DUT

DURBAN UNIVERSITY OF TECHNOLOGY
INYUVESI YASETHEKWINI YEZOBUCHWEPHESHE

FACULTY OF ACCOUNTING AND INFORMATICS

DEPARTMENT OF INFORMATION TECHNOLOGY

2023
YEAR END MAIN EXAMINATION

INSTRUCTIONAL PROGRAMME : Bachelor of ICT/Advanced Diploma in ICT
INSTRUCTIONAL OFFERING : Machine Intelligence 3
PAPER NUMBER : 1
SUBJECT CODE/S : MAIN301/MCHI301
DATE : 22 NOVEMBER 2023
DURATION : 3 HOURS
TIME : 14H00 - 17H00
TOTAL MARKS : 100
NUMBER OF PAGES : 11 (including cover sheet)
EXAMINER/S : MRS S HOOSEN
MODERATOR/S : MR L VORSTER

INSTRUCTIONS/REQUIREMENTS:-

1. Answer all questions in the answer book

2. Calculators are permitted

3. No notes will be permitted.

Do not turn the page until permission is given

QUESTION 1 [15 marks]

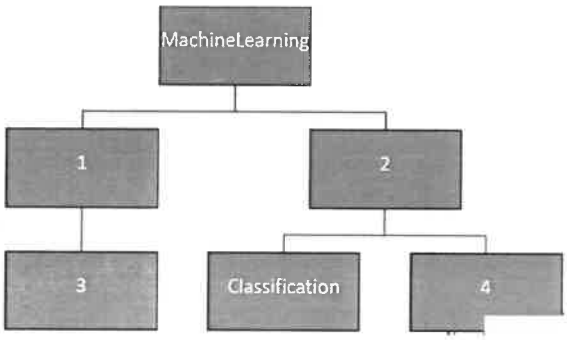
Write down the numbers 1.1 to 1.15 and next to each write down only the alphabet of the choice you have selected.

1.1 What type of learning needs to be applied to the following problem statement?

"You want to train a machine to help you predict how long it will take you to drive home from your workplace."

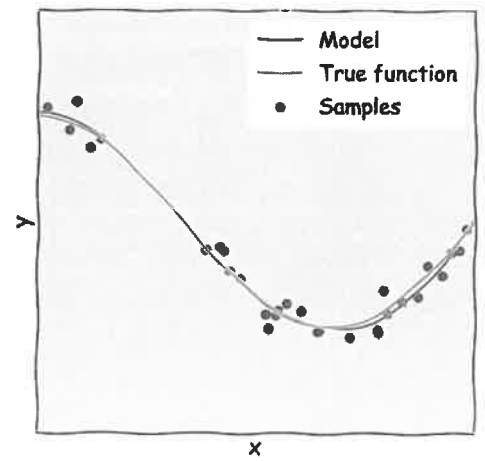
A) Unsupervised Learning with Clustering
B) Supervised Learning with Regression
C) Supervised Learning with Classification
D) Reinforcement Learning with Agents

1.2 The diagram below represents an overview of the types of machine learning. Study this diagram and then select the correct option by matching the numbers in the diagram with the corresponding type of learning.



A) 1: Supervised; 2: Unsupervised; 3: Regression; 4: Clustering
B) 1: Unsupervised; 2: Supervised; 3: Clustering; 4: Regression
C) 1: Supervised; 2: Unsupervised; 3: Clustering; 4: Regression
D) 1: Unsupervised; 2: Supervised; 3: Regression; 4: Clustering

1.3 What best describes the model function illustrated below?



A. This is an example of a model with high bias
B. The model function has the right complexity to fit the true function
C. The model function does not have enough complexity to fit the true function and underfits the model
D. The model function has too many complexities and over fits the model

1.4 In the snippet of code below which method compares features in a dataset and generates a threshold value for the comparisons, and which method locates individual elements in the dataset, respectively.

```
In [134]: def get_correlation(data,threshold):
              corr_column=set()
              corrmatrix=data.corr()
              for i in range(len(corrmatrix.columns)):
                  for j in range(i):
                      if abs(corrmatrix.iloc[i,j])>threshold:
                          colname=corrmatrix.columns[i]
                          corr_column.add(colname)
              return corr_column
```

A) corrmatrix, i
B) .corr(), set()
C) .corr(), .iloc[]
D) set(), iloc[]

1.5 This type of learning is used to identify different segments of customers and group them into categories like gender, age and location.

A) Reinforcement learning with agents
B) Unsupervised learning with clustering
C) Supervised learning with regression
D) Supervised learning with classification

1.6 The algorithm that finds the best fit model by automatically selecting the best features in a dataset and handles missing data is _____?

A) Neural Networks
B) SVM
C) Random Forest
D) Decision Trees

1.7

	A	B	C	D	E	F	G	H	I
	Type	Flour	Milk	Sugar	Butter	Egg	Baking Pov	Vanilla	Salt
	Muffin	55	28	3	7	5	2	0	0
	Muffin	47	24	12	6	9	1	0	0
	Muffin	47	23	18	6	4	1	0	0
	Muffin	45	11	17	17	8	1	0	0
	Muffin	50	25	12	6	5	2	1	0
	Muffin	55	27	3	7	5	2	1	0
	Muffin	54	27	7	5	5	2	0	0
	Muffin	47	26	10	10	4	1	0	0
	Muffin	50	17	17	8	6	1	0	0
	Muffin	50	17	17	11	4	1	0	0
	Cupcake	39	0	26	19	14	1	1	0
	Cupcake	42	21	16	10	8	3	0	0
	Cupcake	34	17	20	20	5	2	1	0
	Cupcake	39	13	17	19	10	1	1	0
	Cupcake	38	15	23	15	8	0	1	0
	Cupcake	42	18	25	9	5	1	0	0
	Cupcake	36	14	21	14	11	2	1	0
	Cupcake	38	15	31	8	6	1	1	0
	Cupcake	36	16	24	12	9	1	1	0
	Cupcake	34	17	23	11	13	0	1	0

x_train,x_test,y_train,y_test=train_test_split(X, y, test_size=0.3)

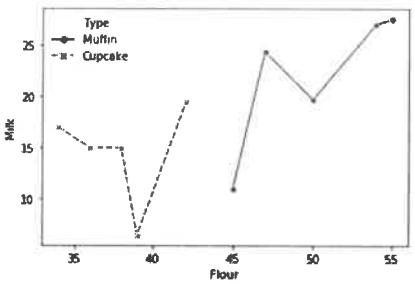
Given the dataset and the code above, what will be the output for x_train.shape, x_test.shape ?

A) (12, 8), (8,)
B) (14,8) (14,)
C) (14,) (6, 8)
D) (14, 8), (6, 8)

1.8 This module implements several loss, score, and utility functions to measure classification performance of algorithms when applied to data.
A) sklearn. scores
B) sklearn. metrics
C) sklearn.models
D) sklearn.accuracy

1.9 These types of machine learning algorithms are based on supervised learning and they model target prediction values based on independent variables. They mainly used for finding out the relationship between variables and forecasting.
A) Decision Trees
B) Random Forest
C) Linear Regression
D) SVM

1.10 What will be the correct coding with parameter listings for the plots below:



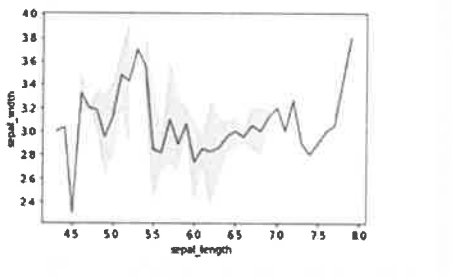
A) ax=sns.lineplot(x='Flour',y='Milk',data=raw_data,hue='Type', ci=False, markers=True, style='Type')
B) ax=sns.line(x='Flour',y='Milk',data=raw_data,hue='Type', ci=True, markers=True, style='Type')
C) ax=sns.lineplot(x='Flour',y='Milk',data=raw_data,hue='Type', ci='False', markers=True, 'style'='Type')
D) ax=sns.lineplot(x=Flour,y=Milk,data=raw_data,hue='Type', ci=False, markers=True, style='Type')

1.11 Given the following code:

```
import seaborn as sns

data = sns.load_dataset("iris")
```

Select the line of code that will plot the graph below:



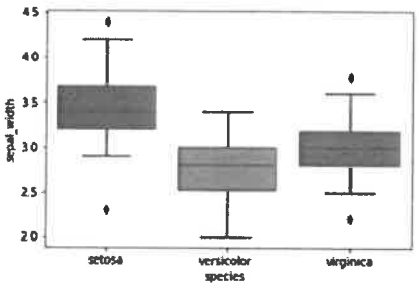
A. sns.lineplot(x="sepal_length", y="sepal_width", data=data, hue='Type')
B. sns.lineplot(x="sepal_length", y="sepal_width", data=data, ci="True")
C. sns.lineplot(x="sepal_width", y="sepal_length", data=data)
D. sns.lineplot(x="sepal_length", y="sepal_width", data=data)

1.12 Given the following code:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset("iris")
```

Select the code that will display the plot below:

Output:



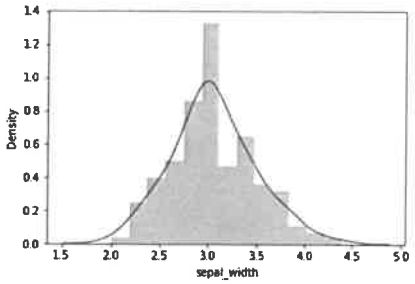
A) plt.boxplot(x='species', y='sepal_width', data=data)
plt.show()
B) sns.barplot(x=species, y=sepal_width, data=data)
plt.show()
C) sns.boxplot(x='species', y='sepal_width', data='data')
plt.show()
D) sns.boxplot(x='species', y='sepal_width', data=data)
plt.show()

1.13 Given the code below:

```
import seaborn as sns
import matplotlib.pyplot as plt
data = sns.load_dataset("iris")
```

Select the code that will display the plot below:

Output:



A) sns.histplot(data['sepal_width'])
B) sns.distplot(data['Density'])
C) sns.distplot(['sepal_width'])
D) sns.distplot(data['sepal_width'])

1.14 This is a type of bar plot where the X-axis represents the bin ranges while the Y-axis gives information about frequency.

A) Histogram
B) Line Graph
C) Scatter Plot
D) Bar chart

1.15 Colouring different points of a scatter plot can be done through the use of:

A) ci
B) markers
C) Hue
D) Matplotlib inline

(15)

QUESTION 2[20 marks]

2.1. Study the diagrams A and B below and and give a clear and detailed explanation of what each illustrates with regards to Machine Learning.

Diagram A

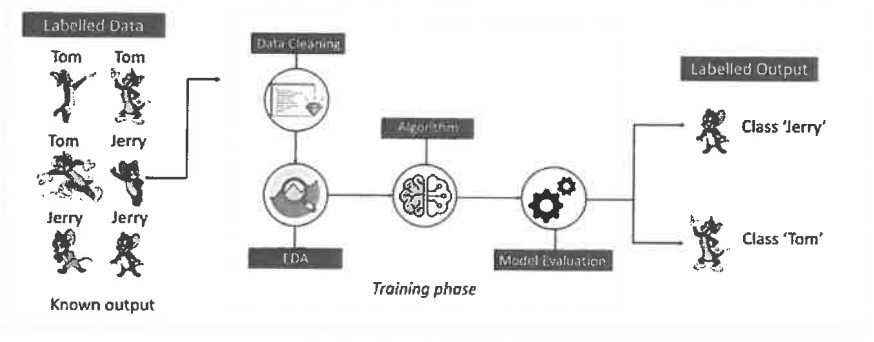
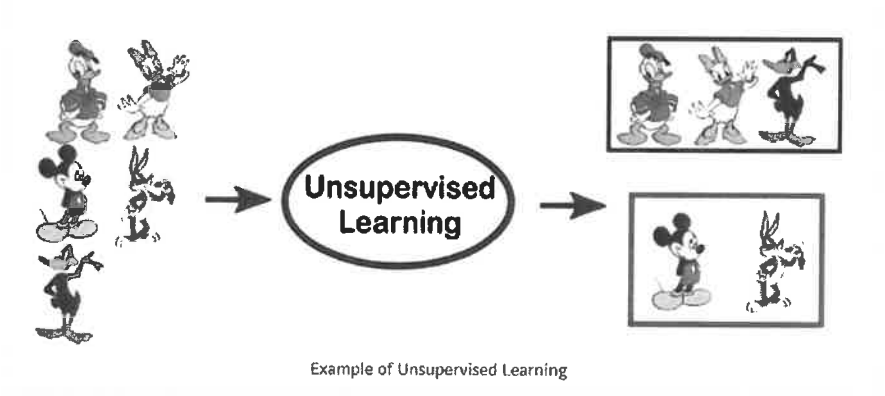


Diagram B



(6)

2.2 **"Ever since the technical revolution, we've been generating an immeasurable amount of data. As per research, we generate around 2.5 quintillion bytes of data every single day! It is estimated that by 2020, 1.7MB of data will be created every second for every person on earth."**

With regards to the statement above, discuss the need and importance of Machine Learning as the most in demand technology in today's market. (8)

2.3 Data Collection is regarded as a very important stage in the ML workflow process. From your understanding of Machine Learning why does data play such a vital role. (3)

2.4 "Data collected from the real world is transformed to a clean dataset."

List 3 ways on how one could clean the raw data in a dataset. (3)

QUESTION 3 [15 marks]

Study the dataset given below and answer the questions that follow:

Age	Height (in foot)	Weight (in kgs)
5	3.5	20
7	3.11	25
9	4.1	26
11	4.7	32
13	4.11	35
15	5.1	40
17	5.2	45
19	5.3	48
21	5.5	50
23	5.55	51
25	5.55	55

3.1 What can you conclude about the relationship between the 3 feature attributes given and explain the impact this may have on the ML model during the training phase of the ML workflow process. (5)

3.2 What statistical method can we implement in ML to help measure the strength of the relationship between 2 variables. (2)

3.3 With regards to ML, list 4 methods of cleaning a dataset and for each method explain what is the reason for removing these feature attributes. (8)

QUESTION 4 [10 marks]

Explain the working of each of the following ML algorithms using a diagram as well as the general Python code for implementation.

4.1 Linear Regression Algorithm (5)

4.2 Support Vector Machine (5)

QUESTION 5 [10 marks]

5.1 The code below implements a Decision Tree classifier to train and test a ML model using the muffins vs cupcakes dataset. Complete the code by filling in the missing information.

