3	3	4	3	3
2	2	2	3	2	2
Standard	Standard	Standard	Standard	Standard	Standard
None	None	None	Pool	None	None
Good	Good	Good	Good	Good	Good

Table C.4: Second set of property attributes and valuation results of the municipal valuer's roll, compared to the expert DSS

APPENDIX D: EXPERT VALUER QUESTIONNAIRE FEEDBACK

Decision support system

A decision support system is a computer software solution that assists businesses with making decisions where the problem domain might not be described very well. These decisions are usually related to what must be done to get the best outcome.

Examples of this include when:

- A business wants to calculate the optimal amount of a product to produce to maximize profit.
- A school wants to calculate the optimal diet boundaries of a meal while minimizing costs.
- A farmer wants to calculate when to plant his produce to maximize profit and output.

My study

The original idea of my study is to develop a decision support system (DSS) for the property valuation environment. My title was registered, however, for the valuation of existing properties, not land management or township development.

The problem I encountered, was that by valuing existing (residential) properties the DSS cannot be used to determine the value of a property. By analyzing the methods of pure property valuation, almost all of the information is facts, based on the subject property and may not be changed for a better outcome at that specific date and time.

I did however find a different method for incorporating the DSS in the valuation environment, and that is through the sales comparison method. In this method, the DSS can calculate the optimal set of properties to select from a database, while minimizing the differences between the subject property and comparable properties.

Questions and answers (Nathen Theron):

I require the following information:

1. According to what I described a DSS does, do you agree with my conclusion, stating that a DSS cannot be used to improve the value of a property?

I think that a DSS can improve the accuracy through which a Property Value is derived. But, I will agree with you that it cannot be used as a tool to improve a value. In other words, it can be used as a tool to enhance research and decision-making on the part of the valuer, and in some case we will see an increase in value, but then the opposite will also be true. The DSS will therefore play an important role in understanding the attributes which add or detract value.

2. Apart from land management and my own solution, where do you think can I incorporate a decision support system in the property valuation environment?

The "Prop Tech" industry in South Africa is booming. Your idea has the potential to leverage the DSS and valuation theory through a broad spectrum of uses within Property Valuation.

Imagine your DSS serves a “thought tool” in a new valuation software, which cuts the valuers time spent on valuations by half. Now imagine for instance that Listed Fund Portfolios load their properties onto your DSS and they never need to pay valuation fees again (which they normally pay every year) because now your DSS can continuously update the property’s performance and model this in terms of what the value is doing.

Questions and answers (Chris van der Vyfer):

1. According to what I described a DSS does, do you agree with my conclusion, stating that a DSS cannot be used to improve the value of a property? **YES! – A DSS cannot be utilized to improve the value of a property but it could be utilized to identify a value range in which the value of the property should fall. REMEMBER all freestanding properties are unique in that they may have the same basic design but they could differ substantially in terms of the type and quality of finishes offered and by the levels of maintenance and by the type and quality of additional improvements such as boreholes, tennis courts, entertainment areas, swimming pools etc. It is thus my opinion that although a reasonable value rate could be indicated for older properties that it could not determine the actual sales price or even predict the optimum value of a specific property.**

The system could however be better applied to sectional title units and especially in newer developments where all units offer similar type and quality accommodation and where all units have a similar level of maintenance. **Micro locality of the unit within the complex could only have a nominal impact and variance on value but one should still be able to reflect a fairly accurate value based on the DSS. The system could however be less accurate in older developments where the interior quality and condition of the units could vary considerably.**

2. Apart from land management and my own solution, where do you think can I incorporate a decision support system in the property valuation environment? **It is my opinion that DSS has a place in the property valuation industry overall but mainly when valuing residential properties. It is further my opinion that these systems could merely be utilized as additional tools (there are various other tools also available in the market) to determine the market range but that more realistic property values could only be indicated by humans as one would determine the value of the property based on a combination of the information indicated by the available tools and physical inspection. DSS systems could be accurate to an extent but it will never replace a “physical valuer” as systems could not physically see or factor in the impact of defects such as the impact of damp, subsidence, crack and maintenance levels of a property. DSS also creates the opportunity for larger and listed companies to “value” their own properties. – Listed companies already use all types of DSS software to predict values based on their cash flows and this could be another tool that could be mis-used if not properly understood as these types of models do not take physical, or makro- or micro environmental changes into consideration.**

Questions and answers (Alet Wentzel):

I refer in this regard to your study parameters regarding a DSS Solution for Valuers.

I am of the opinion that the DSS is a valuable tool for the valuer to assist him to arrive at the correct market value of the property.

In a valuation report, the valuer uses certain factors and discards some to determine an objective and sound market value of the property. There is however a subjective aspect to this process as the valuer must indicate the logical manner in which he approached the valuation and also why he decided to use certain factors and why he discarded some.

My answer to your queries are:

1. According to what I described a DSS does, do you agree with my conclusion, stating that a DSS cannot be used to improve the value of a property?

Answer: that is indeed correct. The DSS cannot improve a value, it can however assist the valuer when he has to use the comparable sales method to determine the best comparable properties and therefore obtain the correct market value of the property.

2. Apart from land management and my own solution, where do you think can I incorporate a decision support system in the property valuation environment?

The most obvious use will be with residential dwellings where the International Valuation Standards Council (IVS) determines that the comparable sales method is the acceptable method of determining value. The South African Council for the Valuers Profession also ascribes to these standards and are part of the IVS. The DSS can however also be used by Estate Agents for research when assisting new buyers and also if such a website is developed, any interested party from over the country can peruse the various suburbs and information to assist with property investment or relocating to that area.

Questions and answers (Grant Pohl):

1. According to what I described a DSS does, do you agree with my conclusion, stating that a DSS cannot be used to improve the value of a property?

I agree with your conclusion. This will be very useful tool to optimize a valuers decision on what comparables to use, as in some instances they will be a multiple amount of comparables for a valuer to compare to the subject property. The comparables the valuer decides to use is subjective based on his/her opinion on the most similar properties. This system will be useful in determining the most applicable comparables.

The DSS cannot improve the value of a subject property, but will reduce the margin of error when determining a value.

2. Apart from land management and my own solution, where do you think can I incorporate a decision support system in the property valuation environment?

The area of the subject property. Often when doing a valuation, there is limited information regarding the area of the property, particularly in small towns. The demographics, ages, municipality service, suburb LSM and general information on the area can be difficult to find. By adding this to your model, you will greatly assist valuers and estate agents. For example, in Johannesburg you have a suburb called Robin Hills. The suburb directly next to it is Windsor East. Robin Hills cater for high income earners, while across the road in Windsor East it is very low income earners, occupied by mostly illegal foreigners. A 4 bedroom house in Robin hills costs between R2,5 – R4,5mil. In Windsor east, a 4 bedroom house costs roughly R800,000 just a street away and in some cases across the road.

Questions and answers (Tersia Prinsloo):

1. According to what I described a DSS does, do you agree with my conclusion, stating that a DSS cannot be used to improve the value of a property?

I agree with your conclusion that a DSS cannot be used to improve the value of a property, as the value is determined by the valuer based on the use of certain factors in an effort to determine an objective and sound market value of the property. It will however assist the valuer when determining the best comparable properties to determine the market value calculation.

2. Apart from land management and my own solution, where do you think can I incorporate a decision support system in the property valuation environment?

The DSS can be used by Valuers, as well as Estate Agents when they are assisting parties in determining a suitable market value before listing properties as well as giving potential buyers assistance in determining market value in specific suburbs or areas. Interested parties can also access this information, depending where such information is made available, to assist in their decision- making process.

APPENDIX E: EXPERT DSS PROPERTY DATABASE

Properties table:

ID	Suburb id	Property price	Date of sale	Index id	Property size	Flat id	Property garage id	Property bedrooms number	Property bathroom number	Out building id	Amenity id	Property condition id
8	6	500000	02/27/2011	14	1230	1	5	3	2	1	1	1
11	10	300000	03/04/2011	15	1581	1	3	3	2	2	1	2
101	9	430000	04/15/2013	40	1605	1	2	3	2	1	2	2
34	11	725000	02/23/2015	62	1745	1	1	3	2	2	2	1
33	7	110000	03/03/2015	63	1628	1	1	3	2	2	1	3
35	7	550000	03/06/2015	63	1945	2	3	3	2	2	2	2
32	11	650000	03/19/2015	63	1665	1	3	4	2	2	2	2
36	11	648000	05/12/2015	65	1368	2	3	3	2	2	2	2
31	11	1047000	07/01/2015	67	1340	2	3	4	2	2	2	1
37	11	558279	07/24/2015	67	1457	2	3	3	2	2	2	3
38	11	790000	09/04/2015	69	1506	2	3	3	2	2	2	1
40	11	702000	11/12/2015	71	1839	2	3	3	2	2	2	1
39	11	825000	11/19/2015	71	1289	1	1	3	2	2	2	1
48	11	101000	12/04/2015	72	1478	1	1	0	0	2	1	3
45	11	1300000	12/08/2015	72	2191	2	4	4	3	1	2	1
21	5	1600000	01/12/2016	73	1348	2	4	3	2	1	2	2
18	9	850000	02/06/2016	74	4653	1	1	0	0	2	1	3
41	11	600000	03/31/2016	75	1745	2	3	3	2	2	1	2
43	11	260000	04/08/2016	76	1401	1	1	2	1	2	1	3
44	11	788000	04/19/2016	76	465	1	1	4	3	2	1	1
42	11	270000	06/15/2016	78	1401	1	1	2	1	2	1	3
46	11	400000	07/19/2016	79	1354	2	3	3	2	1	2	3
47	11	780000	09/19/2016	81	1782	2	3	3	2	1	2	3
49	11	550000	10/05/2016	82	1693	2	3	3	2	2	2	3
22	5	1140000	10/09/2016	82	1662	1	4	3	2	1	2	1
30	11	775000	11/14/2016	83	1448	2	3	3	2	2	2	3
23	5	1160000	11/29/2016	83	1449	1	4	3	2	1	2	1

60	7	480000	04/24/2017	88	1487	2	2	3	2	2	1	2
19	3	800000	05/04/2017	89	1777	2	4	3	2	1	3	2
89	8	450000	05/05/2017	89	971	2	2	3	2	2	1	2
88	8	450000	05/17/2017	89	1805	2	4	3	2	2	1	2
87	8	410000	07/04/2017	91	972	1	2	3	2	1	1	1
16	1	610000	08/22/2017	92	278	1	5	3	2	2	1	3
90	8	310917	09/01/2017	93	1026	1	4	2	2	2	1	2
17	1	840000	10/05/2017	94	2974	1	5	3	2	2	2	3
27	7	610000	12/15/2017	96	1487	1	3	4	3	2	1	1
28	7	1030000	01/18/2018	97	1487	2	3	3	2	1	2	1
13	1	50000	02/16/2018	98	443	1	1	0	0	2	1	1
15	1	690000	03/08/2018	99	1125	1	5	3	2	2	1	3
29	7	935000	04/10/2018	100	1582	1	3	3	2	1	2	1
107	1	2500000	04/16/2018	100	642	1	4	3	2	1	2	1
108	1	2080000	04/23/2018	100	575	1	4	3	2	1	1	1
4	7	620000	06/13/2018	102	326	1	3	2	1	2	1	1
14	1	228000	07/11/2018	103	1501	1	1	0	0	2	1	1
67	4	850000	08/02/2018	104	1467	2	3	3	2	1	1	1
81	11	640000	08/11/2018	104	1524	1	3	3	2	1	1	1
6	8	450000	08/15/2018	104	928	1	5	3	2	1	2	2
72	5	760000	08/15/2018	104	1382	1	4	3	2	1	1	1
20	5	1450000	08/31/2018	104	1388	2	4	3	2	1	2	1
96	10	550000	10/01/2018	106	1697	1	2	3	2	1	1	1
74	5	600000	10/15/2018	106	1498	1	4	3	2	1	1	1
66	4	720000	10/22/2018	106	1456	1	4	3	2	1	1	1
73	5	500000	11/07/2018	107	1382	1	4	3	2	1	1	1
75	5	750000	11/14/2018	107	441	1	4	3	2	1	1	1
52	5	760000	01/07/2019	109	1382	1	3	3	2	1	1	2
82	11	750000	01/23/2019	109	1387	2	4	3	2	1	1	1
54	6	800000	01/31/2019	109	1553	1	4	3	2	1	1	1
76	5	830000	02/17/2019	110	1413	2	4	3	2	1	1	1
50	5	860000	02/25/2019	110	1581	1	3	3	2	1	2	1
97	10	450000	03/07/2019	111	2193	1	2	3	2	1	1	3
103	9	470000	03/17/2019	111	1552	1	3	3	2	2	1	1
51	5	932000	04/04/2019	112	1494	1	3	3	2	1	2	1
91	3	1078000	05/13/2019	113	475	1	4	4	3	1	1	1
92	3	1250000	05/13/2019	113	603	1	4	4	3	1	2	1