```
from __5.1.1___ import ____5.1.2____
model=____5.1.3_____
__5.1.4____(ingredients,type_label)
new =[[47, 26, 10, 10,  4,  1,  0,  0]]
predict=___5.1.5____(new)
if(predict)==0:
    print("muffin")
else:
    print("cupcake")
```

(5)

5.2 What is your understanding of the algorithm below. Exactly what is it used for in machine learning and how does it work. (5)

```
def get_correlation(data,threshold):
    corr_column=set()
    corrmatrix=data.corr()
    for i in range(len(corrmatrix.columns)):
        for j in range(i):
            if abs(corrmatrix.iloc[i,j])>threshold:
                colname=corrmatrix.columns[i]
                corr_column.add(colname)
    return corr_column
```

QUESTION 6 [15 marks]

6.1 Splitting your dataset is essential for an unbiased evaluation of prediction performance. The code below illustrates the application of the train, test, split method, with 60% of data used for training and 40% for testing. The program also uses evaluation measures of precision, accuracy and recall.

Study the incomplete program below and fill in the parts that are missing.

```
from sklearn.model_selection import ___6.1.1_____
from ___6.1.2_____ import accuracy_score,precision_score,___6.1.3____
from sklearn.ensemble import _____6.1.4_____
x_train,x_test,y_train,y_test=train_test_split(ingredients,type_label,
___6.1.5_=0.4)

model=RandomForestClassifier()
model.fit(x_train,____6.1.6____)
predict=model.predict(__6.1.7_____)
accuracy=____6.1.8_____(y_test,predict)
precision=____6.1.9_____(y_test,predict)
recall=   6.1.10   (y_test,predict)
accuracy,precision,recall
```

(10)

6.2 Splitting a dataset might also be important for detecting if your model suffers from one of two very common problems, called under fitting and overfitting. What is your understanding of each problem and explain how it affects the created model? (5)

QUESTION 7 [15 marks]

Study the following segment of Python code on data visualisations and answer the questions that follow.

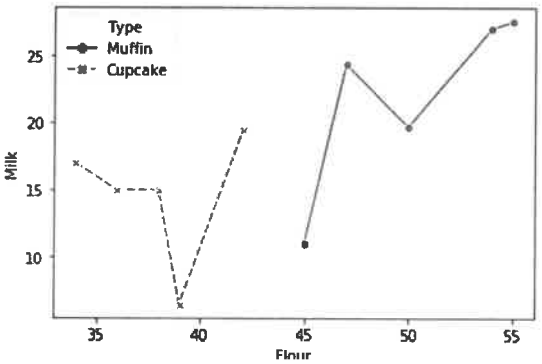
```
In [1]: #Plots and visualisations in ML
        import numpy as np
        import pandas as pd
        from matplotlib import pyplot as plt
        import seaborn as sns
        %matplotlib inline
```

```
In [3]: raw_data=pd.read_csv('muffins_cupcakes.csv')
        raw_data.head()
```

Out[3]:

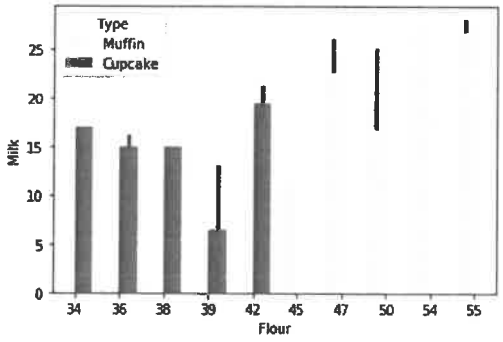
	Type	Flour	Milk	Sugar	Butter	Egg	Baking Powder	Vanilla	Salt
0	Muffin	55	28	3	7	5	2	0	0
1	Muffin	47	24	12	6	9	1	0	0
2	Muffin	47	23	18	6	4	1	0	0
3	Muffin	45	11	17	17	8	1	0	0
4	Muffin	50	25	12	6	5	2	1	0

7.1 Write the code that will display a lineplot as follows:



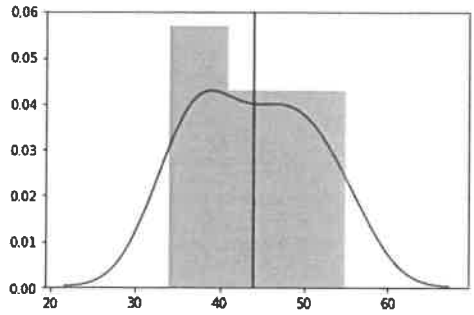
(5)

7.2 Write the code that will display a bar plot, in red, as follows:



(5)

7.3 Write the code that will display a histogram as follows. Note that the distribution plot is in red and the vertical mean axes line is in blue.



THE END

TOTAL=100

An architecture for user preference-based IoT service selection in cloud computing using mobile devices for smart campus

Submitted by

Lindelweyizizwe Siziwe Manqele

Supervised by

Prof Mqhele E. Dlodlo

Co-Supervised by

Dr Quentin Williams and Dr Louis Coetzee

This dissertation is submitted in partial fulfilment of the academic requirements

for the degree of

Master of Science in Electrical Engineering

in the Faculty of Engineering and The Built Environment

University of Cape Town.

June 2015

As the candidate's supervisor, I have approved this dissertation for submission

Professor Mqhele E. Dlodlo, Associate Professor

Signed: _____

Date: _____

As the candidate's supervisor, I have approved this dissertation for submission

Professor Mqhele E. Dlodlo, Associate Professor

Signed: Mqhele E. Dlodlo

Date: _____

Declaration

I declare that this dissertation is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.

This work is being submitted for the Master of Science in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

Lindelweyizizwe Siziwe Manqele	16/02/2015
Name	Date

To my family

Abstract

The Internet of things refers to the set of objects that have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social environments and user context. Interconnected devices communicating to each other or to other machines on the network have increased the number of services. The concepts of discovery, brokerage, selection and reliability are important in dynamic environments. These concepts have emerged as an important field distinguished from conventional distributed computing by its focus on large-scale resource sharing, delivery and innovative applications.

The usage of Internet of Things technology across different service provisioning environments has increased the challenges associated with service selection and discovery. Although a set of terms can be used to express requirements for the desired service, a more detailed and specific user interface would make it easy for the users to express their requirements using high-level constructs. In order to address the challenge of service selection and discovery, we developed an architecture that enables a representation of user preferences and manipulates relevant descriptions of available services.

To ensure that the key components of the architecture work, algorithms (content-based and collaborative filtering) derived from the architecture were proposed. The architecture was tested by selecting services using content-based as well as collaborative algorithms. The performances of the algorithms were evaluated using response time. Their effectiveness was evaluated using recall and precision. The results showed that the content-based recommender system is more effective than the collaborative filtering recommender system. Furthermore, the results showed that the content-based technique is more time-efficient than the collaborative filtering technique.

Acknowledgements

To God almighty who carried me through this journey, I thank Him. My gratitude also goes to:

THE CSIR: I thank my research group leader, Louis Coetzee, who supported my research technically and financially with supervision and his competent leadership. Further thanks go: to my research-group members: Laurie Butgereit, who assisted me with the database and proofreading my dissertation; Siveshnee Moonsamy, who assisted me with the setup of my application; Andrew Smith, who also proof read my dissertation; Dhiren Seetham, who assisted me with support throughout my research; Nomusa Dlodlo, who assisted me to shape the focus of my research; Quentin Williams, who assisted me with co-supervision. To my mentor, Ishmael Makitla, who helped me discover my capabilities; to my technical assistant George Sibiya, who has been with me throughout my research and as caring as my brother; to the HR manager, Antoinette Crafford for her support; to my netball team mates, with special thanks to: Thulile Khanyile, Chipo Gunda, Busi Twala and Masetjaba Tetsa, who have been with me through thick and thin, and always encouraged me.

TO THE UNIVERSITYOF CAPE TOWN: Massive thanks to my supervisor Prof Mqhele Dlodlo and his wife with their support, mentoring, supervision and fathering with so much wisdom that I hope I inherited. To my research-group members, thanks for their support and welcoming spirit. To the UCT postgraduate office, this gave me an opportunity to present my poster, thank you.

TO MY FAMILY: Thanks to my parents, Mr and Mrs Manqele, who allowed me to explore more in life, God bless you. To my kids who tolerated my long-distance parenting, thank you Mlondi and Ndumiso, God bless you. To my siblings (Nathi, Sandile, Minky, Sakhile, Sbahle, Khado, Zanele and Mfana) who have been there for me, thank you. To my friends, Nkanyiso Makhathini, Nonhlanhla Ntuli, Desmond, Kagiso, Celiwe, Londy, Zama Zungu, Promise, Mthobisi, Mncedisi, Martin and Tumi, for encouragement, God bless you. To the CFCI and WOP members and my mentor, Nonhlanhla Mkhwanazi, thank you and God bless you.

Table of Contents

Declaration iii

Abstract v

Acknowledgements vi

List of Figures xi

List of Tables xiv

Glossary xv

1.1 Introduction 1

1.2 Statement of the problem 2

1.3 Scope of research 3

1.3.1 Research questions *3*

1.4 Goal 4

1.5 Research objectives 4

1.6 Implication of research 5

1.7 Summary 6

1.8 Dissertation outline 7

2 Background 8

2.1 The Internet of Things 8

2.1.1 Definition of the Internet of Things *8*

2.1.2 Characteristics of the Internet of Things *11*

2.1.3 Application domains of the Internet of Things *14*

2.1.4 Challenges of the Internet of Things *16*

2.2 Smart Campus 18

2.2.1 Examples of smart-campus services *19*

2.2.2 Significance of the smart campus domain *22*

2.3 Cloud computing 22

2.3.1 Definition of cloud computing *23*

2.3.2 Characteristics of cloud computing *24*

2.3.3 Service model *26*

2.3.4 Deployment of cloud computing *27*

2.3.5 Significance of Cloud Computing *29*

2.4 Service-oriented architecture **30**
2.4.2 Service oriented architecture and reference model *32*
2.4.3 Benefits of SOA *34*
2.4.4 Challenges in SOA *37*
2.4.5 Integration of SOA into smart campus *38*
2.5 Recommender systems **39**
2.5.1 Personalisation *40*
2.5.2 Context-aware recommender systems *40*
2.5.3 Preference-based selection *41*
2.5.4 Classification of recommender systems *46*
2.6 Summary **48**
3 Literature Review **49**
3.1 Context **49**
3.1.1 Internet of Things services *51*
3.2 Service-selection architectures **53**
3.2.1 A QoS broker-based architecture *56*
3.2.2 The semantic broker-based web-service architecture *57*
3.2.3 Architecture of web-service selection frameworks *59*
3.2.4 QoS Broker Architecture *60*
3.2.5 Service-selection architecture *61*
3.2.6 Internet of Things system considerations *62*
3.2.7 Summary of the architectural analyses *63*
3.3 Service-selection approaches **64**
3.3.1 Service selection based on the multi-agent approach *66*
3.3.2 Service selection based on ontology *66*
3.3.3 Service selection based onQoS *68*
3.3.4 Service selection based on functional requirements *72*
3.3.5 Service selection based on user-centred QoS *72*
3.3.6 Internet of Things system considerations *73*
3.3.7 Summary of existing approaches *73*
3.4 Proposed approach **74**
3.5 Summary **78**
4 Architectural development and design **79**
4.1 Use case scenario **79**

4.1.1 Assumptions *79*
4.1.2 Selection criteria *81*
4.1.3 Proxy implements recommender systems *84*
4.2 Proposed architecture **85**
4.2.1 Application layer *87*
4.2.2 Middleware layer *88*
4.2.3 Resource layer *89*
4.3 Proposed ISSA algorithm **89**
4.3.1 User profile and service profile modelling *91*
4.3.2 Matchmaking *95*
4.4 Comparison of ISSA vs architectures in the literature **98**
4.5 Comparison of ISSA approach and existing approaches **100**
4.6 Summary **102**
5 ISSA experimental design and implementation **103**
5.1 ISSA experimental design **103**
5.1.1 ISSA use cases *103*
5.1.2 ISSA sequence diagram *109*
5.1.3 ISSA activity diagram *110*
5.1.4 ISSA entity relation diagram *111*
5.1.5 Summary *112*
5.2 ISSA implementation **113**
5.2.1 IoT service application *113*
5.2.2 Welcome page *114*
5.2.3 Create account *114*
5.2.4 Login *117*
5.2.5 List of recommended services *118*
5.2.6 Service invocation *119*
5.3 Summary **120**
6 ISSA evaluation and results analyses **121**
6.1 ISSA evaluation of the algorithm **121**
6.1.1 Effectiveness of the algorithm *121*
6.1.2 Performance of the algorithms *124*
6.2 ISSA results analysis **125**
6.2.1 Discussion *126*

6.2.2 Advantages *126*
6.2.3 Disadvantages *127*
6.2.4 Challenges *128*
6.3 Summary **128**
7 Conclusion **129**
7.1 Future recommendation **130**
Reference **131**
Appendix **140**
ISSA class diagram *140*

List of Figures

Figure 1: Research outline 7

Figure 2: The NIST definition of cloud computing (Mell and Grance 2011) 28

Figure 3: Service oriented architecture (Samreen 2011) 33

Figure 4: Classification of recommender systems (Manqele, Dlodlo, Adigunand Xulu, 2012) .. 47

Figure 5: Service-delivery lifecycle (Kuropka et al. 2008) 50

Figure 6: Overview of the IoT description model (Barnaghi *et al.*, 2013) 52

Figure 7: A QoS broker-based architecture (D'Mello et al. 2008) 57

Figure 8: The semantic broker-based web-service architecture (D'Mello and Ananthanarayana 2009) 58

Figure 9: Architecture of web service selection framework (Manikrao and Prabhakar, 2005).... 59

Figure 10: QoS broker architecture (Lin and Tao 2006) 60

Figure 11: Service-selection architecture (Guha 2009) 62

Figure 12: Selection criteria 82

Figure 13: SOA-based Selection criteria 84

Figure 14: Service requester submitting jobs 85

Figure 15:IoT service-selection architecture (ISSA) 88

Figure 16: User profile 92

Figure 17: IoT Service profile 93

Figure 18: Use case diagram 104

Figure 19: Sequence diagram 109

Figure 20: Activity diagram 111

Figure 21: Entity relation diagram 112

Figure 22: IoT Service Application 113
Figure 23: Login page 114
Figure 24: Create account 115
Figure 25: Preferred things 116
Figure 26: Validation 117
Figure 27: Login 118
Figure 28: List of recommended services 119
Figure 29: Service Invocation 120
Figure 30: Experimental framework 123
Figure 31: Recall vs precision for CB and CF 124
Figure 32 : Response-time evaluation 125
Figure 33: Class diagram 141
Figure 34: App install 142
Figure 35: Install app 143
Figure 36: Install 143
Figure 37: Installing 144
Figure 38: Open app 144
Figure 39: IoT services app 145
Figure 40: App welcoming page 146
Figure 41: App connection 147
Figure 42: Create account page 147
Figure 43: Registration modes 148
Figure 44:Things 149

Figure 45: Resources 149
Figure 46: Role-based mode 150
Figure 47: Login page in app 151
Figure 48: Wrong password 152
Figure 49: Password reminder 153
Figure 50: Correct password 153
Figure 51: Recommended list with recall 154
Figure 52: Recommended services 155
Figure 53: Edit account 156
Figure 54: Service invocation 157
Figure 55: Experiments 158
Figure 56: Values of recall and response time 158
Figure 57: Uninstall 1 159
Figure 58: Uninstall 2 160

List of Tables

Table 1: Classification of the benefits of SOA (Joachim, 2011) 35

Table 2: Summary of service selection architectures 63

Table 3: Summary of service-selection approaches 73

Table 4: Recommender-systems evaluation 76

Table 5: ISSA comparison 99

Table 6: ISSA algorithm comparison 100

Table 7: Create account 104

Table 8: Use case: Login 105

Table 9: Request service 106

Table 10: Supply available service 107

Table 11: View service 107

Table 12: Use case: invoke service 108

Table 13: Recall vs Precision data 123

Table 14: Response time data 124

Glossary

1. **Internet of Things (IoT)**: is an emerging paradigm that brings together people, machines and animate or inanimate objects, facilitating their interaction over the Internet communication infrastructure, using established Internet standards and technologies. In this paradigm, "things", both physical and virtual, have identities, attributes and intelligent interfaces that enable them to be "smart", and able to sense, connect, and communicate.
2. **Radio frequency identification (RFID)**is a major breakthrough in the embedded communication paradigm that enables the design of microchips for wireless data communication.
3. **General packet radio service (GPRS)** is a geographical navigator packet-oriented mobile data service on the second- and third-generation cellular communication global system for mobile communications.
4. **Service-oriented architecture (SOA)**is an architectural approach that integrates distributed and heterogeneous dynamic environments to perform the selection of a well-defined service that supports remote clients. SOA consists of a composite set of business-aligned services that support a flexible and dynamic environment where even complex services can be easily composed using individual services from various service providers. Those individual services can be selected, reused and integrated dynamically based on service functionalities and performance constraints.
5. **The Council for Scientific and Industrial Research (CSIR)** is an organisation that performs multidisciplinary research and technological innovation with the aim of contributing to both industrial development and the quality of life of the people of South Africa.
6. **Simple Object Access Protocol (SOAP)** is an Extensible Mark-up Language (XML) based protocol that facilitates, publishes, finds, binds, and invokes operations.
7. **Universal Description, Discovery and Integration (UDDI)** is a platform-independent registry commonly used in SOA and web services.
8. **The Web Services Description Language (WSDL)** works as transport protocol, message

formatter and locator and is commonly used today to implement service catalogues.

9. **Quality of Service (QoS)** is the overall performance of a computer network, particularly the performance seen by the users of the network.
10. **Cloud computing** is an elastic dynamic paradigm that integrates, stores and shares on-demand resources from different sources. It can be accessed anywhere and anytime through the connection of the Internet.
11. **The Platform-as-a-service (PaaS)** platform consists of infrastructure software, and typically includes a database, middleware and development tools in cloud computing.
12. **Infrastructure-as-a-service (IaaS)** is capability provided to the consumer to provision processing, storage, networks, and other fundamental computing resources in cloud computing.
13. **Elastic Compute Cloud (EC2)** is an example of a cloud computing platform that provides infrastructure as a service.
14. **Small and medium-sized enterprises (SMEs)** are small businesses.
15. **Service-Oriented Computing (SOC)** is the computing paradigm that utilises services as fundamental elements for developing applications and business solutions.
16. **RESTful service**is an architectural style that specifies constraints – such as the uniform interface – which, if applied to a web service, induce desirable properties such as good performance, scalability and modifiability that enable services to work best on the web.

Chapter 1: Introduction

1.1 Introduction

The Internet of things (IoT) appears to be the anticipated technology occurrence that will influence the future (Vermesan and Friess 2012). The idea of the IoT is creating a network of objects that can communicate with each other through sensors, radio-frequency identification (RFID) techniques, a general packet radio service(GPRS), computers, actuators, mobile phones, etc. The IoT involves an increasing number of smart interconnected devices and sensors (e.g. cameras, biometric, smart meter, and medical sensors) for smart environments. IoT is not dependent on developing new technology but on connecting and integrating existing resources. The IoT paradigm has its characteristics, application and key research issues which are regarded as challenges -- as discussed later in Section 2.1. Therefore, in order for the IoT to realise its potential as a revolutionary Internet technology, a number of open research challenges need to be addressed. Some of the key research issues in IoT include scalability, standardisation, heterogeneity, identification, and addressing objects. The open research challenges are discussed in detail in Section 2.1.4. In view of the foregoing, the scope of this researchis confined to addressing the research challenges relating to heterogeneity.