79	1	1300000	05/20/2019	113	1318	2	4	3	2	1	1	1
53	6	580000	05/23/2019	113	1423	1	4	3	2	1	1	1
56	6	750000	05/30/2019	113	1452	2	2	3	2	1	1	1
24	4	950000	06/16/2019	114	1338	1	5	4	2	1	2	1
55	6	620000	06/19/2019	114	1301	1	4	3	2	1	1	1
25	4	900000	07/05/2019	115	1338	1	5	4	2	1	1	1
69	4	900000	07/05/2019	115	1338	1	3	4	3	1	1	1
7	5	850000	07/28/2019	115	1378	1	3	3	2	1	1	1
59	7	680000	07/30/2019	115	1013	2	3	3	2	1	2	1
57	6	550000	08/01/2019	116	1606	2	2	3	2	2	1	2
93	3	1088000	08/12/2019	116	1163	1	4	3	2	1	1	1
80	1	900000	08/13/2019	116	1448	2	2	3	2	1	1	1
106	1	4000000	09/06/2019	117	12269	2	5	4	3	1	2	1
68	4	850000	09/13/2019	117	1323	2	2	3	2	1	2	1
61	7	630000	09/16/2019	117	593	2	2	3	2	1	2	1
99	10	440000	09/19/2019	117	1353	1	4	3	2	1	1	2
105	1	2800000	09/25/2019	117	5008	1	4	4	3	1	2	1
84	11	605000	10/03/2019	118	1381	1	4	3	2	1	1	1
83	11	580000	10/04/2019	118	2662	1	2	3	2	1	1	2
98	10	635000	10/04/2019	118	937	2	3	3	2	1	1	1
70	4	777000	10/07/2019	118	1415	1	2	3	2	1	2	1
94	3	1075000	10/10/2019	118	1003	1	4	3	2	1	1	1
26	4	1050000	10/22/2019	118	1506	2	5	4	2	1	2	1
71	4	800000	11/04/2019	119	1360	1	4	3	2	1	1	1
85	2	1260000	11/06/2019	119	1825	3	4	3	2	1	1	1
77	1	2150000	11/18/2019	119	822	1	4	4	3	1	1	1
62	7	620000	11/19/2019	119	1487	1	3	3	2	1	1	1
102	9	642000	12/13/2019	120	2382	1	4	3	2	1	1	1
86	2	100000	12/16/2019	120	1834	2	4	3	2	1	1	1
63	7	482000	12/17/2019	120	1985	1	2	3	2	1	1	2
58	6	730000	01/09/2020	121	1301	1	3	3	2	1	1	1
64	7	500000	02/03/2020	122	1636	1	3	3	2	1	1	2
100	9	785000	02/18/2020	122	1110	2	2	3	2	1	2	1
65	7	470000	03/05/2020	123	1636	2	3	3	2	1	1	3
104	9	860000	03/13/2020	123	1110	2	5	3	2	1	1	1
5	11	700000	05/28/2020	125	1596	3	1	4	2	2	1	2
10	3	1300000	06/01/2020	126	1005	1	4	3	2	1	3	1

3	4	985000	06/03/2020	126	1704	2	5	4	2	1	2	2
9	2	1350000	07/02/2020	127	1929	2	5	4	3	1	1	1

Table E.1: Comparable properties tabl

Attribute tables:

Flat table		
ID	Flat description	Value adjustment
1	No flat	0
2	Standard open plan one bedroom	150000
3	Open plan 2 bedrooms	200000

Table E.2: Flat table

Garage table		
ID	Garage description	Garage value
1	No garage	0
2	Car port only	15000
3	Single garage	50000
4	Double garage	85000
5	More than 2 garage/ garage and carport	95000

Table E.3: Garage table

Bedrooms table		
ID	Bedrooms description	Value adjustment
1	Standard	0
2	Adjustment	100000

Table E.4: Bedrooms table

Bathrooms table		
ID	Bathroom description	Adjustment value
1	Standard	0
2	Adjustment	45000

Table E.5: Bathrooms table

Amenity table		
ID	Amenity desc	Amenity factor
1	No amenities	0
2	Swimming pool	100000
3	Built in bbq or playroom or bar	150000
4	Pool plus other amenities	250000

Table E.6: Amenity table

Out building table		
ID	Out building description	Out building factor
1	Standard/domestic quarters	50000
2	If no quarters	0

Table E.7: Out building table

Index table				
ID	Index date	Index value	Index difference	Index average

1	01/01/2010	69,3	0	0
2	02/01/2010	69,8	0	0
3	03/01/2010	70,3	0	0
4	04/01/2010	70,4	0	0
5	05/01/2010	70,6	0	0
6	06/01/2010	70,6	0	0
7	07/01/2010	71,1	0	0
8	08/01/2010	71,1	0	0
9	09/01/2010	71,1	0	0
10	10/01/2010	71,3	0	0
11	11/01/2010	71,4	0	0
12	12/01/2010	71,5	0	0
13	01/01/2011	71,9	0	0
14	02/01/2011	72,3	0	0
15	03/01/2011	73,2	0	0
16	04/01/2011	73,5	0	0
17	05/01/2011	73,8	0	0
18	06/01/2011	74,1	0	0
19	07/01/2011	74,7	0	0
20	08/01/2011	74,9	0	0
21	09/01/2011	75,2	0	0
22	10/01/2011	75,5	0	0
23	11/01/2011	75,8	0	0
24	12/01/2011	75,9	0	0
25	01/01/2012	76,3	0	0
26	02/01/2012	76,7	0	0
27	03/01/2012	77,6	0	0
28	04/01/2012	77,9	0	0
29	05/01/2012	77,9	0	0
30	06/01/2012	78,2	0	0
31	07/01/2012	78,4	0	0
32	08/01/2012	78,6	0	0
33	09/01/2012	79,3	0	0
34	10/01/2012	79,8	0	0
35	11/01/2012	80	0	0
36	12/01/2012	80,2	0	0
37	01/01/2013	80,4	0	0
38	02/01/2013	81,2	0	0
39	03/01/2013	82,2	0	0
40	04/01/2013	82,5	0	0
41	05/01/2013	82,3	0	0
42	06/01/2013	82,5	0	0
43	07/01/2013	83,4	0	0
44	08/01/2013	83,6	0	0
45	09/01/2013	84	0	0
46	10/01/2013	84,2	0	0
47	11/01/2013	84,3	0	0
48	12/01/2013	84,5	0	0
49	01/01/2014	85,1	0	0
50	02/01/2014	86	0	0
51	03/01/2014	87,2	0	0
52	04/01/2014	87,6	0	0
53	05/01/2014	87,7	0	0
54	06/01/2014	88	0	0
55	07/01/2014	88,7	0	0
56	08/01/2014	89	0	0
57	09/01/2014	89	0	0
58	10/01/2014	89,2	0	0

59	11/01/2014	89,2	0	0
60	12/01/2014	89	0	0
61	01/01/2015	88,9	0	0
62	02/01/2015	89,4	0	0
63	03/01/2015	90,7	0	0
64	04/01/2015	91,5	0	0
65	05/01/2015	91,7	0	0
66	06/01/2015	92,1	0	0
67	07/01/2015	93,1	0	0
68	08/01/2015	93,1	0	0
69	09/01/2015	93,1	0	0
70	10/01/2015	93,3	0	0
71	11/01/2015	93,4	0	0
72	12/01/2015	93,7	0	0
73	01/01/2016	94,4	0	0
74	02/01/2016	95,7	0	0
75	03/01/2016	96,4	0	0
76	04/01/2016	97,2	0	0
77	05/01/2016	97,4	0	0
78	06/01/2016	97,9	0	0
79	07/01/2016	98,7	0	0
80	08/01/2016	98,6	0	0
81	09/01/2016	98,8	0	0
82	10/01/2016	99,3	0	0
83	11/01/2016	99,6	0	0
84	12/01/2016	100	0	0
85	01/01/2017	100,6	0	0
86	02/01/2017	101,7	0	0
87	03/01/2017	102,3	0	0
88	04/01/2017	102,4	0	0
89	05/01/2017	102,7	0	0
90	06/01/2017	102,9	0	0
91	07/01/2017	103,2	0	0
92	08/01/2017	103,3	0	0
93	09/01/2017	103,8	0	0
94	10/01/2017	104,1	0	0
95	11/01/2017	104,2	0	0
96	12/01/2017	104,7	0	0
97	01/01/2018	105	0	0
98	02/01/2018	105,8	0	0
99	03/01/2018	106,2	0	0
100	04/01/2018	107	0	0
101	05/01/2018	107,2	0	0
102	06/01/2018	107,6	0	0
103	07/01/2018	108,5	0	0
104	08/01/2018	108,4	0	0
105	09/01/2018	108,9	0	0
106	10/01/2018	109,4	0	0
107	11/01/2018	109,6	0	0
108	12/01/2018	109,4	0	0
109	01/01/2019	109,2	0	0
110	02/01/2019	110,1	0	0
111	03/01/2019	111	0	0
112	04/01/2019	111,7	0	0
113	05/01/2019	112	0	0
114	06/01/2019	112,4	0	0
115	07/01/2019	112,8	0	0
116	08/01/2019	113,1	0	0

117	09/01/2019	113,4	0	0
118	10/01/2019	113,4	0	0
119	11/01/2019	113,5	0	0
120	12/01/2019	113,8	0	0
121	01/01/2020	114,1	0	0
122	02/01/2020	115,2	0	0
123	03/01/2020	115,6	0	0
124	04/01/2020	115	0	0
125	05/01/2020	114,3	0	0
126	06/01/2020	114,9	0	0
127	07/01/2020	116,4	0	0
133	08/01/2020	116,6	0	0
134	09/01/2020	116,8	0	0
135	10/01/2020	117,1	0	0
136	12/01/2020	117,3	0	0

Table E.8: Index table

Condition table		
ID	Condition description	Value adjustment
1	Habitable and properly maintained	0
2	Require standard maintenance eg painting	5
3	Require structural maintenance	20

Table E.9: Condition table

Suburb table		
ID	Suburb name	Average price
1	Wilkoppies	1787000
2	Flamwood	1200000
3	Doringkruin	1098500
4	La Hoff	888200
5	Flimieda	870000
6	Neserhof	850000
7	Meiringspark	800000
8	Ellaton	650000
9	Adamay view	672833
10	Roosheuwel	540833
11	Elandsheuwel	550000

Table E.10: Suburb table

Attribute rank table		
ID	Attribute name	Attribute score
1	suburb_id	6,7E-05
2	index_id	0,808116
3	property_size	0
5	flat_id	0,690482
6	property_garage_id	0,565373
7	property_bedrooms_amount	0,91753
8	property_bathrooms_amount	0,133004
9	out_building_id	0,788372
10	amenity_id	0,003496
11	property_condition_id	0,013823

Table E.11: Attribute rank table

APPENDIX F: PROPERTY VALUATION DSS DEMONSTRATOR C# CODE

F.1 Represents the object classes used throughout the system.