As the number of IoT-enabled devices on the network increases, so do the numbers of potential IoT services provided by these devices. This increase in the number of services compounds the complexity of accurate service selection in dynamic environments such as the IoT.Interoperability at the scale of the IoT goes beyond syntactical interfaces and requires the sharing of common semantics across all software architectures (Stonebraker, Brown and Martin, 1998). This work therefore raises the need of the key components of the architecture that can be used to select IoT services put together within a smart campus. The *smart campus* refers to the normal campus that offers the smart services. According to the focus of this work, smart services are IoT-based services such as switching on/off an air conditioner using a mobile device, whether the user is inside or outside the campus. The key components should enable the architecture to pre-select the services using intelligent computer-based mechanisms that represent user preferences and manipulate relevant service descriptions of available services.

This enables the user to discover, select and ultimately use the service, despite the multitude of options available.

Manikrao and Prabhakar (2005) realised that the use of semantics is one of the mechanisms to differentiate services with closer properties, capabilities and interfaces. However, the mapping performed by the system should be able to deal with user preferences and offer a service to the user. Little work has been done so far in the development of some preference-based service recommendation systems for IoT service selection. Hence, this work will adapt an approach that emanates from preference-based recommender systems. Recommender systems have been used mostly in e-commerce applications, search engines, web services, retrieval systems and online shopping. Although recommender systems have been criticised for not being suitable for service selection because they do not allow a customised schema in terms of the qualities of interest to different users and to data (Maximilien 2004). Balke and Wagner (2003) showed that selection depends not only on service parameters such as execution costs or accuracy, but also on the usefulness of objects or information that a service offers.

Furthermore, Serbanati, Maria and Biader (2011) stated that the IoT builds upon the existing Internet communication infrastructure and foresees a world permeated with embedded smart devices interconnected through this infrastructure. This makes the IoT an encompassing vision of integrating technologies and the real world (people, things and machines) into the Internet. However, this work will adopt service-oriented architecture (SOA) as the underlying structure supporting distributed and loosely coupled applications in such a way that they comprise discrete software agents that have simple, well-defined interfaces and are orchestrated through a loose coupling to perform a required function. The environments of the IoT demand more elastic and centralised storage like cloud computing.

1.2 Statement of the problem

According to the vision of the IoT, devices on the network are quickly growing to trillions of small-embedded devices in the physical world, and that increases challenges associated with

selecting a suitable service (Laine 2012).This research envisages an approach that can be used to select services in complex IoT environments.

1.3 Scope of research

The complexity of selecting a relevant service in the IoT environment raises challenges associated with the selection, discovery and composition of services. In view of the foregoing, the scope of this research is confined to addressing the research challenges relating to heterogeneity, as outlined next in Section 2.1.4.

1.3.1 Research questions

1. What are the key components of an architecture that can be used for integrated IoT service selection adopting user preferences in a smart-campus environment to support dynamic service composition?

   This question seeks to establish the main building blocks or components of an architecture that is capable of supporting dynamic service selection adopting user preferences. It is expected that these components may already exist, for instance in the service-oriented computing domain (i.e. web services) and may need to be repurposed for IoT service selection.

2. Having established the key components necessary for an architecture to support dynamic selection of the IoT service, what preference-based algorithm that integrates user preferences works best?

   This question is aimed at determining the best-performing and effective service-selection approach. In order to answer this question satisfactorily it is necessary to do a performance comparative analysis of recommender-systems-based service-selection approaches.

1.4 Goal

The goal of this study is to develop a preference-based architecture that will enable the selection of IoT services within a cloud-based smart campus through recommender systems using mobile devices.

1.5 Research objectives

1. To survey literature on IoT, service-oriented architectures, cloud computing, recommender systems and smart campus, and synthesise each paradigm according to the purpose of this work. Different service-recommendation approaches will be critically and comparatively evaluated using the IoT system consideration from a theoretical point of view.

2. To analyse the literature in architectures that support user-preference-based service selection and recommender algorithms. The analysis should result in classification of existing approaches in prevailing fields.

3. Using findings acquired from the literature in (1), the IoT system considerations, and the scenario, an architecture that supports IoT service selection will be developed. This architecture will form the basis of our implementation. A prototype of a proposed architecture capturing the pertinent components needed for IoT service selection in the architecture will be implemented.

4. To comparatively evaluate and analyse some existing recommender algorithms found to be applicable in IoT services selection from the theoretical critical and comparative analysis carried out in (2). The evaluation will be based on performance of the algorithms. Response time, recall and precision are the parameters to be used in this work.

1.6 Implication of research

This section discusses the significance of using each paradigm and the contribution to the organisation that provided its sample data to be used in this work. These paradigms are: cloud computing, smart campus, recommender systems and the SOA. The use of cloud computing emanates from the idea that cloud applications use large data centres and powerful servers that host web applications and web services. The cloud is cost-effective for both service providers and service consumers when accessing services on the Internet. The cloud provides an affordable information and communications technology (ICT) for a smart campus that can rapidly disseminate data across a campus. The services provided by a cloud are not limited by computer specification, except that devices need Internet connections. Some services are provided with their own software stored in a cloud, which makes it easy for a user with limited software resources to access services on a cloud. Those services support ubiquitous and dynamic environments.

The use of smart environments is becoming a key component of the IoT by improving the quality of life through introducing smart devices that are highly automated, reducing human intervention to a minimum (Dutta and Bilbao-Osorio 2012). Smart interacting objects adapt to the current situation without any human involvement. Services are context-aware in the IoT.

The use of recommender systems will be based on the selection criteria to be identified in this work; an approach for service selection should be able to handle preferences as soft constraints and semantically understand a request from a user. To do this, an approach-based recommender algorithm that satisfies the needs of the user will be identified. This application may be accessed using mobile devices, but it is developed as a web application. Selections can be extended to support the community, public, or the combination of community and public, cloud called a hybrid cloud. Use of cloud computing means dependence on others and that could possibly limit flexibility and innovation.

Extensibility makes SOA solutions available in all sizes of organisations. SOA is found relevant to integrate and gather data from different units in one place (Kimpim 2014). For example, the Council for Scientific and Industrial Research (CSIR) is an organisation that performs multidisciplinary research and technological innovation with the aim of contributing to industrial development and the quality of life of the people of South Africa. This work is contributing in the smart-world domain by identifying and implementing an application that can be used to select services composed within a smart campus (i.e. experimental facilities being developed for the CSIR campus). The mechanism should be compatible with mobile devices so that employees may access services inside or outside the premises of the CSIR campus.

1.7 Summary

The dissertation outline shown in Figure 1below presents the research concepts addressed in this dissertation. The focus of this work is on the IoT paradigms. Although there are concepts such as characteristics of the IoT, benefits of the IoT and application of the IoT, this work assumes that other components are functioning normally and address the challenges of the IoT. The challenges of the IoT are scalability, standardisation, identification and addressing objects, heterogeneity and security. This work assumes that other challenges' components are ideal, not a challenge and address the challenges in heterogeneity. Heterogeneity is the integration of technologies in IoT that exist in different or across hardware, architectures and infrastructures, as well as the technologies of mobile devices, clouds, and wireless networks. The challenges of integrating technologies are associated with service composition, service discovery and service selection. This work assumes that service composition and service discovery challenges are ideal working and address the research issues associated with service selection. Using wisdom acquired from the literature, this work later proposes IoT service selection architecture (ISSA)

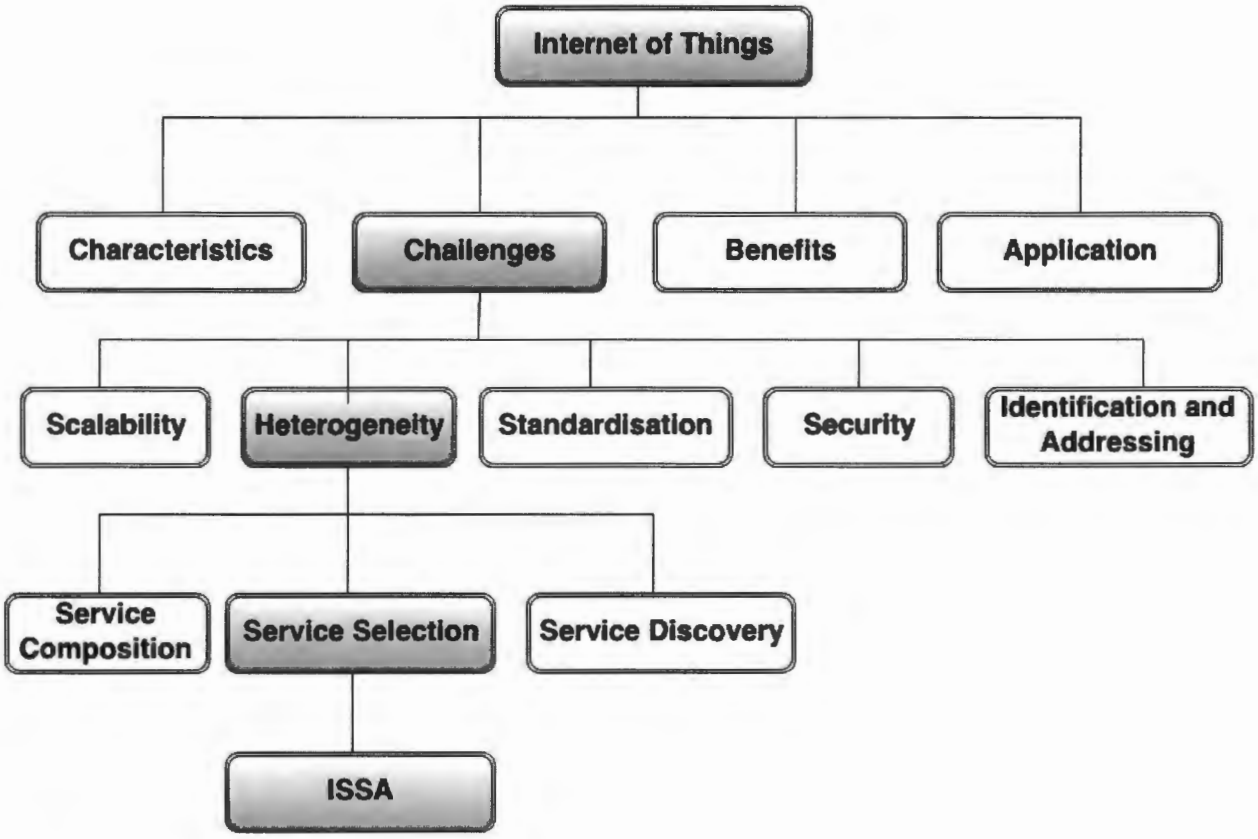


Figure 1: Research outline

1.8 Dissertation outline

This dissertation is organised as follows.

Chapter 1 introduces the concept and the focus of this work. Chapter 2 presents the background across the following paradigms: the Internet of Things, smart campus, cloud computing, service-oriented architecture and recommender systems. Chapter 3 presents the critical analyses on existing scholarship in approaches and architectures that support user preference based service selection and recommender algorithms. Chapter 4 presents the development architecture and algorithm that will select IoT services. Chapter 5 presents Experimental design and implementation. Chapter 6 presents the evaluation and results analysis. Chapter 7 presents the conclusion and future recommendation(s).

2 Background

This chapter presents the survey literature on the aspects relevant to the research problem presented in the previous chapter and synthesises each paradigm according to the purpose of this work. These aspects are Internet of Things, smart campus, cloud computing, service-oriented architectures and recommender systems. This chapter is structured as follows: Section 2.1 discusses the IoT, which is an emerging computing paradigm aimed at connecting intelligent objects to the Internet. Section 2.2 discusses the smart campus and presents the different types of service that can be provided by the smart campus. Section 2.3 discusses cloud computing as the essential storage paradigm used to store smart-campus services and make computing resources available on demand. Section 2.4 discusses SOA that concerns the architectural style for building loosely coupled and Internet-scaled distributed applications. Section 2.5 discusses user-preference algorithms that emanate from the recommender systems.

2.1 The Internet of Things

This section defines the concept of the IoT. Definitions of IoT are presented in Section 2.1.1, which draws from the existing body of knowledge and research literature on the subject of IoT.

2.1.1 Definition of the Internet of Things

The majority of objects will soon have communication and computation capabilities enabling them to connect, interact and cooperate with their surroundings and with other objects, including human beings (Dlodlo, Foko, Mvelase and Mathaba, 2012). These objects are referred to as "Things" and may take part in Thing-to-Thing, Thing-to-Person and Person-to-Thing interactions, which are supported by a number of Internet technologies and standards. The resulting phenomenon, where "Things" are able to communicate, or are accessible, over an Internet connection forms the necessary basis for the IoT (Uckelmann, Harrison and Michahelles, 2011).The vision of IoT is a global network infrastructure that integrates physical and virtual objects by exploiting the data-capture and communication capabilities (Lange, 2008).

Kevin Ashton is credited with coining the phrase "Internet of Things". In the journal by (Kevin 2010), Ashton suggested that there is great value in empowering computers with means of gathering data and knowledge about the environment without relying on human intervention. He argues further that the full potential of the future Internet will be restricted by human limitations (examples of anticipated human limitations are limited time, attention and accuracy) and hence the need to connect smart "things". The IoT is an emerging paradigm and, as such, as of this writing, there is no consensus on a standardised definition of the Internet of Things. Some of the most common definitions of Internet of Things are captured in this section, which gives a cursory yet useful summary of the contemporary understanding and views of the Internet of Things phenomenon (Revell (2013); Dlodlo,Foko, Mvelaseand Mathaba (2012); Atzori, Iera and Morabito (2010); Gubbi *et al*. (2013); Serbanati, Maria and Biader(2011). In the absence of any standard definition, this section reviews a number of existing definitions of the Internet of Things with an aim of forming a working definition of the Internet of Things for the purpose of this study.

The IoT special report group (Revell, 2013), defines the Internet of Things as the evolutionary state of Internet computing where "Things" (objects, vehicles, factory machinery and/or the environment) are augmented with useful information and are able to participate actively in Internet-mediated interactions, being able to sense, communicate and collaborate. Until recently with the advent of the Internet of Things, these Internet-mediated activities have been exclusively human-to-machine interactions. In its action plan for the Internet of Things, the European Commission views it as an evolution of the Internet from a global network of interconnected computers to a global network of interconnected objects (Atzori, Iera and Morabito, 2010). Furthermore, the Internet of Things represents views towards a dynamic and self-configuring global network infrastructure that is standards based and supported by interoperable communication protocols. Connected to this global network, according to the IoT strategic research roadmap (Vermesan *et al*., 2011) are physical and virtual autonomous "Things". This makes the Internet of Things a global network infrastructure that connects smart objects, European Lighthouse Integrated Project (Gubbi, Buyya, Marusic and Palaniswami,

2013). These "Things" have identities, physical attributes and virtual personalities as well as intelligent interfaces that enable them to seamlessly integrate into the information network. These intelligent interfaces also make the object controllable via the Internet and other means of connectivity such as RFID, a wireless Local Area Network(LAN) and Wide Area Network (WAN) (Badkas *et al.* 2010) .

Atzori *et al.* (2010) decomposes the name "Internet of Things" into its two key terms, the "Internet" and the "Things". The "Internet" pertains to the network orientation and suggests the Internet-like structure as the operating environment, whereas the "Things" relates to objects being integrated into a common framework (Atzori *et al.*, 2010). This composition implies that heterogeneous systems and technologies are involved.

According to Dlodlo, Foko, Mvelase and Mathaba(2012) the IoT is about objects communicating with each other over an Internet connection. According to this view, the idea of the Internet of Things is to create a network of uniquely addressable and identifiable objects that can communicate with each other using their different communication capabilities. The IoT is shifting web applications and services concepts towards wider integration and accessibility to enable the inter-networking paradigm of "anything, any time, and anywhere". Furthermore, the future Internet builds on such web applications and services to make up a dynamic entity, yielding novel means of interaction between services, users and the environment (Dlodlo *et al.*, 2012).

In providing the historical perspective of the IoT, Serbanati, Maria andBiader (2011) state that the IoT builds upon the existing Internet communication infrastructure and foresees a world permeated with embedded smart devices interconnected through this infrastructure. This makes the IoT an encompassing vision of integrating the real world (people, things and machines) into the Internet. Based on the preceding definitions of the IoT, and by literature synthesis, a working definition is arrived at. This study defines the Internet of Things as follows:

The Internet of Things is an emerging paradigm that brings together people, machines and inanimate or animate objects, facilitating their interactions over the Internet communication infrastructure, using established Internet standards and technologies. In this paradigm, "things", both physical and virtual, have identities, attributes and intelligent interfaces that enable them to be "smart" – able to sense, compute, connect, and communicate.

Having formulated a working definition of IoT, the next section describes common characteristics of the Internet of Things.

2.1.2 Characteristics of the Internet of Things

This section discusses the characteristics that make it possible to define something as the Internet of Things (IoT). These characteristics appear in IoT definitions in Section 2.1. In order to realise the IoT paradigm, the following characteristics must be developed and integrated in or on top of the IoT network infrastructure. Having the following characteristics, the IoT will transform the network to an infrastructure that is capable of providing global services to interact with the physical world (Serbanati *et al.*, 2011).

The key characteristics of the IoT are as follows:

- *Autonomy* - Gubbi *et al.* (2013) sees real-world agents in the IoT expressing their capabilities and needs autonomously, and getting those needs met to the benefit of us all. Internet of Things (2020), Vermesan *et al.*, (2011) and Lange (2008) concur that the IoT is characterised by a high degree of autonomous data capture, event transfer, network connectivity and interoperability. Autonomy means that intelligent devices should have capabilities such as context awareness and inter-machine communication. Intelligent devices lead to smart connectivity with existing networks and context-aware computation, using network resources as part of the IoT (Gubbi *et al.*, 2013).

- *Standard-based protocol* - Su *et al.* (2011) defines IoT as representative views towards a dynamic and self-configuring global network infrastructure that is standard-based and supported by interoperable communication protocols. Open standards will be among the

key enablers of the IoT. Fully global, energy-efficient communication standards that are security- and privacy-centred, and which use compatible or identical protocols at different frequencies, are needed. Future tags will need to integrate different communication standards and protocols that operate at different frequencies and allow different architectures – centralised or distributed – and which are able to communicate with other networks unless global, well-defined standards emerge.

- *Identification*- is crucial for public administration. Every object must have its unique IP address for identification on the network. Having the capacity of addressing each other and verifying their identities, all these objects will be able to exchange information and, if necessary, actively process information according to predefined schemes which may or may not be deterministic (Atzori *et al.*, 2010).

- *Network-orientation* - the very nature of the IoT is a network-oriented paradigm connecting smart objects to a global network. New, smart multi-frequency band antennas, integrated on-chip and made of new materials are the communication means that will enable the devices to communicate (Dutton 2005).

- *Integration* – the integration of smart devices into packaging or, better, into the products themselves, allows a significant cost saving and increases the eco-friendliness of the products (Lange 2008). The idea of integration improves performance and reduces cost. For example, Radio-frequency identification (RFID) inlays with strap-coupling structures are used to connect the integrated circuit chip and antenna in order to produce a variety of shapes and sizes of labels instead of direct mounting. In the context of this study, smart-campus services need to be integrated into a package to significantly save costs and increase the user-friendliness of services (Lange, 2008).

- *Scalability* - this means that the IoT offers measurable processing and storage capacity. Scalability becomes of interest in the IoT in order to better cope with dynamically scalable data streams (Vermesan *et al.*, 2011)

- *Energy-efficiency* - issues such as energy harvesting and low-power chip-sets are central to the development of the IoT, having as an objective ultra-low power devices, as current

devices seem inadequate considering the processing power needed and energy limitations of the future. Vermesan *et al.* (2011) suggested that "More Than Moore's" technologies, which focus on system integration, will increase the efficiency of current systems, and will provide a number of solutions for the future needs of the IoT.

2.1.2.1 Building blocks of the IoT

- *RFID* - the IoT consists of radio-frequency identification (RFID), sensor networks and web standards and protocols as the building blocks. RFID technology is a major breakthrough in the embedded-communication paradigm that enables the design of microchips for wireless data communication. RFID tags help in automatic identification of anything attached, acting as electronic barcodes. The passive RFID tags are not battery powered and they use the power of the reader's interrogation signal to communicate the ID to the RFID reader (Gubbi *et al.*, 2013). The applications can be in transportation (replacement of tickets, registration stickers) and access-control applications as well. The passive tags are currently used in many bank cards and road-toll tags are among the first global deployments. The integration of communication capabilities between RFID tags, sensors and actuators is seen as a very important area that needs to be studied, together with the integration of such devices into hybrid wireless sensor networks that are characterised by modularity, reliability, flexibility, robustness and scalability. The development of RFID tags, sensors, actuators and mobile phones makes it possible for the IoT to interact and cooperate to make the service better and accessible anytime, from anywhere (Lee and Park 2013).

- *Sensor networks* - Culler *et al.* (2004) define sensor networks as inexpensive, low-power communication devices, which can be deployed throughout a physical space, providing dense, sensing close to physical phenomena, while processing and communicating this information and coordinating actions with other nodes. However, the combination of sensor networks' capabilities with the system software technology that forms the Internet makes it possible to instrument the world with increasing fidelity. Sensor networks become a component of the future Internet (the IoT) (Lange, 2008).

- *Web standards and protocols* - the Internet of Things will support interactions among many heterogeneous sources of data, and many heterogeneous devices using standard interfaces and data models, to ensure a high degree of interoperability among diverse systems. Standards have an important role to play, both within an organisation or entity and across organisations. Communication protocols will ensure that the IoT cooperates with other users of the radio spectrum (Steen *et al*. 2011).

Having discussed the characteristics and building blocks of the IoT, the next section discusses the potential areas where the IoT can be or has been applied.

2.1.3 Application domains of the Internet of Things

Dutton (2005) defines the IoT as a technological revolution that represents the future of computing and communications, and says its development depends on dynamic technical innovation in various fields from wireless to nanotechnology. Vouk (2008) concurs that the IoT will become a reality over the coming years with omnipresent smart devices communicating wirelessly to improve the quality of our lives and consistently reducing the impact of humanity on the planet. Furthermore Culler *et al.* (2004) suggested that any kind of next-generation Internet-enabled portable device will set up advanced interactions with the "things" making up the new IoT, resulting in a pervasive infrastructure of fixed and mobile heterogeneous nodes seamlessly providing, exploiting or sharing context-based services and applications. Application of the IoT in the real world is possible through integration of several enabling technologies; capabilities offered by the IoT make it possible to develop a huge number of applications (Atzori *et al*., 2010).

This section discusses potential areas that are promising in application of the IoT, their enablers and "use cases".

- *Energy* - the world is investing heavily in ICT in the energy domain (Haller, Karnouskos and Schroth, 2009). Innovative technologies and concepts will emerge as the world moves towards a more dynamic, service-based and market-driven infrastructure where energy efficiency can be addressed.

Use cases: Advanced metering infrastructure systems that measure, collect and analyse energy usage from advanced devices such as electricity, gas and water meters.

Enablers: Wireless sensor networks.

- *Transportation and logistics* - systems where public and private transportation interacts, choosing the best path to avoid delays and congestion, and where multi-modal transport is smooth and easy. Roads and transported goods are equipped with tags and sensors that send important information to traffic-control sites and transportation facilities to better route the traffic. Such systems help tourists with maps and monitor the status of the transported goods (Atzori *et al.*, 2010).

  Use cases: Logistics real time information-processing technology-assisted driving provides cars, trains, and buses with better navigation and safety with mobile ticketing. A quick mobile web service allows for checking stations, the number of passengers, costs, available seats and service types. Logistics technology allows users to buy tickets. An augmented map-equipped technology enables travellers to browse the map and automatically call web services for information about hotels, restaurants and other services. Electronic toll gates (e-tolls) sense vehicles and deduct fees from vehicle owners passing and many related use cases.

  Enablers: Sensors, actuators, processing power and tags.

- *Health* - this is a non-intrusive monitoring system, preventing serious illness by adjusting the environment and selecting appropriate drugs and diet. This benefits both the healthcare providers and patients in term of cost-effectiveness. The IoT becomes essential in realising the vision of ambient assisted living (Haller, Karnouskos and Schroth, 2009).

  Use case: Ambient assisted living – tracking of objects and people, identification and authentication of people, automatic data-collection and sensing.

  Enablers: RFID and network sensors.

- *Smart environment* - this makes life easy and comfortable with the use of intelligent objects used in various sub-domains. Smart connectivity with existing networks and context-aware computation is part of IoT.

  Use cases:

  Smart cities - where productive areas, retail, residential and green spaces will co-exist and will be enhanced by IoT technologies.

  Smart homes - where no energy is wasted, interactive walls are able to display useful information, as well as pictures of art, videos of friends or relatives that live far apart.

  Smart offices or productive business environments, where offices become smart and interactive, factories relay production-related data in real time, face-to-face meetings are established through holograms, and documents are fully integrated in the workflow.

  Smart campuses - where services are integrated and authenticated users can access services within the campus, and guests and other smart environment sub-domains can access basic services (maps, parking, speed warnings, etc.)

Enablers: RFID tag, wireless sensor networks and actuators.

Even though the IoT has realised many of its potential applications, there are a number of challenges that still need to be addressed. The next section discusses the challenges faced by the IoT.

2.1.4 Challenges of the Internet of Things

Some of the key research issues in the IoT include scalability, standardisation, heterogeneity, identification and addressing objects. As the technology develops and matures, the range of corporate deployments will increase (Michael, Markusand Roger 2010). Therefore, the IoT presents significant challenges in terms of who can see what, and with which credentials. Therefore, in order for the IoT to realise its potential as a revolutionary Internet technology, the following are among the number of open research challenges that need to be addressed.

- *Scalability* - is affected by the huge increase in the number of network-connected devices. The IoT has a number of devices in one place; therefore the scale is much larger than

situations assumed in other systems (Paridel, Bainomugisha, Vanrompay, Berbersand Meuter, 2010). When a sensing or actuation task that pertains to millions of sensors or actuators, it is often not feasible to coordinate all the required devices owing to constraints such as time, memory, processing power, and energy consumption (Teixeira, Hachemand Georgantas, 2013). Solutions attempted so far indicate the shortcomings that indicate that a new middleware and language abstraction is needed to address the challenge of scalability in the IoT (Paridel et al. 2010).

- *Identification and addressing* - according to Dlodlo *et al.*(2012)the idea of the IoT is to create a network of uniquely addressable and identifiable objects that can communicate with each other, using their different communication capabilities. In order to address the billions of entities in the IoT, it is important to first identify them with a unique ID (identity). The ID can also be used to find other information about the entity of interest (Haller *et al.*, 2011).This forms the basis for object discovery that has proved very useful in coordinating highly distributed operations. Tracking and tracing of objects as they move along the supply chain is one of the most important basic functions of the IoT. It provides the foundation for product authentication. In order to implement this functionality, discovery services are required that allow for dynamically finding all information about a specific object.

- *Standardisation* - the IoT represents views towards a dynamic and self-configuring global network infrastructure that is standard-based and supported by interoperable communication protocols. Standardisation is essential for deployment and diffusion of any technology. Almost all commercially successful technologies have undergone a process of standardisation to achieve massive market penetration. For example, today's Internet and mobile phones would not have thrived without standards such as Internet Protocol (IP) (Dutton 2005). More standardisation needs to be done in areas of security, privacy architecture and communications (Evans 2011). Having discussed the challenges of scalability, identification, and addressing and standardisation, there is still the challenge of integrating technologies. The next section discusses the challenge of integrating the technologies with common goals.