F.1.1 Object classes that represent the attributes of a property.

```
class Amenity
{
    public int amenity_id { get; set; }
    public string amenity_description { get; set; }
    public double amenity_adjustment { get; set; }
}

class Condition
{
    public int condition_id { get; set; }
    public string condition_description { get; set; }
    public double condition_modifier { get; set; }
}

class Flat
{
    public int flat_id { get; set; }
    public string flat_description { get; set; }
    public double flat_adjustment { get; set; }
}

class Out_Buildings
{
    public int out_building_id { get; set; }
    public string out_building_description { get; set; }
    public double out_building_adjustment { get; set; }
}

class Garage
{
    public int garage_id { get; set; }
    public string garage_description { get; set; }
    public double garage_adjustment { get; set; }
}

class Index
{
    public int index_id { get; set; }
    public DateTime index_date { get; set; }
    public double index_value { get; set; }
}

class Suburb
{
    public int suburb_id { get; set; }
    public string suburb_name { get; set; }
}
```

```

        public double suburb_average_price { get; set; }
    }

class Type
{
    public int type_id { get; set; }
    public string type_description { get; set; }
}

```

F.1.2 Object class that represents how the expert DSS utilizes a property for calculation purposes and to link a property's attributes through identification numbers.

```

class Property
{
    public int adjust_need { get; set; }
    public string property_address { get; set; }
    public int suburb_id { get; set; }
    public double property_price { get; set; }
    public DateTime sale_date { get; set; }
    public int index_id { get; set; }
    public double property_size { get; set; }
    public int flat_id { get; set; }
    public int property_garage_id { get; set; }
    public int property_bedrooms_amount { get; set; }
    public int property_bathrooms_amount { get; set; }
    public int out_building_id { get; set; }
    public int amenity_id { get; set; }
    public int property_condition_id { get; set; }
    public int type_id { get; set; }
}

```

F.1.3. Object class that represents a property when it is displayed to the user. This excludes the identification numbers of the previous object class.

```

class Complete_Prop
{
    public int Adjustments { get; set; }
    public string property_address { get; set; }
    public string suburb_name { get; set; }
    public double property_price { get; set; }
    public DateTime sale_date { get; set; }
    public int index_id { get; set; }
    public double property_size { get; set; }
    public string flat_description { get; set; }
    public string property_garage_description { get; set; }
    public int property_bedrooms_amount { get; set; }
    public int property_bathrooms_amount { get; set; }
    public string out_building_description { get; set; }
    public string amenity_description { get; set; }
    public string property_condition_description { get; set; }
    public string type_description { get; set; }
}

```

F.1.4. Object class displays how the expert DSS stores adjustment values within an object, along with its accompanying attribute name. An example of an adjustment object contains

the name *Garage* and an adjustment value of *R15,000* if the subject property must be adjusted for the garage.

```
class Adjustments
{
    public string adjustment_name { get; set; }
    public double adjustment_value { get; set; }
}
```

F.1.5. The attribute ranking object which describes how the system would store the list of attributes with their corresponding ranks. This is used when selecting comparable properties.

```
class Attribute_Rank
{
    public int attribute_id { get; set; }
    public string attribute_description { get; set; }
    public double attribute_rank { get; set; }
}
```

F.1.6. Object class, typically used within a list, that represents a translated version of the class in F.1.5. into SQL criteria when selecting comparable properties from the database.

```
class Prop_Crit
{
    public double rank { get; set; }
    public string definition { get; set; }
    public string type { get; set; }
}
```

F.2. Represent action classes containing function to communicate with the database, select comparable properties, calculate new attribute rankings, search for a new index and calculate the value of a property.

F.2.1 An action class that is used to communicate with the database to retrieve all the historic data of attributes and properties to populate the dropdown lists on the user interface and supply adjustment values to the mathematical model.

```
class DB_Statements
{
    public static Property subject_property;

    public static List<Suburb> combo_suburb = new List<Suburb>();
    public static List<Type> combo_type = new List<Type>();
    public static List<Flat> combo_flat = new List<Flat>();
    public static List<Garage> combo_garage = new List<Garage>();
    public static List<Out_Buildings> combo_out_buildings = new
List<Out_Buildings>();
    public static List<Amenity> combo_amenity = new List<Amenity>();
    public static List<Condition> combo_condition = new List<Condition>();
    public static List<Index> combo_index = new List<Index>();
}
```

```

        public static List<Attribute_Rank> main_subject_attributes = new
List<Attribute_Rank>();

        public static string comparable_suburb = "";
        public static Index latIndex = new Index();

//This class contains all of the functions and statements that is used to interact with
the database to populate the front screen with values from the database

        //This function retrieves the property attributes to be used by the system for
calculation and selection from the database
        public static List<Attribute_Rank> get_attributes()
        {
            List<Attribute_Rank> attribute_return = new List<Attribute_Rank>();
            string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

            using (OleDbConnection connection = new OleDbConnection(connString))
            {
                connection.Open();
                try
                {
                    OleDbDataReader reader = null;
                    OleDbCommand command = new OleDbCommand("SELECT * FROM
attribute_rank_table", connection);
                    reader = command.ExecuteReader();
                    while (reader.Read())
                    {
                        attribute_return.Add(new Attribute_Rank { attribute_id =
Convert.ToInt32(reader[0].ToString()), attribute_description = reader[1].ToString(),
attribute_rank = Convert.ToDouble(reader[2].ToString()) });
                        //Convert.ToInt32(reader[0].ToString());
                    }
                    connection.Close();
                }
                catch (Exception exc)
                {
                    MessageBox.Show($"Something happened from attributes: \r\n\r\n
{exc}");
                }
                connection.Close();
            }
            return attribute_return;
        }

//The following functions retrieve the attributes and their associated values and store
them within their corresponding lists
        public static List<Suburb> populate_suburb_list ()
        {
            List<Suburb> suburb_return = new List<Suburb>();
            string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

            using (OleDbConnection connection = new OleDbConnection(connString))
            {
                connection.Open();
                try
                {
                    OleDbDataReader reader = null;
                    OleDbCommand command = new OleDbCommand("SELECT ID, suburb_name,
average_price FROM suburb_table", connection);

```

```

        reader = command.ExecuteReader();
        while (reader.Read())
        {
            suburb_return.Add(new Suburb { suburb_id =
Convert.ToInt32(reader[0].ToString()), suburb_name = reader[1].ToString(),
suburb_average_price = Convert.ToDouble(reader[2].ToString()) });
        }
        connection.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show($"Something happened: \r\n\r\n {exc}");
    }
    connection.Close();
}
return suburb_return;
}

public static List<Index> populate_index_list()
{
    List<Index> index_return = new List<Index>();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;
            OleDbCommand command = new OleDbCommand("SELECT ID, index_date,
index_val FROM index_table", connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                index_return.Add(new Index { index_id =
Convert.ToInt32(reader[0].ToString()), index_date =
Convert.ToDateTime(reader[1].ToString()), index_value =
Convert.ToDouble(reader[2].ToString()) }); //Convert.ToInt32(reader[0].ToString());
            }
            connection.Close();
        }
        catch (Exception exc)
        {
            MessageBox.Show($"Something happened: \r\n\r\n {exc}");
        }
        connection.Close();
    }
    return index_return;
}

public static List<Type> populate_type_list()
{
    List<Type> type_return = new List<Type>();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;

```

```

        OleDbCommand command = new OleDbCommand("SELECT ID, type_desc FROM
type_table", connection);
        reader = command.ExecuteReader();
        while (reader.Read())
        {
            type_return.Add(new Type {type_id =
Convert.ToInt32(reader[0].ToString()), type_description = reader[1].ToString()});
        }
        connection.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show($"Something happened: \r\n\r\n {exc}");
    }
    connection.Close();
}
return type_return;
}

```

```

public static List<Flat> populate_flat_list()
{
    List<Flat> flat_return = new List<Flat>();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;
            OleDbCommand command = new OleDbCommand("SELECT ID, flat_desc,
value_adjustment FROM flat_table", connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                flat_return.Add(new Flat {flat_id =
Convert.ToInt32(reader[0].ToString()), flat_description = reader[1].ToString(),
flat_adjustment = Convert.ToDouble(reader[2].ToString()) });
            }
            connection.Close();
        }
        catch (Exception exc)
        {
            MessageBox.Show($"Something happened: \r\n\r\n {exc}");
        }
        connection.Close();
    }
    return flat_return;
}

```

```

public static List<Garage> populate_garage_list()
{
    List<Garage> garage_return = new List<Garage>();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {

```

```

        OleDbDataReader reader = null;
        OleDbCommand command = new OleDbCommand("SELECT ID,
garage_description, garage_factor FROM garage_table", connection);
        reader = command.ExecuteReader();
        while (reader.Read())
        {
            garage_return.Add(new Garage {garage_id =
Convert.ToInt32(reader[0].ToString()), garage_description = reader[1].ToString(),
garage_adjustment = Convert.ToDouble(reader[2].ToString()) });
        }
        connection.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show($"Something happened: \r\n\r\n {exc}");
    }
    connection.Close();
}
return garage_return;
}

public static List<Out_Buildings> populate_out_buildings_list()
{
    List<Out_Buildings> out_buildings_return = new List<Out_Buildings>();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;
            OleDbCommand command = new OleDbCommand("SELECT ID,
out_building_desc, out_building_factor FROM out_building_table", connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                out_buildings_return.Add(new Out_Buildings {out_building_id =
Convert.ToInt32(reader[0].ToString()), out_building_description = reader[1].ToString(),
out_building_adjustment = Convert.ToDouble(reader[2].ToString()) });
            }
            connection.Close();
        }
        catch (Exception exc)
        {
            MessageBox.Show($"Something happened: \r\n\r\n {exc}");
        }
        connection.Close();
    }
    return out_buildings_return;
}

public static List<Amenity> populate_amenity_list()
{
    List<Amenity> amenity_return = new List<Amenity>();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try

```

```

    {
        OleDbDataReader reader = null;
        OleDbCommand command = new OleDbCommand("SELECT ID, amenity_desc,
amenity_factor FROM amenity_table", connection);
        reader = command.ExecuteReader();
        while (reader.Read())
        {
            amenity_return.Add(new Amenity {amenity_id =
Convert.ToInt32(reader[0].ToString()), amenity_description = reader[1].ToString(),
amenity_adjustment = Convert.ToDouble(reader[2].ToString()) });
        }
        connection.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show($"Something happened: \r\n\r\n {exc}");
    }
    connection.Close();
}
return amenity_return;
}

```

```

public static List<Condition> populate_condition_list()
{
    List<Condition> condition_return = new List<Condition>();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;
            OleDbCommand command = new OleDbCommand("SELECT ID,
condition_description, value_adjustment FROM condition_table", connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                condition_return.Add(new Condition {condition_id =
Convert.ToInt32(reader[0].ToString()), condition_description = reader[1].ToString(),
condition_modifier = Convert.ToDouble(reader[2].ToString()) });
            }
            connection.Close();
        }
        catch (Exception exc)
        {
            MessageBox.Show($"Something happened: \r\n\r\n {exc}");
        }
        connection.Close();
    }
    return condition_return;
}

```

//This function retrieves the latest index from the database to be used for any properties that are valuated at a newer date than the latest index stored within the database

```

public static Index getLatestIndexDB()
{
    Index index_return = new Index();
    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

```

```

using (OleDbConnection connection = new OleDbConnection(connString))
{
    connection.Open();
    try
    {
        OleDbDataReader reader = null;
        OleDbCommand command = new OleDbCommand("SELECT TOP 1 * FROM
index_table ORDER BY ID DESC", connection);
        reader = command.ExecuteReader();
        while (reader.Read())
        {
            index_return = new Index { index_id =
Convert.ToInt32(reader[0].ToString()), index_date =
Convert.ToDateTime(reader[1].ToString()), index_value =
Convert.ToDouble(reader[2].ToString()) };
        }
        connection.Close();
    }
    catch (Exception exc)
    {
        MessageBox.Show($"Something happened: \r\n\r\n {exc}");
    }
    connection.Close();
}

return index_return;
}

//This function is used to store a new index value when a new value exists online
public static void postNewIndex(DateTime indDate, string indval)
{
    string indsval = indval;
    indsval = indsval.Replace('.', ',');

    string connectionString = @"Provider = Microsoft.ACE.OLEDB.12.0; Data Source
= pvDB.accdb";

    using (OleDbConnection con = new OleDbConnection(connectionString))
    {
        con.Open();

        string query = "INSERT INTO index_table " +
            "([index_date], [index_val]) " +
            "VALUES (@index_date,@index_val)";

        using (var command = new System.Data.OleDb.OleDbCommand(query, con))
        {
            command.Parameters.AddWithValue("@index_date",
indDate.ToString("MM/dd/yyyy"));
            command.Parameters.AddWithValue("@index_val", indsval);
            command.ExecuteNonQuery();
        }

        con.Close();

        MessageBox.Show("New Index successfully submitted!");
    }
}

//This function retrieves all of the databases properties and stores it within a list for
later use

```

```

public static List<Property> get_properties()
{
    List<Property> properties_return = new List<Property>();

    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;
            OleDbCommand command = new OleDbCommand("SELECT *" +
                " FROM properties_table",
                connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                properties_return.Add(new Property
                {
                    property_address = reader[1].ToString(),
                    suburb_id = Convert.ToInt32(reader[2].ToString()),
                    property_price = Convert.ToDouble(reader[3].ToString()),
                    sale_date = Convert.ToDateTime(reader[4].ToString()),
                    index_id = Convert.ToInt32(reader[5].ToString()),
                    property_size = Convert.ToDouble(reader[6].ToString()),
                    flat_id = Convert.ToInt32(reader[7].ToString()),
                    property_garage_id = Convert.ToInt32(reader[8].ToString()),
                    property_bedrooms_amount =
Convert.ToInt32(reader[9].ToString()),
                    property_bathrooms_amount =
Convert.ToInt32(reader[10].ToString()),
                    out_building_id = Convert.ToInt32(reader[11].ToString()),
                    amenity_id = Convert.ToInt32(reader[12].ToString()),
                    property_condition_id =
Convert.ToInt32(reader[13].ToString()),
                    type_id = Convert.ToInt32(reader[14].ToString())
                });
            }
            connection.Close();
        }
        catch (Exception exc)
        {
            MessageBox.Show($"Something happened: \r\n\r\n {exc}");
        }
        connection.Close();
    }
    return properties_return;
}

```

F.2.2 Class that represents the regression functionality of the expert DSS to calculate new attribute ranking values of each property attributes, using an external python program.

```

//This class describes how the system's regression functionality works.
class Regression
{
    public static List<Property> dblist;
    public static List<Attribute_Rank> regressRanks;
    public static float presult;

```

//This function retrieves the regression results that were produced by the python tool from a file "results.txt"

```
public static string[] getRegressResults()
{
    string[] lines = System.IO.File.ReadAllLines("results.txt");
    for (int i = 0; i < lines.Length; i++)
    {
        lines[i] = Regex.Replace(lines[i], "[^0-9.]", "");
        lines[i] = lines[i].Replace('.', ',');
    }
    return lines;
}
```

//This function writes the total comparable properties table's records to a format that the python tool can read.

//Each individual attribute is written to its own file, that contains the entire properties table's attributes values

```
public static void writeFiles()
{
    dblist = new List<Property>();

    dblist = get_properties();

    File.Delete("suburb_id.txt");
    using (StreamWriter writer = new StreamWriter("suburb_id.txt"))
    {
        //writer.WriteLine("Monica Rathbun");
        foreach (var x in dblist)
        {
            writer.WriteLine(x.suburb_id);
        }
    }

    File.Delete("index_id.txt");
    using (StreamWriter writer = new StreamWriter("index_id.txt"))
    {
        //writer.WriteLine("Monica Rathbun");
        foreach (var x in dblist)
        {
            writer.WriteLine(x.index_id);
        }
    }

    File.Delete("property_size.txt");
    using (StreamWriter writer = new StreamWriter("property_size.txt"))
    {
        //writer.WriteLine("Monica Rathbun");
        foreach (var x in dblist)
        {
            writer.WriteLine(x.property_size);
        }
    }

    File.Delete("flat_id.txt");
    using (StreamWriter writer = new StreamWriter("flat_id.txt"))
    {
        //writer.WriteLine("Monica Rathbun");
        foreach (var x in dblist)
        {
            writer.WriteLine(x.flat_id);
        }
    }
}
```

```

File.Delete("property_garage_id.txt");
using (StreamWriter writer = new StreamWriter("property_garage_id.txt"))
{
    //writer.WriteLine("Monica Rathbun");
    foreach (var x in dblist)
    {
        writer.WriteLine(x.property_garage_id);
    }
}

File.Delete("property_bedrooms_amount.txt");
using (StreamWriter writer = new
StreamWriter("property_bedrooms_amount.txt"))
{
    //writer.WriteLine("Monica Rathbun");
    foreach (var x in dblist)
    {
        writer.WriteLine(x.property_bedrooms_amount);
    }
}

File.Delete("property_bathrooms_amount.txt");
using (StreamWriter writer = new
StreamWriter("property_bathrooms_amount.txt"))
{
    //writer.WriteLine("Monica Rathbun");
    foreach (var x in dblist)
    {
        writer.WriteLine(x.property_bathrooms_amount);
    }
}

File.Delete("out_building_id.txt");
using (StreamWriter writer = new StreamWriter("out_building_id.txt"))
{
    //writer.WriteLine("Monica Rathbun");
    foreach (var x in dblist)
    {
        writer.WriteLine(x.out_building_id);
    }
}

File.Delete("amenity_id.txt");
using (StreamWriter writer = new StreamWriter("amenity_id.txt"))
{
    //writer.WriteLine("Monica Rathbun");
    foreach (var x in dblist)
    {
        writer.WriteLine(x.amenity_id);
    }
}

File.Delete("property_condition_id.txt");
using (StreamWriter writer = new StreamWriter("property_condition_id.txt"))
{
    //writer.WriteLine("Monica Rathbun");
    foreach (var x in dblist)
    {
        writer.WriteLine(x.property_condition_id);
    }
}

```

```

File.Delete("property_price.txt");
using (StreamWriter writer = new StreamWriter("property_price.txt"))
{
    //writer.WriteLine("Monica Rathbun");
    foreach (var x in dblist)
    {
        writer.WriteLine(x.property_price);
    }
}

}

//This function executes the python regression file
public static bool runPythonScript(string filepy)
{
    var psi = new ProcessStartInfo();
    psi.FileName = @"C:\Users\Thys
Wentzel\AppData\Local\Programs\Python\Python38\python.exe";

    var script = filepy;

    psi.Arguments = $"\"{script}\"";

    psi.UseShellExecute = false;
    psi.CreateNoWindow = true;
    psi.RedirectStandardInput = true;
    psi.RedirectStandardError = true;

    try
    {
        using (var process = Process.Start(psi))
        {
        }
        return true;
    }
    catch
    {
        return false;
    }
}

//This function allows the new regression results to be posted to the database
public static void postNewRanks(List<double> mydbl)
{
    string connectionString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (var con = new System.Data.OleDb.OleDbConnection(connectionString))
    {
        con.Open();

        string query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'suburb_id'";
        var cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[0].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'index_id'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[1].ToString());
        cmd.ExecuteNonQuery();
    }
}

```

```

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'property_size'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[2].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'flat_id'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[3].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'property_garage_id'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[4].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'property_bedrooms_amount'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[5].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'property_bathrooms_amount'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[6].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'out_building_id'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[7].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'amenity_id'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[8].ToString());
        cmd.ExecuteNonQuery();

        query = "UPDATE attribute_rank_table SET attribute_score =
@attributescore WHERE attribute_name = 'property_condition_id'";
        cmd = new OleDbCommand(query, con);
        cmd.Parameters.AddWithValue("@attributescore", mydbl[9].ToString());
        cmd.ExecuteNonQuery();

        con.Close();

        MessageBox.Show("Successfully updated rekords!");
    }
}

```

F.2.2.1 The python program code that executes regression on the property attributes.

```
import statsmodels.api as sm
```

```

from pandas import DataFrame

my_file = "suburb_id.txt"
suburb_id = [line.rstrip('\n') for line in open(my_file)]
suburb_id = [int(i) for i in suburb_id]
print(suburb_id)

my_file = "index_id.txt"
index_id = [line.rstrip('\n') for line in open(my_file)]
index_id = [int(i) for i in index_id]
print(index_id)

my_file = "property_size.txt"
property_size = [line.rstrip('\n') for line in open(my_file)]
property_size = [int(i) for i in property_size]
print(property_size)

my_file = "flat_id.txt"
flat_id = [line.rstrip('\n') for line in open(my_file)]
flat_id = [int(i) for i in flat_id]
print(flat_id)

my_file = "property_garage_id.txt"
property_garage_id = [line.rstrip('\n') for line in open(my_file)]
property_garage_id = [int(i) for i in property_garage_id]
print(property_garage_id)

my_file = "property_bedrooms_amount.txt"
property_bedrooms_amount = [line.rstrip('\n') for line in open(my_file)]
property_bedrooms_amount = [int(i) for i in property_bedrooms_amount]
print(property_bedrooms_amount)

my_file = "property_bathrooms_amount.txt"
property_bathrooms_amount = [line.rstrip('\n') for line in open(my_file)]
property_bathrooms_amount = [int(i) for i in property_bathrooms_amount]
print(property_bathrooms_amount)

my_file = "out_building_id.txt"
out_building_id = [line.rstrip('\n') for line in open(my_file)]
out_building_id = [int(i) for i in out_building_id]
print(out_building_id)

my_file = "amenity_id.txt"
amenity_id = [line.rstrip('\n') for line in open(my_file)]
amenity_id = [int(i) for i in amenity_id]
print(amenity_id)

my_file = "property_condition_id.txt"
property_condition_id = [line.rstrip('\n') for line in open(my_file)]
property_condition_id = [int(i) for i in property_condition_id]
print(property_condition_id)

my_file = "property_price.txt"
property_price = [line.rstrip('\n') for line in open(my_file)]
property_price = [float(i) for i in property_price]
print(property_price)

properties = { 'suburb_id': suburb_id,
               'index_id': index_id,
               'property_size': property_size,
               'flat_id': flat_id,
               'property_garage_id': property_garage_id,
               'property_bedrooms_amount': property_bedrooms_amount,

```

```

        'property_bathrooms_amount': property_bathrooms_amount,
        'out_building_id': out_building_id,
        'amenity_id': amenity_id,
        'property_condition_id': property_condition_id,
        'property_price': property_price
    }

print("Everything together")
print(properties)

df = DataFrame(properties, columns=['suburb_id', 'index_id', 'property_size', 'flat_id',
    'property_garage_id', 'property_bedrooms_amount', 'property_bathrooms_amount',
    'out_building_id', 'amenity_id', 'property_condition_id', 'property_price'])

X = df[['suburb_id', 'index_id', 'property_size', 'flat_id',
    'property_garage_id', 'property_bedrooms_amount', 'property_bathrooms_amount',
    'out_building_id', 'amenity_id', 'property_condition_id']]

Y = df['property_price']

X = sm.add_constant(X)

model = sm.OLS(Y, X).fit()
predictions = model.predict(X)

print_model = model.summary()
print(print_model)

pvallist = model.pvalues.tolist()

print(pvallist)

fig = sm.graphics.plot_partregress_grid(prestige_model)
fig.tight_layout(pad=1.0)

with open('results.txt', 'w') as file:
    for x in pvallist:
        file.write("%f\n" % x)

```

F.2.3 Action class that represents the expert DSS's functionality to retrieve a new consumer price index (CPI) value from online.

```

//This class is used to search online for the latest index value to be posted towards the
database
class webcrawl
{
    //This function retrieves the latest index from an online source
    public static string[] getLatestIndexWeb()
    {
        string[] results = {"", ""};
        string HTML;
        using (var wc = new WebClient()) // "using" keyword automatically closes
WebClient stream on download completed
        {
            HTML = wc.DownloadString("https://www.economy.com/south-africa/consumer-
price-index-cpi");
            //Console.WriteLine("The retrieved HTML is \r\n" + HTML);
        }
    }
}

```

```

        // create HtmlAgilityPack document object from HTML
        var doc = new HtmlAgilityPack.HtmlDocument();
        doc.LoadHtml(HTML);

        // parsing HTML
        results[0] =
doc.DocumentNode.SelectSingleNode("//th[.='Data']").NextSibling.NextSibling.InnerText;
        results[1] =
doc.DocumentNode.SelectSingleNode("//th[.='Data']").NextSibling.NextSibling.NextSibling.N
extSibling.FirstChild.NextSibling.InnerText;
        results[1] = Regex.Replace(results[1], @"\s+", "");

        /*foreach(string value in results)
        {
            Console.WriteLine(value);
        } */

        return results;
    }

    public static int checkIndMonthS(string month)
    {
        int monthval = 1;
        switch (month)
        {
            case "Jan":
                monthval = 1;
                break;
            case "Feb":
                monthval = 2;
                break;
            case "Mar":
                monthval = 3;
                break;
            case "Apr":
                monthval = 4;
                break;
            case "May":
                monthval = 5;
                break;
            case "Jun":
                monthval = 6;
                break;
            case "Jul":
                monthval = 7;
                break;
            case "Aug":
                monthval = 8;
                break;
            case "Sep":
                monthval = 9;
                break;
            case "Oct":
                monthval = 10;
                break;
            case "Nov":
                monthval = 11;
                break;
            case "Dec":
                monthval = 12;
                break;
        }
        return monthval;
    }

```

```

}

public static string checkIndMonthI(int month)
{
    string monthval = "Null";
    switch (month)
    {
        case 1:
            monthval = "Jan";
            break;
        case 2:
            monthval = "Feb";
            break;
        case 3:
            monthval = "Mar";
            break;
        case 4:
            monthval = "Apr";
            break;
        case 5:
            monthval = "May";
            break;
        case 6:
            monthval = "Jun";
            break;
        case 7:
            monthval = "Jul";
            break;
        case 8:
            monthval = "Aug";
            break;
        case 9:
            monthval = "Sep";
            break;
        case 10:
            monthval = "Oct";
            break;
        case 11:
            monthval = "Nov";
            break;
        case 12:
            monthval = "Dec";
            break;
    }
    return monthval;
}

public static DateTime getIndexDate(string month, int year)
{
    DateTime result;

    result = new DateTime(year, checkIndMonthS(month), 1);

    return result;
}