- *Heterogeneity* - Serbanati et al. (2011) define the IoT as the technology that builds upon the existing Internet communication infrastructure and foresees a world permeated with embedded smart devices interconnected through this infrastructure. The IoT integrates different technologies to solve a problem. The integration of technologies in the IoT is referred as technology heterogeneity that is the existence of differentiated hardware, architectures, infrastructure, and the technologies of mobile devices, clouds, and wireless networks. Cutting-edge technologies are expected to initiate and facilitate collaboration among these heterogeneous computing devices towards unrestricted mobile computing. This idea makes the IoT an encompassing vision of integrating the real world (people, things and machines) into the Internet. For this reason, Haller *et al.* (2011)suggest that there must be a method of dealing with heterogeneity when building software infrastructure for the IoT. Interoperability, which has been discussed as characteristic of the IoT, becomes the key to separate its functionality from its technical implementation. SOA is ideally suited for this, since it encapsulates functionality in services with a common interface, abstracting from the underlying hardware protocols, and recognised as an adequate architectural style for the organisation of large-scale and distributed business logic.

In view of the foregoing, the scope of this researchis confined to addressing the research challenges relating to *heterogeneity*. The next section discusses the application of SOA as a component in addressing the problem of this study.

2.2 Smart Campus

A smart environment is based on ubiquitous computing where the environment interacts with its inhabitants at a physical layer (Silva, Costa, Geyer, Augustin and Maria, 2008). The physical layer consists of embedded and dedicated devices that are capable of interacting with each other through Internet connections in this environment. The idea of a smart environment is to simplify operations through automation; that is, there is no human intervention in the operations. The other goal of a smart environment is to support and enhance the abilities of its occupants in the execution of tasks, such as navigating unfamiliar spaces and providing reminders for some activities and functionalities (Dey, Abowdand Salber, 2000). However, based on the nature of

the environment, there must be a method of detecting the current state or context of such services, recommending the list of services available and invoking the appropriate service.

There are many smart-environment domain applications. These domains include the smart world, smart city, smart campus, smart building, smart home and smart house, etc. This work, however, focuses on the smart-campus domain. According to the focus of this research, the smart campus refers to an organisation that utilises services as provided by the IoT. The storage, ownership and expiry of large data become critical in the smart campus. The data has to be stored intelligently and used for smart monitoring and actuation. Therefore(Evans 2011) saw a need to develop artificial intelligence algorithms that could be centralised or distributed based on the need to make sense of the collected data.

There are major concerns driving smart-environment technologies that include decision-making, sense making, improving user experience, catering for convenient situations, saving energy, saving costs and other concerns that may arise as needs increase and technology improves each day. Smart-environment technologies improve the quality of life and consistently reduce the ecological impact of humankind. This work proposes a mechanism that can be used to select services composed by the smart campus domain.

2.2.1 Examples of smart-campus services

There are a number of services that can be provided through the IoT within a smart campus, some of which are discussed in this section.

2.2.1.1 Smart access

Smart access both controls and authenticates who is authorised within the campus. However, there are different methods that can be used to provide access. Those methods differ according to their purposes. The following are the different methods that can be used as smart access services:

- *Biometrics* - this is a method used for identification, tampering, counterfeiting and terrorist exploitation (*Federal Information Processing Standards,* 2006). Each unique data set would be stored in a database using unique information including fingerprints, hand geometry, iris patterns, facial patterns or voice prints.

- *Heat signature* - this method normally works on a digital model with a built-in Internet connection and pixel-perfect pictures, with an ability to record a heat signature. A heat signature is an infrared radiation that indicates temperature. The user will be requested to sign. The user's signature will then be stored on the database. Every time the user signs as per required to access the premises, the signature will be compared with the one stored on the database.
- *Smart card*- refers to the embedded computer-integrated circuit that is either a microprocessor in collaboration with an internal memory, or only a memory chip. These chips interact with a chip reader. Chips are used worldwide in finance, healthcare, secure identification and other applications.

2.2.1.2 Traffic-directing system

This kind of composite service serves visitors and people who are new to an organisation. The following are examples of traffic-directing systems.

- *A system to register and login to the mobile phone to find the map* - this kind of service allows a user to login (if already registered on the system) or register (if new) to access the organisation's map using a mobile phone.
- *Smart parking system* - this kind of service is flexible for both visitor and staff members. It works within the range of the user's destination. Itgives the options of the available parking spaces.
- *Speedometer* - this kind of service detects the speed of vehicles passing by and gives warnings. A supervisor gets feedback regarding the speed of the drivers on campus

2.2.1.3 Fire- and water-detection system

This kind of composite service is provided by the maintenance and safety departments for different purposes.

- *Fire alarm* - this kind of service senses if there is any smoke inside the building. If there is any, it then sends a notification to a central office with the context (building number, office number, date, time).
- *Water detector* - this kind of service detects if water has exceeded its watering level limit and sends a command to close the tank. It also detects if there is any water leakage and automatically sends the instruction to the departmental officer for maintenance.
- *Smart gardening* - this composite service detects if the soil needs nutrients and sends an instruction to a central point. It can also suggest which crops can be grown in that type of soil.

2.2.1.4 Smart security

Every employee would like to work in a place that is secure and safe. The security within the organisation benefits both employees and the employers. However, the possible services that can be provided by the IoT under security and safe departments are as follows:

- *Intruder detection with smart fence* - this kind of service detects if a user is authorised to be inside the premises of a smart electrical fence. If an identified moving object is prohibited to be in the premises, the system sends a report with the context of the intruder.
- *Surveillance camera* - cameras provide safety. They record and store data on digital storage for an organisation's reference for useful purposes.

2.2.1.5 Smart energy

This refers to connecting devices, rooms and other energy-consuming entities with the aim of monitoring their energy consumption and state of use. When the connection has been achieved, we can now attempt to keep energy use at a minimum, curb energy wastage and thus save money as well.

The challenge is that a smart campus becomes busy in such a manner that the energy demands surpass the supply that is available. In order to tackle this challenge, one would need to

interconnect all energy consumers to a regulatory system. This would monitor energy use, decrease or shut down some devices not in use. This would be in a form of deploying sensors in rooms or offices, which would check for the presence of people, automatically shutting down lights, air conditioners and other facilities. Sensors can also be embedded in devices such as television sets so that, if there is no user, devices automatically shut down.

The success of building smart energy technology can help governments to cut down costs such as electricity bills. As consumption is reduced, this will help bridge the gap between supply and demand of electricity.

2.2.2 Significance of the smart campus domain

A smart environment can be *instrumented* by providing a sandbox in which the ubiquitous computing scenario can be investigated. Smart environments depend on interconnected devices for their stability and recovery. The engineering of interconnection becomes possible through integrated technologies and heterogeneous systems. The interconnection becomes possible through intelligent systems that can make decisions on their own without human interaction.

In light of the significance of using the smart campus, the IoT can have different facets, depending on the perspective taken. As the number of connected devices in a physical layer increase, so do services offered through those devices (Laine 2012). The increase of services on the network is associated with the challenge of service selection. The problem with selection on such a large scale is recommending a service that best satisfies the user according to the user's context.

2.3 Cloud computing

Cloud computing is viewed as the service of computing in which the computing resources (hardware and software) are provided on demand and commercially at a certain cost (Mell and Grance 2011). The provision of such computing services can be done in three forms

(Infrastructure-as-a-service, Platform-as-a-service, and Software-as-a-service), whereas the cloud computing model allows access to information and computer resources from anywhere that a network connection is available to the cloud. The adoption of cloud computing has had a massive impact in different dimensions.

Section 2.3.1 discusses the various definitions of cloud computing. Section 2.3.2 discusses the characteristics of cloud computing. Section 2.3.3 discusses cloud computing service models. Section 2.3.4discusses how cloud computing can be deployed. In addition, Section 2.3.5 discusses the significance of cloud computing and its challenges.

2.3.1 Definition of cloud computing

The scope of the cloud computing definition is determined by the focus of any particular research project; hence, authors define cloud computing in different ways. This section looks at various authors' definitions of cloud computing and later synthesises them to come up with a definition of cloud computing according to the focus of this research.

Computing services delivered over the Internet are regarded as cloud computing. Commercially, cloud services need to develop a cloud with adequate storage capacity. Instead of keeping data on a hard drive or updating applications to meet their needs, operators may use a service over the Internet and at another location to store information or use the service's applications. These are services such as online file storage, social networking sites, webmail and online business applications STORAGE NETWORKING INDUSTRY ASSOCIATION, (2010).

The cloud-computing model allows access to information and computer resources from anywhere, provided there is a network connection available. However, Huth and Cebula (2011) define cloud computing as a subscription-based service where one can obtain networked storage space and computer resources. While a traditional computer setup requires the user to be in the same location as the data-storage device, the cloud obviates that step. It removes the need for a user to be in the same physical location as the hardware that stores the data. A cloud provider can both own and house the hardware and software necessary to run home or business applications. This is especially helpful for businesses that cannot afford the same amount of hardware and storage space as a bigger company. Small companies can store their information in the cloud, removing the cost of purchasing and storing memory devices. Additionally, because one is able

to buy only the amount of storage space required, a business can purchase more space or reduce its subscription as the business grows or less storage space is needed.

Stonebraker *et al.*(1998) regard cloud computing as a data centre hosting service. Cloud computing represents a different way to architect and remotely manage computing resources. For example, assuming that a user has accounts with Dropbox, Google, Amazon, etc., the user requires a web application with http service, a relational database, web service infrastructure, message queuing, an e-commerce application, persistent storage, remote hosting, etc. Santos, Gummadiand Rodrigues (2009) concur that cloud-computing infrastructures enable companies to cut costs by outsourcing computations on demand.

Cloud computing assembles large networks of virtual services: infrastructure services (e.g. to compute, storage, network) and software services (e.g. databases, message queuing systems, monitoring systems, load balancers). It embraces an elastic paradigm in which applications establish on-demand interactions with services to satisfy a required quality of service (QoS) including cost, response time and throughput (Zhang, Ranjan, Nepal, Menzel, and Armin, 2012). However, selecting and composing the right services to meet application requirements is a challenging problem.

With the background and the capabilities of cloud computing, the definition of cloud computing is synthesised. According to the focus of this research, cloud computing is defined as follows.

Cloud computing is an elastic, dynamic paradigm that integrates, stores and shares on-demand resources from different sources without worrying about the ownership and it can be accessed anywhere and anytime through the connection of the Internet.

The next section discusses the characteristics of cloud computing.

2.3.2 Characteristics of cloud computing

Every technology has its attributes that differentiate it from other technologies. However, this section discusses the characteristics of cloud computing that make it unique. The following are five major characteristics of cloud computing.

- *On-demand service* - is the capability that refers to computer services such as email, applications, networking or server services that can be provided in the cloud without requiring human interaction with each service provider (Uckelmann, Harrisonand Michahelles, 2011). Services within cloud databases are on-demand services. Users can request them anytime and anywhere.

- *Broad network access* - is the capability that refers to resources hosted in a private cloud network that are available for access from a wide range of devices, such as tablets, personal computers, MacBooks and Smartphones (Dialogic Corporation, 2010).The wide range of locations offers online access through standard mechanisms that promote use by heterogeneous client platforms.

- *Resource pooling* - capability refers to a collection of management servers used to distribute work among themselves and take over work from a failed member. The pooling together of the resources improves economic scalability (Bardsiri, Hashemi and Branch, 2012).In short, resource pooling optimises the use of provided resources.

- *Rapid elasticity* - this characteristic presents a cloud service from the ability of a provider to meet the IT needs of the consumer, which creates the perception that the service is infinitely scalable and increases its value (Computer and Division 2011). These services can be purchased in any quantity at any time.

- *Measured service* - here aspects of the cloud service are controlled and monitored by the cloud provider (Computer and Division 2011). This is crucial for billing, access control, resource optimisation, capacity planning and other tasks. The measured service provides performance guarantees.

Cloud computing allows for the sharing of servers and storage devices and increased utilisation. Applications can be easily migrated from one physical server to another. The cloud's

performance is monitored, its consistency is ensured, and loosely coupled architectures are constructed using web services as the system interface.

The next section discusses the model of services offered by the cloud.

2.3.3 Service model

Cloud computing offers services at three different levels. These levels support virtualisation and management of differing levels of the solution stack.