```

```

    }
}

```

F.2.4. The class that represents the expert DSS's main functionality to select comparable properties from the database, adjust for each attribute that is different and calculate a new property value.

```

//This class combines mathematical model and knowledge base related functions to find
comparable properties to the subject property and calculate the subject property's new
value
class Property_Price_Calculation
{
    public static List<Attribute_Rank> attribute_list_exclusion = new
List<Attribute_Rank>();
    public static List<Property> comparable_properties = new List<Property>();
    public static List<Complete_Prop> main_properties = new List<Complete_Prop>();
    public static List<Prop_Crit> prop_crit_list = new List<Prop_Crit>();
    public static List<Prop_Crit> prop_crit_main_list = new List<Prop_Crit>();
    public static List<Prop_Crit> crit_adjust_list = new List<Prop_Crit>();
    public static List<Prop_Crit> compare_adjust_list = new List<Prop_Crit>();
    public static List<Property> myProperties = new List<Property>();
    public static List<Adjustments> main_adjustments = new List<Adjustments>();
    public static float main_property_value = 0;
    public static int property_counter;

    //This function populates a list of required adjustments for the current subject
property
    public static void adjust_criteria()
    {
        if (prop_crit_list.Count > 0)
        {
            crit_adjust_list.Add(prop_crit_main_list[prop_crit_main_list.Count - 1]);

            prop_crit_list.RemoveAt(prop_crit_list.Count - 1);
            prop_crit_main_list.RemoveAt(prop_crit_main_list.Count - 1);
        }
        else
        {
            Console.WriteLine("Cannot adjust any further");
        }
    }

    //This function returns a list of all the required adjustments to be made to the
subject property
    public static List<Prop_Crit> get_adjust_list(Property subject, Property
comparable)
    {
        List<Prop_Crit> need_adjust = new List<Prop_Crit>();

        if(subject.amenity_id != comparable.amenity_id)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Amenities", rank
= 0 });
        }
        if (subject.flat_id != comparable.flat_id)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Flat", rank = 0
});
        }
    }
}

```

```

        if (subject.index_id != comparable.index_id)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Date of sale",
rank = 0 });
        }
        if (subject.out_building_id != comparable.out_building_id)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Out buildings",
rank = 0 });
        }
        if (subject.property_garage_id != comparable.property_garage_id)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Garage", rank =
0 });
        }
        if (subject.property_condition_id != comparable.property_condition_id)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Condition", rank
= 0 });
        }
        if (subject.property_bathrooms_amount !=
comparable.property_bathrooms_amount)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Amount of
bathrooms", rank = 0 });
        }
        if (subject.property_bedrooms_amount != comparable.property_bedrooms_amount)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Amount of
bedrooms", rank = 0 });
        }
        if (subject.suburb_id != comparable.suburb_id)
        {
            need_adjust.Add(new Prop_Crit { definition = "", type = "Suburb", rank =
0 });
        }
    }
    return need_adjust;
}

//This function counts the number of required adjustments
public static List<Property> check_adjust_count(List<Property> comparables,
Property subject)
{
    List<Property> adjust_return = new List<Property>();
    adjust_return = comparables;

    if(adjust_return.Count > 0)
    {
        for(int i = 0; i < adjust_return.Count; i++)
        {
            int adjust_count = 0;

            if(subject.amenity_id != adjust_return[i].amenity_id)
            {
                adjust_count++;
            }
            if (subject.flat_id != adjust_return[i].flat_id)
            {
                adjust_count++;
            }
            if (subject.out_building_id != adjust_return[i].out_building_id)
            {

```

```

        adjust_count++;
    }
    if (subject.property_bathrooms_amount !=
adjust_return[i].property_bathrooms_amount)
    {
        adjust_count++;
    }
    if (subject.property_bedrooms_amount !=
adjust_return[i].property_bedrooms_amount)
    {
        adjust_count++;
    }
    if (subject.property_condition_id !=
adjust_return[i].property_condition_id)
    {
        adjust_count++;
    }
    if (subject.property_garage_id !=
adjust_return[i].property_garage_id)
    {
        adjust_count++;
    }
    if (subject.suburb_id != adjust_return[i].suburb_id)
    {
        adjust_count++;
    }
    if (subject.index_id != adjust_return[i].index_id)
    {
        adjust_count++;
    }

    adjust_return[i].adjust_need = adjust_count;
}
}

return adjust_return;
}

```

```

public static List<Complete_Prop> check_adjust_main_count(List<Property>
comparables, Property subject, List<Complete_Prop> complete_compare)
{
    List<Complete_Prop> adjust_return = new List<Complete_Prop>();
    adjust_return = complete_compare;

    if (comparables.Count > 0)
    {
        for (int i = 0; i < comparables.Count; i++)
        {
            int adjust_count = 0;

            if (subject.amenity_id != comparables[i].amenity_id)
            {
                adjust_count++;
            }
            if (subject.flat_id != comparables[i].flat_id)
            {
                adjust_count++;
            }
            if (subject.out_building_id != comparables[i].out_building_id)
            {
                adjust_count++;
            }
        }
    }
}

```

```

        if (subject.property_bathrooms_amount !=
comparables[i].property_bathrooms_amount)
        {
            adjust_count++;
        }
        if (subject.property_bedrooms_amount !=
comparables[i].property_bedrooms_amount)
        {
            adjust_count++;
        }
        if (subject.property_condition_id !=
comparables[i].property_condition_id)
        {
            adjust_count++;
        }
        if (subject.property_garage_id != comparables[i].property_garage_id)
        {
            adjust_count++;
        }
        if (subject.suburb_id != comparables[i].suburb_id)
        {
            adjust_count++;
        }
        if (subject.index_id != comparables[i].index_id)
        {
            adjust_count++;
        }

        adjust_return[i].Adjustments = adjust_count;
    }
}

return adjust_return;
}

```

//This function the selection query's list of attributes to search for, in order of importance

```

//The values represents the subject property's attributes values.
//The property size has been removed from the equation, because it did not
properly describe the value of a property
public static void get_criteria(Property my_sproperty)
{
    compare_adjust_list.Clear();
    crit_adjust_list.Clear();
    prop_crit_list.Clear();
    prop_crit_main_list.Clear();

    double size_crit_higher = my_sproperty.property_size + 500;
    double size_crit_lower = my_sproperty.property_size - 500;

    prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[0].attribute_description} =
{my_sproperty.suburb_id}", rank =
DB_Statements.main_subject_attributes[0].attribute_rank, type = "suburb_id" });
    prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[1].attribute_description} =
{my_sproperty.index_id}", rank = DB_Statements.main_subject_attributes[1].attribute_rank,
type = "index_id" });
    //prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[2].attribute_description} >= {size_crit_lower}",
rank = DB_Statements.main_subject_attributes[2].attribute_rank, type = "property_size"
});
}

```

```

        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[3].attribute_description} =
{my_sproperty.flat_id}", rank = DB_Statements.main_subject_attributes[3].attribute_rank,
type = "flat_id" });
        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[4].attribute_description} =
{my_sproperty.property_garage_id}", rank =
DB_Statements.main_subject_attributes[4].attribute_rank, type = "garage_id" });
        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[5].attribute_description} =
{my_sproperty.property_bedrooms_amount}", rank =
DB_Statements.main_subject_attributes[5].attribute_rank, type =
"property_bedrooms_amount" });
        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[6].attribute_description} =
{my_sproperty.property_bathrooms_amount}", rank =
DB_Statements.main_subject_attributes[6].attribute_rank, type =
"property_bathrooms_amount" });
        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[7].attribute_description} =
{my_sproperty.out_building_id}", rank =
DB_Statements.main_subject_attributes[7].attribute_rank, type = "out_building_id" });
        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[8].attribute_description} =
{my_sproperty.amenity_id}", rank =
DB_Statements.main_subject_attributes[8].attribute_rank, type = "amenity_id" });
        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[9].attribute_description} =
{my_sproperty.property_condition_id}", rank =
DB_Statements.main_subject_attributes[9].attribute_rank, type = "condition_id" });
        prop_crit_list.Add(new Prop_Crit { definition =
"${DB_Statements.main_subject_attributes[10].attribute_description} =
{my_sproperty.type_id}", rank = DB_Statements.main_subject_attributes[10].attribute_rank,
type = "type_id" });

        prop_crit_main_list.Add(new Prop_Crit { definition =
"$p.{DB_Statements.main_subject_attributes[0].attribute_description} =
{my_sproperty.suburb_id}", rank =
DB_Statements.main_subject_attributes[0].attribute_rank, type = "Suburb" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
"$p.{DB_Statements.main_subject_attributes[1].attribute_description} =
{my_sproperty.index_id}", rank = DB_Statements.main_subject_attributes[1].attribute_rank,
type = "Date of sale" });
        //prop_crit_main_list.Add(new Prop_Crit { definition =
"$p.{DB_Statements.main_subject_attributes[2].attribute_description} >=
{size_crit_lower}", rank = DB_Statements.main_subject_attributes[2].attribute_rank, type
= "Property Size" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
"$p.{DB_Statements.main_subject_attributes[3].attribute_description} =
{my_sproperty.flat_id}", rank = DB_Statements.main_subject_attributes[3].attribute_rank,
type = "Flat" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
"$p.{DB_Statements.main_subject_attributes[4].attribute_description} =
{my_sproperty.property_garage_id}", rank =
DB_Statements.main_subject_attributes[4].attribute_rank, type = "Garage" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
"$p.{DB_Statements.main_subject_attributes[5].attribute_description} =
{my_sproperty.property_bedrooms_amount}", rank =
DB_Statements.main_subject_attributes[5].attribute_rank, type = "Amount of bedrooms" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
"$p.{DB_Statements.main_subject_attributes[6].attribute_description} =
{my_sproperty.property_bathrooms_amount}", rank =
DB_Statements.main_subject_attributes[6].attribute_rank, type = "Amount of bathrooms" });

```

```

        prop_crit_main_list.Add(new Prop_Crit { definition =
$"p.{DB_Statements.main_subject_attributes[7].attribute_description} =
{my_sproperty.out_building_id}", rank =
DB_Statements.main_subject_attributes[7].attribute_rank, type = "Out buildings" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
$"p.{DB_Statements.main_subject_attributes[8].attribute_description} =
{my_sproperty.amenity_id}", rank =
DB_Statements.main_subject_attributes[8].attribute_rank, type = "Amenities" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
$"p.{DB_Statements.main_subject_attributes[9].attribute_description} =
{my_sproperty.property_condition_id}", rank =
DB_Statements.main_subject_attributes[9].attribute_rank, type = "Condition" });
        prop_crit_main_list.Add(new Prop_Crit { definition =
$"p.{DB_Statements.main_subject_attributes[10].attribute_description} =
{my_sproperty.type_id}", rank = DB_Statements.main_subject_attributes[10].attribute_rank,
type = "Type" });

        prop_crit_list = prop_crit_list.OrderBy(o => o.rank).ToList();
        prop_crit_main_list = prop_crit_main_list.OrderBy(o => o.rank).ToList();
    }

    //This function compiles and returns the comparable property's selection query
    criteria (WHERE clause)
    public static string create_criteria()
    {
        string crit_return = "";
        for(int i = 0; i < prop_crit_list.Count; i++)
        {
            if(i == prop_crit_list.Count - 1)
            {
                crit_return += prop_crit_list[i].definition;
            }
            else
            {
                crit_return += prop_crit_list[i].definition + " AND ";
            }
        }

        return crit_return;
    }

    public static string create_main_criteria()
    {
        string crit_return = "";
        for (int i = 0; i < prop_crit_list.Count; i++)
        {
            if (i == prop_crit_main_list.Count - 1)
            {
                crit_return += prop_crit_main_list[i].definition;
            }
            else
            {
                crit_return += prop_crit_main_list[i].definition + " AND ";
            }
        }

        return crit_return;
    }
}

```

//This function retrieves comparable properties from the database, according to the previously compiled query criteria and subject property attributes

```

public static List<Property> get_comparable_properties(string criteria)
{
    property_counter = 0;
    List<Property> comparables_return = new List<Property>();

    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;
            OleDbCommand command = new OleDbCommand("SELECT " +
                "property_address, " +
                "suburb_id, " +
                "property_price, " +
                "date_of_sale, " +
                "index_id, " +
                "property_size, " +
                "flat_id, " +
                "property_garage_id, " +
                "property_bedrooms_amount, " +
                "property_bathrooms_amount, " +
                "out_building_id, " +
                "amenity_id, " +
                "property_condition_id, " +
                "type_id" +
                " FROM properties_table" +
                $" WHERE {criteria}", connection);
            reader = command.ExecuteReader();
            while (reader.Read())
            {
                comparables_return.Add(new Property {
                    property_address = reader[0].ToString(),
                    suburb_id = Convert.ToInt32(reader[1].ToString()),
                    property_price = Convert.ToDouble(reader[2].ToString()),
                    sale_date = Convert.ToDateTime(reader[3].ToString()),
                    index_id = Convert.ToInt32(reader[4].ToString()),
                    property_size = Convert.ToDouble(reader[5].ToString()),
                    flat_id = Convert.ToInt32(reader[6].ToString()),
                    property_garage_id = Convert.ToInt32(reader[7].ToString()),
                    property_bedrooms_amount =
Convert.ToInt32(reader[8].ToString()),
                    property_bathrooms_amount =
Convert.ToInt32(reader[9].ToString()),
                    out_building_id = Convert.ToInt32(reader[10].ToString()),
                    amenity_id = Convert.ToInt32(reader[11].ToString()),
                    property_condition_id =
Convert.ToInt32(reader[12].ToString()),
                    type_id = Convert.ToInt32(reader[13].ToString())
                });
                if (reader != null)
                {
                    property_counter++;
                }
            }
            connection.Close();
        }
    }
}

```

```

    }
    catch (Exception exc)
    {
        MessageBox.Show($"Something happened: \r\n\r\n {exc}");
    }
    connection.Close();
}
return comparables_return;
}

//cosmetic function to display the comparable properties without ID's
criteria)
public static List<Complete_Prop> get_complete_comparable_properties(string
{
    List<Complete_Prop> comparables_return = new List<Complete_Prop>();

    string connString = @"Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=pvDB.accdb";

    using (OleDbConnection connection = new OleDbConnection(connString))
    {
        connection.Open();
        try
        {
            OleDbDataReader reader = null;
            OleDbCommand command = new OleDbCommand("SELECT " +
                "p.property_address, " +
                "s.suburb_name," +
                "p.property_price," +
                "p.date_of_sale," +
                "p.index_id," +
                "p.property_size," +
                "f.flat_desc," +
                "g.garage_description," +
                "p.property_bedrooms_amount," +
                "p.property_bathrooms_amount," +
                "o.out_building_desc," +
                "a.amenity_desc," +
                "c.condition_description," +
                "t.type_desc" +
                " FROM " +
                "properties_table AS p, " +
                "suburb_table AS s, " +
                "flat_table AS f, " +
                "garage_table AS g, " +
                "out_building_table AS o, " +
                "amenity_table AS a," +
                "condition_table AS c," +
                "type_table AS t" +
                " WHERE " +
                "p.suburb_id = s.ID AND " +
                "p.flat_id = f.ID AND " +
                "p.property_garage_id = g.ID AND " +
                "p.out_building_id = o.ID AND " +
                "p.amenity_id = a.ID AND " +
                "p.property_condition_id = c.ID AND " +
                "p.type_id = t.ID AND " +
                $"{criteria}", connection);

            reader = command.ExecuteReader();
            while (reader.Read())
            {
                comparables_return.Add(new Complete_Prop

```

```

        {
            property_address = reader[0].ToString(),
            suburb_name = reader[1].ToString(),
            property_price = Convert.ToDouble(reader[2].ToString()),
            sale_date = Convert.ToDateTime(reader[3].ToString()),
            index_id = Convert.ToInt32(reader[4].ToString()),
            property_size = Convert.ToDouble(reader[5].ToString()),
            flat_description = reader[6].ToString(),
            property_garage_description = reader[7].ToString(),
            property_bedrooms_amount =
Convert.ToInt32(reader[8].ToString()),
            property_bathrooms_amount =
Convert.ToInt32(reader[9].ToString()),
            out_building_description = reader[10].ToString(),
            amenity_description = reader[11].ToString(),
            property_condition_description = reader[12].ToString(),
            type_description = reader[13].ToString()
        });
        if (reader != null)
        {
            property_counter++;
        }
    }
    connection.Close();
}
catch (Exception exc)
{
    MessageBox.Show($"Something happened: \r\n\r\n {exc}");
}
connection.Close();
}
return comparables_return;
}

```

//The following few functions, labled "_adjust" at the end, describe how each attribute is adjusted when needed.

```

public static double index_adjust(int sub_ind_id, int com_ind_id, bool adjust)
{
    List<Index> adjust_indexes = new List<Index>();
    adjust_indexes = populate_index_list();

    double ind_val_sub = 0;
    double ind_val_com = 0;

    for(int i = 0; i < adjust_indexes.Count; i++)
    {
        if(sub_ind_id == adjust_indexes[i].index_id)
        {
            ind_val_sub = adjust_indexes[i].index_value;
            break;
        }
    }

    for (int i = 0; i < adjust_indexes.Count; i++)
    {
        if (com_ind_id == adjust_indexes[i].index_id)
        {
            ind_val_com = adjust_indexes[i].index_value;
            break;
        }
    }
}

```

```

        if(adjust == true)
        {
            double adjustv = ind_val_sub / ind_val_com;
            main_adjustments.Add(new Adjustments {adjustment_name = "date of sale",
adjustment_value = adjustv });
            return adjustv;
        }
        else
        {
            return 1;
        }
    }

    public static double suburb_adjust(int s_sub_id, int c_sub_id, bool adjust)
    {
        List<Suburb> adjust_suburbs = new List<Suburb>();
        adjust_suburbs = populate_suburb_list();

        double s_sub_val = 0;
        double c_sub_val = 0;

        for (int i = 0; i < adjust_suburbs.Count; i++)
        {
            if (s_sub_id == adjust_suburbs[i].suburb_id)
            {
                s_sub_val = adjust_suburbs[i].suburb_average_price;
                Console.WriteLine($"{adjust_suburbs[i].suburb_name}'s average price
is {adjust_suburbs[i].suburb_average_price}");
                break;
            }
        }

        for (int i = 0; i < adjust_suburbs.Count; i++)
        {
            if (c_sub_id == adjust_suburbs[i].suburb_id)
            {
                c_sub_val = adjust_suburbs[i].suburb_average_price;
                Console.WriteLine($"{adjust_suburbs[i].suburb_name}'s average price
is {adjust_suburbs[i].suburb_average_price}");
                break;
            }
        }

        if (adjust == true)
        {
            main_adjustments.Add(new Adjustments { adjustment_name = "suburb",
adjustment_value = s_sub_val / c_sub_val });
            return s_sub_val / c_sub_val;
        }
        else
        {
            return 1;
        }
    }

    public static double flat_adjust(int s_flat_id, int c_flat_id, bool adjust)
    {
        List<Flat> adjust_flats = new List<Flat>();

```

```

adjust_flats = populate_flat_list();

double s_flat_val = 0;
double c_flat_val = 0;

for (int i = 0; i < adjust_flats.Count; i++)
{
    if (s_flat_id == adjust_flats[i].flat_id)
    {
        s_flat_val = adjust_flats[i].flat_adjustment;
        break;
    }
}

for (int i = 0; i < adjust_flats.Count; i++)
{
    if (c_flat_id == adjust_flats[i].flat_id)
    {
        c_flat_val = adjust_flats[i].flat_adjustment;
        break;
    }
}

if (adjust == true)
{
    main_adjustments.Add(new Adjustments { adjustment_name = "flat",
adjustment_value = s_flat_val - c_flat_val });
    return s_flat_val - c_flat_val;
}
else
{
    return 0;
}

}

public static double garage_adjust(int s_garage_id, int c_garage_id, bool adjust)
{
    List<Garage> adjust_garages = new List<Garage>();
    adjust_garages = populate_garage_list();

    double s_garage_val = 0;
    double c_garage_val = 0;

    for (int i = 0; i < adjust_garages.Count; i++)
    {
        if (s_garage_id == adjust_garages[i].garage_id)
        {
            s_garage_val = adjust_garages[i].garage_adjustment;
            break;
        }
    }

    for (int i = 0; i < adjust_garages.Count; i++)
    {
        if (c_garage_id == adjust_garages[i].garage_id)
        {
            c_garage_val = adjust_garages[i].garage_adjustment;
            break;
        }
    }
}

```

```

        if (adjust == true)
        {
            main_adjustments.Add(new Adjustments { adjustment_name = "garage",
adjustment_value = s_garage_val - c_garage_val });
            return s_garage_val - c_garage_val;
        }
        else
        {
            return 0;
        }
    }

    public static double out_building_adjust(int s_out_building_id, int c_out_building_id,
bool adjust)
    {
        List<Out_Buildings> adjust_out_buildings = new List<Out_Buildings>();
        adjust_out_buildings = populate_out_buildings_list();

        double s_out_building_val = 0;
        double c_out_building_val = 0;

        for (int i = 0; i < adjust_out_buildings.Count; i++)
        {
            if (s_out_building_id == adjust_out_buildings[i].out_building_id)
            {
                s_out_building_val = adjust_out_buildings[i].out_building_adjustment;
                break;
            }
        }

        for (int i = 0; i < adjust_out_buildings.Count; i++)
        {
            if (c_out_building_id == adjust_out_buildings[i].out_building_id)
            {
                c_out_building_val = adjust_out_buildings[i].out_building_adjustment;
                break;
            }
        }

        if (adjust == true)
        {
            main_adjustments.Add(new Adjustments { adjustment_name = "out_building",
adjustment_value = s_out_building_val - c_out_building_val });
            return s_out_building_val - c_out_building_val;
        }
        else
        {
            return 0;
        }
    }

    public static double amenities_adjust(int s_amenities_id, int c_amenities_id,
bool adjust)
    {
        List<Amenity> adjust_amenities = new List<Amenity>();
        adjust_amenities = populate_amenity_list();

        double s_amenities_val = 0;

```

```

        double c_amenities_val = 0;

        for (int i = 0; i < adjust_amenities.Count; i++)
        {
            if (s_amenities_id == adjust_amenities[i].amenity_id)
            {
                s_amenities_val = adjust_amenities[i].amenity_adjustment;
                break;
            }
        }

        for (int i = 0; i < adjust_amenities.Count; i++)
        {
            if (c_amenities_id == adjust_amenities[i].amenity_id)
            {
                c_amenities_val = adjust_amenities[i].amenity_adjustment;
                break;
            }
        }

        if (adjust == true)
        {
            main_adjustments.Add(new Adjustments { adjustment_name = "amenities",
adjustment_value = s_amenities_val - c_amenities_val });
            return s_amenities_val - c_amenities_val;
        }
        else
        {
            return 0;
        }
    }

    public static double condition_adjust(int s_condition_id, int c_condition_id,
bool adjust)
    {
        List<Condition> adjust_condition = new List<Condition>();
        adjust_condition = populate_condition_list();

        double s_condition_val = 0;
        double c_condition_val = 0;

        for (int i = 0; i < adjust_condition.Count; i++)
        {
            if (s_condition_id == adjust_condition[i].condition_id)
            {
                s_condition_val = adjust_condition[i].condition_modifier;
                break;
            }
        }

        for (int i = 0; i < adjust_condition.Count; i++)
        {
            if (c_condition_id == adjust_condition[i].condition_id)
            {
                c_condition_val = adjust_condition[i].condition_modifier;
                break;
            }
        }

        if (adjust == true)
        {

```

```

        main_adjustments.Add(new Adjustments { adjustment_name = "condition",
adjustment_value = 1 - (s_condition_val - c_condition_val) });
        return 1 - ((s_condition_val - c_condition_val)/100);
    }
    else
    {
        return 1;
    }
}

//Both the bedroom and bathroom adjustment values, represent the identified
adjustment values for both
public static double bedroom_adjust(int s_bed_am, int c_bed_am, bool adjust)
{
    if (adjust == true)
    {
        int actual_adjust = s_bed_am - c_bed_am;
        main_adjustments.Add(new Adjustments { adjustment_name = "bedrooms",
adjustment_value = actual_adjust * 100000 });
        return actual_adjust * 100000;
    }
    else
    {
        return 0;
    }
}

public static double bathroom_adjust(int s_bath_am, int c_bath_am, bool adjust)
{
    if (adjust == true)
    {
        int actual_adjust = s_bath_am - c_bath_am;
        main_adjustments.Add(new Adjustments { adjustment_name = "bathrooms",
adjustment_value = actual_adjust * 45000 });
        return actual_adjust * 45000;
    }
    else
    {
        return 0;
    }
}

public static double size_adjust(double s_size, double c_size, bool adjust)
{
    if (adjust == true)
    {
        double actual_adjust = 1;
        main_adjustments.Add(new Adjustments { adjustment_name = "property size",
adjustment_value = actual_adjust });
        return actual_adjust;
    }
    else
    {
        return 1;
    }
}

//This function evaluates the structural attribute adjustment lists and removes
them after they are adjusted for
public static void check_adjust_build(string desc, string desc2)

```

```

{
    if (main_adjustments.Count > 0)
    {
        for (int i = 0; i < main_adjustments.Count; i++)
        {
            if(desc == main_adjustments[i].adjustment_name)
            {
                if(main_adjustments[i].adjustment_value == 0)
                {
                    main_adjustments.RemoveAt(i);
                    for(int j = 0; j < crit_adjust_list.Count; j++)
                    {
                        if(desc2 == crit_adjust_list[j].type)
                        {
                            crit_adjust_list.RemoveAt(j);
                            break;
                        }
                    }
                    break;
                }
            }
        }
    }
}