- *Software-as-a–service (SaaS)* - the capability provided to the consumer is to use the provider's applications running on a cloud infrastructure (*Dialogic White Paper*, 2010).Providers typically host and manage a given application in their own data centres and make it available to multiple tenants and users over the web (Stonebraker et al. 1998). The applications are accessible from various client devices through a client interface such as a web browser (e.g. web-based email). The software interacts with a user directly.

- *Platform-as-a-service (PaaS)* - the capability provided to the consumer is to deploy onto the cloud infrastructure (network, servers, operating systems or storage) consumer-created or acquired applications created using programming languages and tools supported by the provider (Kulkarni, Sutarand Gambhir, 2012).This platform consists of infrastructure software, and typically includes a database, middleware and development tools. An example of PaaS is Google AppEngine, which is written in the Python or Java programming language.

- *Infrastructure-as-a-service (IaaS)* - the capability provided to the consumer is to provide processing, storage, networks and other fundamental computing resources. A consumer is able to deploy and run arbitrary software, which can include operating systems and applications (Vouk 2008). Seung-hee (2012)recommends services based on IaaS because an IaaS provider does very little management other than keep the data centre operational, and users must deploy and manage the software services themselves just the way they

would in their own data centre. Examples of IaaS are Amazon web services and Elastic Compute Cloud (EC2).

Having examined the different service models, one would like to deploy them. Therefore, the next section gives the scope overview on how cloud computing can be deployed

2.3.4 Deployment of cloud computing

Through cloud computing, a company can rapidly deploy applications where the underlying technology components can expand and contract with the flow of the business life cycle. This section discusses the different deployments of cloud computing, as shown in Figure 2.

- *A public or external cloud* - is where resources are dynamically provisioned on a self-service basis over the Internet via web applications (Computer and Division 2011). The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.

- *A community cloud* - infrastructure is shared by several organizations and supports a specific community that has shared concerns (e.g. mission, security requirement, policy, or compliance considerations). It may be managed by the organizations or a third party and may exist on-premises or off-premises (Bardsiri et al. 2012).

- *A private cloud or internal cloud* - infrastructure is operated solely for a single organization. It may be managed by the organization or a third party, and may exist on-premises or off-premises (Huth and Cebula 2011). Larger enterprises may find it more economical to develop future state architectures internally to deliver the benefits of cloud computing to internal subscribers. This model is ideal for enterprises that are organised with a shared-services IT infrastructure. The private cloud can also be used by existing legacy IT departments to dramatically reduce their costs and as an opportunity to shift from a cost centre to a value centre in the eyes of the business.

- *A hybrid cloud* -infrastructure is a composition of any of two or more clouds (private, community or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g. cloud bursting for load-balancing between clouds) (Mell and Grance 2011). It embraces an elastic paradigm in which applications establish on-demand interactions with services to satisfy required QoS including cost, response time and throughput.

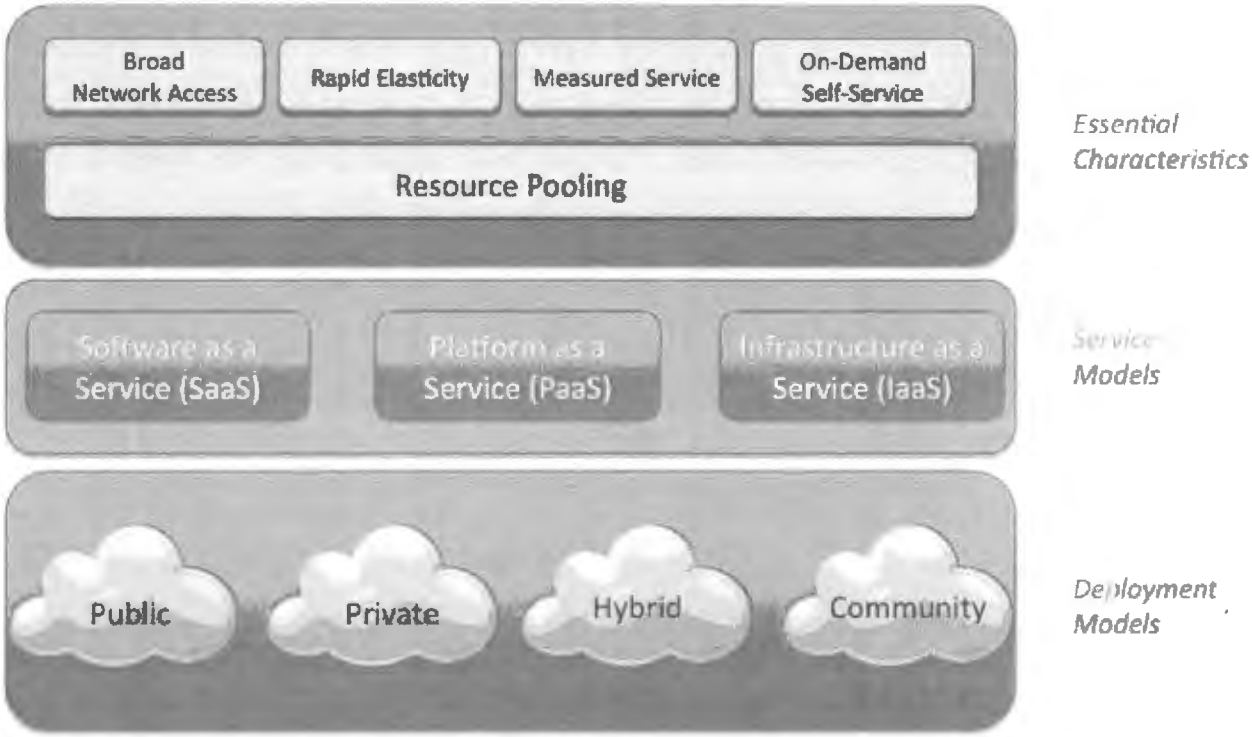


Figure 2: The NIST definition of cloud computing (Mell and Grance 2011)

Figure 2 summarises the definition, service models, deployments and characteristics of cloud computing by the National Institute of Standards and Technology (NIST) organisation. The next section discusses the motivation behind the deployment of cloud computing.

2.3.5 Significance of Cloud Computing

Governments, businesses and researchers can all benefit from the adoption of cloud computing services in many ways. Here we discuss opportunities that have been raised by the adoption of cloud computing in different dimensions.

2.3.5.1 Opportunities

- *Technical*- cloudcomputing technology integrates services in a scalable server. Storage and processing needs can also be met by the cloud. Cloud computing presents IT organisations with a fundamentally different model of operation, one that takes advantage of the maturity of web applications and networks and the rising interoperability of computing systems to provide IT services. Cloud providers specialise in particular applications and services, and this expertise allows them to efficiently manage upgrades and maintenance, backups, disaster recovery and failover functions(Hauck, Huber, Klems, Kounev, Pretschner, Reussner and Tai, 2010). As a result, consumers of cloud services may see increased reliability, even as costs decline due to economies of scale and other production factors. In a mature cloud computing environment, institutions would be able to add new IT services that are scalable in a sense that when additional resources are needed they can be accessed (Rader 2012). Cloud computing software also facilitates the manipulation of large databases that increase the competiveness and productivity of service providers by adopting local/hybrid/public computing platforms.

- *Government* - governments can benefit from the adoption of cloud computing by providing services more efficiently to a broad range of customers. In the United States the Obama administration has launched projects to identify services and solutions that can use cloud computing (Lundberg 2009). The United Kingdom government is adopting cloud computing to reduce its administrative costs (Black et al. 2009). Japan has established a government cloud to enable various ministries to consolidate hardware and create shared platforms(*Japanese Ministry of Internal Affairs and Communications;* Wakamatsu-cho, 2011). Governments have important roles to play in encouraging the

adoption of cloud computing and facilitating the use of cloud-based services by researchers and the business sector.

- *Business*- users pay per use for services and infrastructure, reducing the requirement for capital investment. Cloud computing can enable entirely new innovative business services. Businesses can develop new services based on cloud computing, as well as use the cloud to manage data-intensive activities more efficiently (Vouk 2008). The major reduction in capital costs that cloud computing provides makes it attractive to small and medium-sized enterprises (SMEs) with limited access to capital. Clouds provide an excellent backend for mobile phone applications as well.

- *Challenges* - the use of cloud services raises new issues in regard to privacy, security, trust, data-transfer capacity and lock-in with service providers (Lundberg 2009). The other challenges are associated with the loss of service control, unclear schemes on the pay-as-you-use approach, migration and the lack of a platform to find or select cloud providers. Research timelines are challenged by the fast-moving markets. Selecting and composing the right services that meet application requirements is still a challenging problem. However, this work will present a mechanism we can use to select services provided by a smart campus in the private cloud. This is the idea that someone can offer hosted set software that a user does not own but pays for some element of utilisation.

2.4 Service-oriented architecture

This section gives an overview of service-oriented architecture (SOA) by looking at different definitions. From the literature, it then synthesises a definition of SOA according to the purpose of this study. SOA is a broad and complex architecture that cannot be explained only by definition; hence, Section 2.4.2discusses the architecture and reference models of SOA. Section 2.4.3discusses the how this study benefits from using SOA. Lastly, Section 2.4.4discusses potential research areas in the SOA paradigm in general and later identifies the research area focus of this study.

2.4.1.1 Definition of SOA

This subsection defines SOA from different literature resources and later synthesises all the definitions and compiles a definition of SOA according to the focus of this research.

Looking at the overview of Information Technology (IT) Infrastructure, Samreen(2011)defines SOA as an architectural style of building with distributed and loosely coupled applications in such a way that they are composed of discrete software agents that have simple, well-defined interfaces that are orchestrated through a loose coupling to perform a required function. To this extent, a service becomes a logical representation of a repeatable business activity that has a specified outcome, is self-contained and comprises other services.

Architectural components of SOA systems consist of service consumers and service providers. Therefore SOA becomes a transparent interface that integrates the functionalities performed by both consumers and providers (Bianco, Lewis, Mersonand Simanta, 2011). SOA is created to satisfy business goals that include easy and flexible integration with legacy systems. Bianco *et al.*(2011)further elaborates that from a software point of view the architecture is the bridge between business goals and the software system.

The operational characteristics of SOA as an architectural approach is that it defines, links and integrates reusable business services with clear boundaries and is self-contained with its own functionalities (Mabrouk 2008). This definition implies that SOA is more suited to interoperability and heterogeneous environments like the IoT. From a technical perspective and in terms of the significance of SOA an Oracle White Paper (Pavlik, Danand Tugdual, 2007) sees SOA as a modern approach used to build and integrate distributed applications and greatly reduce the time, effort and cost involved in developing the software projects. In business, SOA exposes legacy functionality to remote clients by implementing new business process models that reduce the overall IT expenditure for innovation through software investment (Bieberstein, Bose, Fiammante, Jonesand Shah, 2006).

In light of the key concepts, ideas and literature synthesise in each definition of SOA.As a working definition, this study defines SOA as follows:

SOA is an architectural approach that integrates distributed and heterogeneous dynamic environments to perform a selection of a well-defined service that support remote clients. SOA consists of a composite set of business-aligned services that support a flexible and dynamic environment where even complex services can easily be composed using individual services from various service providers. Those individual services can be selected, reused and integrated dynamically, based on service functionalities and performance constraints.

Using the above working definition of SOA, the next section discusses the service-oriented architecture and reference model.

2.4.2 Service oriented architecture and reference model

The SOA reference model is an abstract framework for understanding important relationships between components of SOA. This section discusses the components of SOA and the significance of the relationships between its entities. It further discusses the protocols and standards used to realise the components of the SOA.

Architectural component - Samreen (2011) captures the interaction pattern among the key components of SOA, as depicted in Figure 3 below. The *service consumer* performs dynamic service locating by querying the *service registry* for a service that matches its criteria. The *service provider* defines service descriptions and publishes the descriptions of services so that services can be discovered. These services are available either on corporate intranets or on the Internet, and they are delivered either on open or on proprietary network protocols (Kontogiannis, Lewis and Smith, 2008).

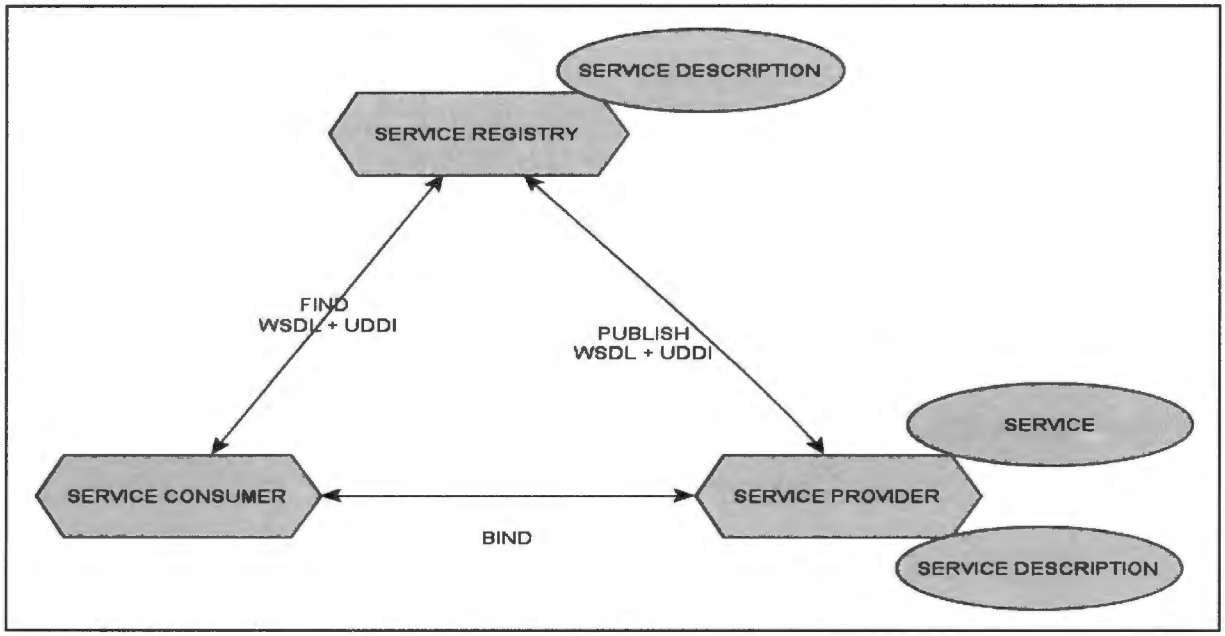


Figure 3: Service oriented architecture (Samreen 2011)

Standards and protocols: Simple Object Access Protocol (SOAP) is an Extensible Mark-up Language (XML) based protocol that facilitates, publishes, finds, binds, and invokes operation as indicated in Figure 3. The Universal Description, Discovery and Integration (UDDI) protocol performs discovery of service(s). Any action (search or select a service) gives the consumer access to the Web Services Description Language (WSDL) for a service. The SOA contributes to building up web services, where web services are software systems designed to support interoperable machine-to-machine interaction over a network. Web services provide standard means of interoperating different software applications, running on different platforms and/or a framework (Maximilien 2004).

The WSDL works as a transport protocol, message formatter and locator and is commonly used today to produce service catalogues although it still essentially lacks strong concepts for service personalisation, which is crucial for advanced usability (Balke and Wagner, 2003).

The next section presents the benefits of using SOA.

2.4.3 Benefits of SOA

The literature shows that SOA can have great value and contributes in different dimensions of life. This section discusses the business values of using SOA with the adoption of the framework proposed by Joachim(2012). This entire study focuses on six dimensions: IT infrastructure, components, technical aspects, and organisational, operational and business aspects. SOA reduces costs, provides innovative service to customers, streamlines business processes and facilitates agile adaptation and reaction to opportunities (Kontogiannis et al. 2008). The IT infrastructure is an aspect that ensures that the composite of hardware, software, network and services works hand in hand, especially in dynamic environments like SOA. The relationship between entities is vital in order to improve the technology and user needs. If the technical aspects are not satisfied by the abilities of the paradigm, that creates more research opportunities. Technical aspects are more concerned with the logic in the adoption of the technology, and they check if the resources are available to make it possible to run the technology and benefit the business.

The adoption of SOA must benefit business to the extent of making a business partner; that is why the organisational aspect is essential. The SOA is relevant in organisations through investment, banking systems, self-service vendors, and the use of web services in enterprises, business-to-business integration and other functionalities.

The operational aspect is essential for establishing trust and a relationship between the service consumer and the service provider on an implementation and application level. Table 1 summarises the dimensional aspects to zero down the scope of which SOA may have an impact. This table briefly gives a description on how SOA benefit from that dimension. The last six aspects are the most important aspect to make SOA relevant and adopted. These aspects are used to evaluate the dimensions as shown in the table.

Table 1: Classification of the benefits of SOA (Joachim, 2011)

Dimension	Description	Cost reduction	Agility	Data quality	Process monitoring	Time consumption	B2B Integration
IT Infrastructure	Sharable and usable IT resources (Kimpim 2014)	√	√	X	X	X	X
Components	Promotes and publishes services, making it easier for a customer to find services. Makes it easier to find new partners and services by allowing the user to search for the most suitable service that meets a user's needs (Mabrouk 2008)	X	X	X	√	√	√
Technical	Loosely coupled applications change the way IT costs are distributed, with less expense in implementation and more investment in reuse (Samreen 2011)	√	X	√	√	X	X
Organisational	Extensibility makes SOA solutions available in all sizes	√	**X**	√	**X**	**X**	**X**

	of organisations. SOA found relevant to integrate and gather data of different units in one place (Kimpim 2014).						
Operational	Reusable services change is easier. This not only promotes efficiency but provides a strong ability to change and align IT with business (Bianco et al. 2011)	√	√	X	X	X	√

The SONIC Software Corporation (Bobby 2005)views *cost reduction and less time consumption* as significant features that support heterogeneity and distribution, in concert with the improvement of business performance. SOA-based application development holds the promise of reduced application-development and maintenance costs through service reuse. However, organisations need *agility* to maintain strategic advantages in businesses operating on faster and faster time scales (Schelp and Aier 2009). Agility is the ability to change the state or data while under efficient control.