```

//This function evaluates the umbrella attribute adjustment lists and removes them after they are adjusted for

```

public static void check_adjust_ext(string desc, string desc2)
{
    if (main_adjustments.Count > 0)
    {
        for (int i = 0; i < main_adjustments.Count; i++)
        {
            if (desc == main_adjustments[i].adjustment_name)
            {
                if (main_adjustments[i].adjustment_value == 1)
                {
                    main_adjustments.RemoveAt(i);
                    for (int j = 0; j < crit_adjust_list.Count; j++)
                    {
                        if (desc2 == crit_adjust_list[j].type)
                        {
                            crit_adjust_list.RemoveAt(j);
                            break;
                        }
                    }
                    break;
                }
            }
        }
    }
}

```

//This function, along with all the above adjustment functions, calculates a new porperty value for the subject property, according to the most comparable property

```

public static float calculate_value(List<Prop_Crit> adjust_list, Property
comparable_p, Property subject_p)
{
    float building_adjust = 0;
    float property_value = 0;
    float comparable_value = 0;

```

```

double bedroom_adjusts = 0;
double bathroom_adjusts = 0;
double garage_adjusts = 0;
double flat_adjusts = 0;
double out_building_adjusts = 0;
double amenity_adjusts = 0;
double suburb_adjusts = 0;
double index_adjusts = 0;
double condition_adjusts = 0;
double size_adjusts = 0;

bool main_adjust = false;
bool bedroom_t = false;
bool bathroom_t = false;
bool garage_t = false;
bool flat_t = false;
bool out_building_t = false;
bool amenity_t = false;
bool suburb_t = false;
bool index_t = false;
bool condition_t = false;
bool size_t = false;

comparable_value = float.Parse(comparable_p.property_price.ToString());
Console.WriteLine($"The comparable price is {comparable_value}");

if(adjust_list.Count > 0)
{
    main_adjust = true;
    for(int i = 0; i < adjust_list.Count; i++)
    {
        switch(adjust_list[i].type)
        {
            case "Suburb":
                suburb_t = true;
                break;
            case "Date of sale":
                index_t = true;
                break;
            case "Property Size":
                size_t = true;
                break;
            case "Flat":
                flat_t = true;
                break;
            case "Garage":
                garage_t = true;
                break;
            case "Amount of bedrooms":
                bedroom_t = true;
                break;
            case "Amount of bathrooms":
                bathroom_t = true;
                break;
            case "Out buildings":
                out_building_t = true;
                break;
            case "Amenities":
                amenity_t = true;
                break;
            case "Condition":
                condition_t = true;

```

```

        break;
    }
}

if (main_adjust == true)
{
    //Structural adjustments
    amenity_adjusts = amenities_adjust(subject_p.amenity_id,
comparable_p.amenity_id, amenity_t);
    check_adjust_build("amenities", "Amenities");
    Console.WriteLine(amenity_adjusts);

    flat_adjusts = flat_adjust(subject_p.flat_id, comparable_p.flat_id,
flat_t);
    check_adjust_build("flat", "Flat");
    Console.WriteLine(flat_adjusts);

    garage_adjusts = garage_adjust(subject_p.property_garage_id,
comparable_p.property_garage_id, garage_t);
    check_adjust_build("garage", "Garage");
    Console.WriteLine(garage_adjusts);

    bedroom_adjusts = bedroom_adjust(subject_p.property_bedrooms_amount,
comparable_p.property_bedrooms_amount, bedroom_t);
    check_adjust_build("bedrooms", "Amount of bedrooms");
    Console.WriteLine(bedroom_adjusts);

    bathroom_adjusts = bathroom_adjust(subject_p.property_bathrooms_amount,
comparable_p.property_bathrooms_amount, bathroom_t);
    check_adjust_build("bathrooms", "Amount of bathrooms");
    Console.WriteLine(bathroom_adjusts);

    out_building_adjusts = out_building_adjust(subject_p.out_building_id,
comparable_p.out_building_id, out_building_t);
    check_adjust_build("out_building", "Out buildings");
    Console.WriteLine(out_building_adjusts);

    //Total structural adjustments
    building_adjust = float.Parse((amenity_adjusts + flat_adjusts +
garage_adjusts + bedroom_adjusts + bathroom_adjusts + out_building_adjusts).ToString());
    Console.WriteLine(building_adjust);

    //Umbrella adjustments
    Console.WriteLine($"The index adjustment value is: {index_adjusts}\r\nThe
subject p index is: {subject_p.index_id}\r\nThe comparable p index is
{comparable_p.index_id}\r\nMust adjust: {index_t}");
    index_adjusts = index_adjust(subject_p.index_id, comparable_p.index_id,
index_t);
    check_adjust_ext("date of sale", "Date of sale");
    Console.WriteLine($"The index adjustment value is: {index_adjusts}");

    suburb_adjusts = suburb_adjust(subject_p.suburb_id,
comparable_p.suburb_id, suburb_t);
    check_adjust_ext("suburb", "Suburb");
    Console.WriteLine(suburb_adjusts);

    condition_adjusts = condition_adjust(subject_p.property_condition_id,
comparable_p.property_condition_id, condition_t);
    check_adjust_ext("condition", "Condition");
    Console.WriteLine(condition_adjusts);

    //This function is not used due to the property size being irrelevant

```

```

        size_adjusts = size_adjust(Convert.ToDouble(subject_p.property_size),
Convert.ToDouble(comparable_p.property_size), size_t);
        check_adjust_ext("property size", "Property Size");
        Console.WriteLine(size_adjusts);

        //The final property value calculations
        property_value = float.Parse((comparable_value +
building_adjust).ToString());
        Console.WriteLine($"The value is {property_value}");
        property_value *= float.Parse(condition_adjusts.ToString());
        Console.WriteLine($"The value is {property_value}");
        property_value = property_value * float.Parse(suburb_adjusts.ToString());
        Console.WriteLine($"The value is {property_value}");
        property_value *= float.Parse(index_adjusts.ToString());
        Console.WriteLine($"The value is {property_value}");

        return property_value;
    }
    else
    {
        return comparable_value;
    }
}
}

```

F.3. Represents the collection of page C# code that is utilized directly by the expert DSS's user interface that also executes the classes and functions discussed in F.1. and F.2.

F.3.1 The page code that is responsible for communicating with the system's regression action class to calculate the property attribute rankings, display it to the user and provides the option to store it within the database.

```

/// <summary>
/// Interaction logic for Attribute_Regression_Page.xaml
/// This page handles the regression functionality of the system with it's
communication with the python program
/// </summary>
///
public partial class Attribute_Regression_Page : Page
{

    public Attribute_Regression_Page()
    {
        InitializeComponent();
    }

    //This button event executes the python regression program, retrieves the results
and displays it on the front end
    private void btnRegAttr_Click(object sender, RoutedEventArgs e)
    {
        writeFiles();
        bool completed = false;
        regressRanks = new List<Attribute_Rank>();
        MessageBox.Show("Data successfully written. Starting regression.");
    }
}

```

```

try
{
    completed = runPythonScript("RegressAttributes.py");
    MessageBox.Show("Regression successful! Importing results");
}
catch(Exception exc)
{
    MessageBox.Show($"There was an error during regression: \r\n\r\n{exc}");
}

if(completed == true)
{
    string[] rResults = getRegressResults();
    regressRanks.Add(new Attribute_Rank { attribute_id = 1,
attribute_description = "suburb_id", attribute_rank = Convert.ToDouble(rResults[1]) });
    regressRanks.Add(new Attribute_Rank { attribute_id = 2,
attribute_description = "index_id", attribute_rank = Convert.ToDouble(rResults[2]) });
    regressRanks.Add(new Attribute_Rank { attribute_id = 3,
attribute_description = "flat_id", attribute_rank = Convert.ToDouble(rResults[4]) });
    regressRanks.Add(new Attribute_Rank { attribute_id = 4,
attribute_description = "property_garage_id", attribute_rank =
Convert.ToDouble(rResults[5]) });
    regressRanks.Add(new Attribute_Rank { attribute_id = 5,
attribute_description = "property_bedrooms_amount", attribute_rank =
Convert.ToDouble(rResults[6]) });
    regressRanks.Add(new Attribute_Rank { attribute_id = 6,
attribute_description = "property_bathrooms_amount", attribute_rank =
Convert.ToDouble(rResults[7]) });
    regressRanks.Add(new Attribute_Rank { attribute_id = 7,
attribute_description = "out_building_id", attribute_rank = Convert.ToDouble(rResults[8])
});
    regressRanks.Add(new Attribute_Rank { attribute_id = 8,
attribute_description = "amenity_id", attribute_rank = Convert.ToDouble(rResults[9]) });
    regressRanks.Add(new Attribute_Rank { attribute_id = 9,
attribute_description = "property_condition_id", attribute_rank =
Convert.ToDouble(rResults[10]) });

    dgRegress.ItemsSource = regressRanks;

    dgRegress.Loaded += (s, o) =>
    {
        Console.WriteLine("Editing headers....");
        dgRegress.Columns[0].Header = "Attribute ID Number";
        dgRegress.Columns[1].Header = "Attribute Name";
        dgRegress.Columns[2].Header = "Attribute Score";
    };

    foreach(var x in regressRanks)
    {
        Console.WriteLine(x.attribute_rank);
    }

    btnRegPost.Visibility = Visibility.Visible;
}
}

```

//This button allows the user to update the database's attribute rankings with the newly retrieved rankings

```
private void btnRegPost_Click(object sender, RoutedEventArgs e)
```

```

{
    List<double> regresslist = new List<double>();

    foreach (var x in regressRanks)
    {
        regresslist.Add(x.attribute_rank);
    }

    postNewRanks(regresslist);
}

private void btnBack_Click(object sender, RoutedEventArgs e)
{
    this.NavigationService.Navigate(new Page1());
}
}

```

F.3.2. The page class that executes the valuation process by executing the functions located within the *Property_price_calculation* class (F.2.4.).

```

/// <summary>
/// Interaction logic for New_Valuation_Page.xaml
/// This page is responsible for capturing the attributes values of the subject
property and executing the sales comparison method on it to calculate a new value, that
is displayed later on
/// </summary>
public partial class New_Valuation_Page : Page
{
    public New_Valuation_Page()
    {
        InitializeComponent();

        try
        {
            latIndex = getLatestIndexDB();
            combo_suburb = populate_suburb_list();
            combo_type = populate_type_list();
            combo_flat = populate_flat_list();
            combo_garage = populate_garage_list();
            combo_out_buildings = populate_out_buildings_list();
            combo_amenity = populate_amenity_list();
            combo_condition = populate_condition_list();
            combo_index = populate_index_list();
            main_subject_attributes = get_attributes();

            tbIndex.Text = $"{latIndex.index_value} for the month of
{checkIndMonthI(latIndex.index_date.Month)} of the year {latIndex.index_date.Year}";

            cbSuburb.ItemsSource = combo_suburb;
            cbType.ItemsSource = combo_type;
            cbFlat.ItemsSource = combo_flat;
            cbGarage.ItemsSource = combo_garage;
            cbOutBuild.ItemsSource = combo_out_buildings;
            cbAmenities.ItemsSource = combo_amenity;
            cbCondition.ItemsSource = combo_condition;

            tbBathrooms.Text = "2";
            tbBedrooms.Text = "3";
            tbSize.Text = "1200";
        }
        catch { }
    }
}

```

```

        cbAmenities.SelectedIndex = 0;
        cbCondition.SelectedIndex = 0;
        cbGarage.SelectedIndex = 2;
        cbFlat.SelectedIndex = 0;
        cbOutBuild.SelectedIndex = 0;
        cbSuburb.SelectedIndex = 0;
        cbType.SelectedIndex = 0;
    }
    catch(Exception exc)
    {
        System.Windows.MessageBox.Show($"Unable to retrieve latest
data:\r\n\r\n{exc}");
    }

}

public void display_props()
{
}

//This function, in conjunction with the webcrawl class, is responsible for retrieving
the latest online consumer price index (CPI)
public int search_index (DateTime pdate)
{
    int index_return_value = 0;
    bool found = false;
    for(int i = 0; i < combo_index.Count; i++)
    {
        if(pdate.Year == combo_index[i].index_date.Year && pdate.Month ==
combo_index[i].index_date.Month)
        {
            index_return_value = combo_index[i].index_id;
            found = true;
        }
    }
    if(found == false)
    {
        index_return_value = combo_index[combo_index.Count - 1].index_id;
    }
    return index_return_value;
}

//This function submits the attributes of the subject property and executes the
sales comparison method on it, by identifying comparable properties and calculating a new
value
private void btnSubmit_Click(object sender, RoutedEventArgs e)
{
    int index = search_index(dpValueDate.SelectedDate.Value);

    comparable_suburb = cbSuburb.SelectedItem.ToString();

    //Set the subject property, with its attributes
    subject_property = new Property
    {
        property_address = tbAdres.Text,
        suburb_id = Convert.ToInt32(cbSuburb.SelectedIndex + 1),
        sale_date = Convert.ToDateTime(dpValueDate.SelectedDate.Value),
        index_id = index,
        property_size = Convert.ToDouble(tbSize.Text),
        flat_id = Convert.ToInt32(cbFlat.SelectedIndex + 1),
        property_garage_id = Convert.ToInt32(cbGarage.SelectedIndex + 1),
        property_bedrooms_amount = Convert.ToInt32(tbBedrooms.Text),
    }
}

```

```

        property_bathrooms_amount = Convert.ToInt32(tbBathrooms.Text),
        out_building_id = Convert.ToInt32(cbOutBuild.SelectedIndex + 1),
        amenity_id = Convert.ToInt32(cbAmenities.SelectedIndex + 1),
        property_condition_id = Convert.ToInt32(cbCondition.SelectedIndex + 1),
        type_id = Convert.ToInt32(cbType.SelectedIndex + 1)
    };

    Complete_Prop comp_sub = new Complete_Prop
    {
        property_address = tbAdres.Text,
        suburb_name = cbSuburb.SelectedItem.ToString(),
        sale_date = Convert.ToDateTime(dpValueDate.SelectedDate.Value),
        index_id = index,
        property_size = Convert.ToDouble(tbSize.Text),
        flat_description = cbFlat.SelectedItem.ToString(),
        property_garage_description = cbGarage.SelectedItem.ToString(),
        property_bedrooms_amount = Convert.ToInt32(tbBedrooms.Text),
        property_bathrooms_amount = Convert.ToInt32(tbBathrooms.Text),
        out_building_description = cbOutBuild.SelectedItem.ToString(),
        amenity_description = cbAmenities.SelectedItem.ToString(),
        property_condition_description = cbCondition.SelectedItem.ToString(),
        type_description = cbType.SelectedItem.ToString()
    };

    comparable_properties.Clear();
    main_properties.Clear();
    main_adjustments.Clear();

    get_criteria(subject_property);
    string subject_criteria = create_criteria();

    //Get comparable properties. according to the subject property's values
    comparable_properties = get_comparable_properties(subject_criteria);

    Console.WriteLine($"Using the attributes " +
        $"{subject_criteria}");
    Console.WriteLine($"The program yielded {comparable_properties.Count}
results.");

    //Repeat the selection process, until at least 3 properties are retrieved
    while (comparable_properties.Count <= 3 && prop_crit_list.Count > 0)
    {
        comparable_properties.Clear();

        adjust_criteria();
        subject_criteria = create_criteria();
        comparable_properties = get_comparable_properties(subject_criteria);

        Console.WriteLine($"The attributes that were used is: " +
            $"{subject_criteria}");
    }

    if (comparable_properties.Count != 0)
    {
        Console.WriteLine("Results were found!");
        for (int i = 0; i < comparable_properties.Count; i++)
        {
            Console.WriteLine($"{ comparable_properties[i].property_address}, " +
                $"{ comparable_properties[i].suburb_id}, " +
                $"{ comparable_properties[i].property_price}, " +
                $"{ comparable_properties[i].sale_date}, " +

```

```

        $"{ comparable_properties[i].index_id}," +
        $"{ comparable_properties[i].property_size}," +
        $"{ comparable_properties[i].flat_id}," +
        $"{ comparable_properties[i].property_garage_id}," +
        $"{ comparable_properties[i].property_bedrooms_amount}," +
        $"{ comparable_properties[i].property_bathrooms_amount}," +
        $"{ comparable_properties[i].out_building_id}," +
        $"{ comparable_properties[i].amenity_id}," +
        $"{ comparable_properties[i].property_condition_id}," +
        $"{ comparable_properties[i].type_id} ");
    }
    Console.WriteLine($"The total results is:
{comparable_properties.Count}");
    subject_criteria = create_criteria();
    Console.WriteLine($"The attributes that were used is: " +
        $"{subject_criteria}");
    Console.WriteLine("No more adjustment needed");

    string main_crit = create_main_criteria();
    main_properties = get_complete_comparable_properties(main_crit);

    System.Windows.MessageBox.Show("Results were found");

    //Get the list of comparable properties down to 3
    main_properties = check_adjust_main_count(comparable_properties,
subject_property, main_properties);
    comparable_properties = check_adjust_count(comparable_properties,
subject_property);

    main_properties = main_properties.OrderBy(o =>
o.Adjustments).ThenByDescending(n => n.sale_date).ToList();
    if(main_properties.Count > 3)
    {
        while (main_properties.Count > 3)
        {
            main_properties.RemoveAt(main_properties.Count - 1);
        }
    }
    comparable_properties = comparable_properties.OrderBy(o =>
o.adjust_need).ThenByDescending(n => n.sale_date).ToList();
    if(comparable_properties.Count > 3)
    {
        while(comparable_properties.Count > 3)
        {
            comparable_properties.RemoveAt(comparable_properties.Count - 1);
        }
    }

    Console.WriteLine("Successfully checked the required adjustments");

    main_property_value = 0;
    Property comparable_property = comparable_properties[0];

    compare_adjust_list = get_adjust_list(subject_property,
comparable_properties[0]);

    //Calculate a new property value
    main_property_value = calculate_value(compare_adjust_list,
comparable_property, subject_property);
    Console.WriteLine($"The value of the property is:
{main_property_value}");

```

```

        this.NavigationService.Navigate(new Comparable_Page());
    }
}

private void btnBack_Click(object sender, RoutedEventArgs e)
{
    this.NavigationService.Navigate(new Page2());
}

//This button check online for the latest CPI value
private void btnCheckIndex_Click(object sender, RoutedEventArgs e)
{
    bool connect = false;
    try
    {
        using (var client = new WebClient())
        using (client.OpenRead("http://google.com/generate_204"))
        connect = true;
    }
    catch
    {
        connect = false;
    }

    if(connect == true)
    {
        string[] windex = { };
        latIndex = new Index();
        latIndex = getLatestIndexDB();

        DateTime indDate = new DateTime();

        try
        {
            windex = getLatestIndexWeb();

            string[] idate = windex[0].Split(' ');

            indDate = getIndexDate(idate[0], Convert.ToInt32(idate[1]));

            if (latIndex.index_date.ToString("MM/dd/yyyy") !=
indDate.ToString("MM/dd/yyyy"))
            {
                DialogResult dresult = System.Windows.Forms.MessageBox.Show($"The
latest index found online is: {windex[1]} for the month of
{checkIndMonthI(indDate.Month)}\r\n\r\nThe database is not up to date. Would you like to
update the database with this value?", "Online results", MessageBoxButtons.YesNo,
MessageBoxIcon.Information);
                if (dresult == DialogResult.Yes)
                {
                    postNewIndex(indDate, windex[1]);
                    latIndex = getLatestIndexDB();
                    tbIndex.Text = $"{latIndex.index_value} for the month of
{checkIndMonthI(latIndex.index_date.Month)} of the year {latIndex.index_date.Year}";
                }
            }
            else
            {
                System.Windows.Forms.MessageBox.Show($"The latest index found
online is: {windex[1]} for the month of {checkIndMonthI(indDate.Month)}\r\n\r\nThe

```

```

database contains the latest index value posted online. This value is going to be used
for any date later than this date.", "Online Results", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }
}
catch( Exception exc)
{
    System.Windows.Forms.MessageBox.Show("Something went wrong. Log:
\r\n\r\n" + exc);
}

}
else
{
    System.Windows.Forms.MessageBox.Show($"You seem to not be connected to
the internet. Please check your connection and try again.", "Connection error",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}

}
}

```

F.3.3. The page class that is responsible for displaying the comparable properties to the user.

```

/// <summary>
/// Interaction logic for Comparable_Page.xaml
/// This page is responsible for displaying the comparable properties of the subject
property
/// </summary>
public partial class Comparable_Page : Page
{
    public Comparable_Page()
    {
        InitializeComponent();

        dgComparables.ItemsSource = main_properties;

        //Set column header names
        dgComparables.Loaded += (s, o) =>
        {
            Console.WriteLine("The Number of columns is: " +
dgComparables.Columns.Count.ToString());
            if(dgComparables.Columns.Count == 15)
            {
                dgComparables.Columns[0].Header = "Adjusments";
                dgComparables.Columns[1].Header = "Property Address";
                dgComparables.Columns[2].Header = "Suburb";
                dgComparables.Columns[3].Header = "Property Price";
                dgComparables.Columns[4].Header = "Date of Sale";
                dgComparables.Columns[5].Header = "CPI Identification Number";
                dgComparables.Columns[6].Header = "Property Size";
                dgComparables.Columns[7].Header = "Flat";
                dgComparables.Columns[8].Header = "Garage";
                dgComparables.Columns[9].Header = "Number of Bedrooms";
                dgComparables.Columns[10].Header = "Number of Bathrooms";
                dgComparables.Columns[11].Header = "Domestic Quarters";
                dgComparables.Columns[12].Header = "Amenities";
                dgComparables.Columns[13].Header = "Condition";
            }
        }
    }
}

```

```

        dgComparables.Columns[14].Header = "Type";
    }
};

string adjustments_list = "";

if(compare_adjust_list.Count > 0)
{
    for(int i = 0; i < compare_adjust_list.Count; i++)
    {
        adjustments_list += compare_adjust_list[i].type + "\r\n";
    }
}

tbCompProp.Text = comparable_properties[0].property_address;
tbAdjustments.Text = adjustments_list;

Console.WriteLine($"The value of the property is:
{comparable_properties[0].property_price}");
}

private void btnNext_Click(object sender, RoutedEventArgs e)
{
    this.NavigationService.Navigate(new Results_Page());
}

private void Grid_Loaded(object sender, RoutedEventArgs e)
{
}

//This button allows a user to store the comparable properties to an external
text file for reporting purposes
private void btnSaveCompare_Click(object sender, RoutedEventArgs e)
{
    string contents = "";
    string desktopPath =
Environment.GetFolderPath(Environment.SpecialFolder.DesktopDirectory);
    desktopPath += "\\Comparables.txt";

    if (main_properties.Count >= 3)
    {
        for(int i = 0; i < 3; i++)
        {
            contents += $"{main_properties[i].property_address} || " +
                $"{main_properties[i].property_price} || " +
                $"{main_properties[i].sale_date} || " +
                $"{main_properties[i].suburb_name} || " +
                $"{main_properties[i].property_size} || " +
                $"{main_properties[i].property_garage_description} || " +
                $"{main_properties[i].property_bedrooms_amount} || " +
                $"{main_properties[i].property_bathrooms_amount} || " +
                $"{main_properties[i].property_condition_description} || " +
                $"{main_properties[i].out_building_description} || " +
                $"{main_properties[i].type_description} || " +
                $"{main_properties[i].flat_description} || " +
                $"{main_properties[i].amenity_description}\r\n";
        }
        File.WriteAllText(desktopPath, contents);
    }
}

```

```

        MessageBox.Show("Successfully saved records! Check desktop for a file
named Comparables.txt");
    }
    else if(main_properties.Count > 0)
    {
        foreach(var x in main_properties)
        {
            contents += $"{x.property_address} || " +
                $"{x.property_price} || " +
                $"{x.sale_date} || " +
                $"{x.suburb_name} || " +
                $"{x.property_size} || " +
                $"{x.property_garage_description} || " +
                $"{x.property_bedrooms_amount} || " +
                $"{x.property_bathrooms_amount} || " +
                $"{x.property_condition_description} || " +
                $"{x.out_building_description} || " +
                $"{x.type_description} || " +
                $"{x.flat_description} || " +
                $"{x.amenity_description}\r\n";
        }
        File.WriteAllText(desktopPath, contents);
        MessageBox.Show("Successfully saved records! Check desktop for a file
named Comparables.txt");
    }
}
}

```

F.3.4. The final page class that displays the newly calculated property value to the user.

```

/// <summary>
/// Interaction logic for Results_Page.xaml
/// This page displays the system's final results of the subject property's value to
the user
/// </summary>
public partial class Results_Page : Page
{
    public Results_Page()
    {
        InitializeComponent();

        tbFinalValue.Text = $"R {main_property_value}";

        tbComparable.Text = $"Comparable property value R
{comparable_properties[0].property_price}";

        tbAdjments.Text = "";
        if(main_adjustments.Count > 0)
        {
            for(int i = 0; i < main_adjustments.Count; i++)
            {
                tbAdjments.Text += $"{main_adjustments[i].adjustment_name}, with an
adjusted value of {main_adjustments[i].adjustment_value}\r\n";
            }
        }
        else
        {
            tbAdjments.Text = "No Adjustments were required";
        }
    }
}

```

```
    }  
}  
  
private void btnNext_Click(object sender, RoutedEventArgs e)  
{  
    this.NavigationService.Navigate(new Page1());  
}  
  
private void btnBack_Click(object sender, RoutedEventArgs e)  
{  
    this.NavigationService.Navigate(new Comparable_Page());  
}  
}
```