The European company UNISERV pointed out other important aspects in the context of improving SOA. The *data quality* assists in preventing incomplete or incorrect data from being stored in the database. The *processing monitor* is promoted by SOA, where a loosely couple of businesses are connected to execute processes. Lastly, *business-to-business integration* principles and infrastructure go a long way toward reaching *business* goals and building an IT ecosystem.

The theoretical evaluation presented in Table 1 shows the enablers of the adoption of SOA, although there are still barriers within the same paradigm. However, the next section discusses the challenges associated with SOA in different dimensions.

2.4.4 Challenges in SOA

The benefits of adopting SOA are discussed in the previous section. However, for the SOA to realise its potential, there are a number of challenges that need to be addressed and overcome. This study adopts the Kontogiannis *et al.* (2008) method of classifying research issues – that is, business, engineering, and operational and cross-cutting aspects.

- *Business* -Modern organisations rely on IT for their day-to-day operations. The dependency on IT is even greater in organisations that deliver services rather than physical products. The problems that have been observed in businesses are in two major areas. The first one is the mismatch between the business and IT. Secondly, it is the duplication of functionality and process. Hence, to solve these two major problems, SOA has been recommended by many businesses as a solution. As a means of increasing business and IT benefits, SOA must return to its original principles, such as the enterprise-class shared.

- *Operations* -All service composition work done so far is based on a given selection of services under a well-set environment. The ultimate goal in personalised service provisioning has to be the fulfilment of individual user needs expressed as a complex task that can further be divided into sub-goals and subsequently matched to different services. Even though UDDI and WSDL are commonly used today to produce service catalogues they still essentially lack strong concepts for service *personalisation,* which is crucial for advanced usability (Balke and Wagner, 2003).

- *Service discovery*- is a critical element of large-scale systems. In service discovery, users are not expected to systematically query all service directories in a discovery process, but a set of service directories should provide a single point of contact for the service

discovery. Users should provide a semantic description of the service they want and the discovery should match the user requirements against semantic descriptions of the published services. The discovery also provides the ability to monitor and discover resources and services by means of cloud computing. However, the fundamental problem is the need to be scalable to handle huge amounts of data from multiple sources.

- *Engineering* -in the service-oriented environment, system components are distributed and deployed on heterogeneous platforms. However, service composition is necessitated by the fact that in such environments there may be no single service that meets that required functionality, and a combination of services may need less environmental capabilities than a single service. There will be a large number of service combinations that fulfil a given goal. This research envisages an approach which uses that emanating from recommender systems in general for service selection in highly dynamic environments (Papazoglou and Heuvel 2006). In this approach, users should been provided with the language and a tool to express their requirements and preferences, and to assess suitable solutions when submitting a job. Such a tool should be able to be integrated in the cloud-scheduling process in order to improve the selection capabilities of current systems.

- *Crosscutting* - Traditionally, companies are organised functionally. This means that there are different departments in each unit for different functions in a company; a customer-service department to service the customers; a claims department that assesses claims; the human resources department for the workforce; and other departments. All these departments use their own IT systems to keep track of the data that is needed. Because all the departments use their own IT systems, and these systems are not connected to each other, information is duplicated within an organisation. This can lead to differences between departments, because the information is not only stored but also changed within these systems. This leads to inconsistencies across the organisation unless the information is synchronised between all the systems.

The next section describes how SOA can be integrated into a smart campus environment. For this, an illustrative example is presented.

2.4.5 Integration of SOA into smart campus

A smart campus environment is considered as a campus that has multiple buildings, each housing a variety of IoT capabilities. The capabilities such as sensing, shrinking, actuating, and connecting. These capabilities need to be accessible within the individual buildings as well as at campus level (i.e. from other buildings). For IoT capabilities to be accessible from outside, it is necessary to expose the network-accessible and public interfaces of these capabilities (Tsaimos et al. 2008). Given the fact that these IoT capabilities are in different buildings, and therefore under different administrative controls, integration and interoperability remain critical requirements.

The service-oriented design architecture can address these challenges. The IoT capabilities within the individual buildings can be transformed into IoT services and develop following the service-oriented architectural style. Since SOA is based on accepted industry standards, SOA-based IoT services will support interoperability and integration with other SOA-based IoT services from other buildings within the smart campus (Spiess *et al.* 2009). Furthermore, SOA-based services can be accessed and invoked over the network; this capability will ensure that SOA-based IoT services from one building can be accessed and invoked from other buildings through the smart campus area network. The standard-based description of individual IoT services means that these services can be discovered and linked, either dynamically or manually, to provide more complex service functionalities (Fensel *et al.* 2002). In the business software domain, complex applications based on composition and collaboration among diverse services has been appearing. SOA provides the dynamic paradigm for composing the distributed application, even in heterogeneous environments with different ownership domains. The middleware that hosts these services can be hosted on the cloud. This deployment strategy has added benefits that will be discussed in the next section.

2.5 Recommender systems

Recommender systems are deployed in e-commerce settings as a product site to help customers find products according to their special preferences. According to Manouselis, Costopoulou and Sideridis (2006) those preferences may infer the rating of the user based on their purchase history, browsing and return visits to different pages that result in a major challenge of incompleteness of data. Available data is biased because products, which the user would rate

negatively, are simply not purchased or not visited multiple times, thus null values are always positive. The greatest challenge is to correlate users reliably when they overlap on a few services.

Users save time by using a recommender system that helps them to choose from a variety of options. The purpose of recommender systems is to pre-select information a user might be interested in(Prabhakar and Manikrao, 2006). They make a prediction of the needs or interests of the user. They are deployed for service selection (Singh and Hunhs 2005).

The remainder of this section is structured as follows. Section2.5.1discusses the personalisation concept in recommender systems. Section 2.5.2discusses recommender systems that are context-aware. Section 2.5.3 discusses preference-based recommender systems. Lastly, Section 2.5.4 presents the classification of the preference-based recommender systems.

2.5.1 Personalisation

Personalisation is any information that can be used to adapt the interaction of a user with a system or service to the needs and preferences of the user or user group (Jembere, Adigun, Xuluand Ojong, 2007). Personalisation of a user's information should be well defined and made available to a context-aware system (Jørstad and Thanh 2006). Context is the structure that reflects the information that characterises the interactions between human application and the surrounding environment.

2.5.2 Context-aware recommender systems

Context-aware recommender systems cover the dynamicity of the environment. A collaborative filtering recommender system provides filtering capability where a large number of similar services might be available. A semantic matching algorithm is advisable to be used for filtering (Prabhakarand Manikrao , 2006). Recommender systems select a service from a set of similar services and provide a rating of services that simply indicate the satisfaction levels of users. The

ratings are stored in a repository in order to be used as input to the next recommendation. Recommender-system technologies produce quality recommendations for very large-scale problems. Recommender systems rank a list of similar services using given algorithms. Some systems are based on memory or on the model. Collaborative filtering is applicable in both memory- and model-based systems as well as in dynamic environments.

2.5.3 Preference-based selection

We are moving from traditional applications to advanced technology (Mahmoud 2014). Recommender systems have been used mostly in e-commerce applications, search engines and retrieval systems. Now we are introducing recommender systems to smart environments.

Services selected on recommender algorithms are based on user requirements or satisfaction, called user preferences according to this study. In this approach, users need a tool to express their requirements or preferences and to assess suitable solutions when submitting a job to a cloud system. Such a tool can be integrated into the cloud's scheduling process in order to improve the selection capabilities of current systems. User-preference-based approaches help users to express their needs; the service providers satisfactorily provide users' requested services, according to this study. At some point, the user may require a service; but maybe available services might not meet the need of the user to accomplish a certain task. Therefore, user-preference based approaches helps users to express their wishes and receive relevant services. In this study, we have used collaborative filtering as one of user-preference-based approaches applicable in a highly dynamic environment. Collaborative filtering says those who agreed in the past tend to agree again in the future. It looks for users who share the same selection pattern with the active user. This approach supports the dynamic selection of services in a smart environment. User-preference collaborative filtering is applicable in a hybrid of memory-based and model-based recommender systems.

2.5.3.1 Model-based approach

The model-based approach uses the database to estimate or learn a model which will be used for prediction (Li, Matejka, Grossman, Konstanand Fitzmaurice, 2011). This class of approach first builds a model from the users and then uses the model for making predictions about the active user. After clustering the modelled users ahead of time, the active user is placed in one of the clusters. Alternatively, one might build a decision tree (Drachster, Hummel and Koper, 2007).

2.5.3.2 Memory-based approach

Memory-based approaches consider the ratings of all users directly, instead of via an intervening step of building a model (Drachster et al. 2007). There are some limitations to applying recommender system approaches to service selection. The first shortcoming is the fact that purchase of a product or selection of a service does not imply preference fora given product or service. The registry or broker does not provide the service that it is recommending and may have little to say about its quality and other features. Service registries do not have control of service interaction, whereas an e-commerce site would know that a product was shipped. The last shortcoming of recommendation systems considered in this study is that services would be invoked multiple times and the registry would not even be aware of the repeat customers of a provider. Services should be selected based on user preference dynamically in a pervasive grid environment.

2.5.3.3 Preference-based recommender systems

This section discusses a list of preference-based recommender systems that are later evaluated based on theoretical IoT requirements.

- *Collaborative filtering* -Wei, Jianwei, Shuiguang, Ying and Zhaohui (2012) state that the underlying assumption of the collaborative filtering approach is that those who agreed in the past tend to agree again in the future. Collaborative filtering systems usually take two steps. The first step is they look for users who share the same selection patterns with the active user (the user whom the prediction is for); and the last step is they use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user.

Collaborative filtering recommends items that were used by similar users in the past. It uses this to predict the utility of items to a particular user based on a database of user utility from samples or popularity votes from other users. Manikrao and Prabhakar(2005)used recommendation systems in dynamic web services frameworks. The recommendation system uses the item-based collaborative filtering approach. As users rate web services, it is possible to predict how a given user will rate a particular web service. Once it knows the prediction of ratings for each web service satisfying user requirements, it can recommend web services in order of their rating. This approach looks at the set of web services the target user has rated and computes how similar they are to the web service for which the user rating is to be predicted. Once the similar web services are found, the prediction is computed by taking a weighted average of the target user's rating on the similar web services. The item-based collaborative filtering approach has two aspects, namely similarity computation and prediction generation. (Manouselis et al. 2006). Collaborative filtering has been used by well-known e-commerce sites, e.g. Amazon.com, where user's ratings for different products are stored centrally.

- *Item-based collaborative filtering*- focuses on items, assuming that items rated similarly are probably similar. It recommends items with the highest correlation (based on ratings of the items). It improves quality, is domain-independent and without content analyses. The problem is with adding an item or user benefit from the user's experience (Yang, Steckand Hill, 2012).

- *In user-based collaborative filtering* -users who rated the same item similarly probably have the same taste; based on that assumption, this technique recommends unseen items already rated by similar users (Drachster et al. 2007). This approach has a new user problem.

- *In demographic-based collaborative filtering*- users with similar attributes are matched, and items that are preferred by similar users are recommended. It is based on user data instead of on ratings (Drachster et al. 2007).

- *Content-based recommender systems*- Jembere *et al.*(2006) attempt to recommend an item that is similar to items the user liked in the past. This model focuses on an algorithm for learning user preferences and filters a stream of new items for those that most closely match that user's preferences. This approach builds on the fundamental assumption that users are not able to formulate queries that express their interests and which could be applied to services by indexing the textual description of a service based on the words that occur in them. However, this is a step backward from current web service standards, which involve formal, structured descriptions of services, and support discovery based on those descriptions. Content-based methods make recommendations by analysing the description of the item that has been rated by or for the user, and the description of items to be recommended (Pazzani and Billsus 2007). Many approaches in this area are specialised versions of classification learners with the goal of learning a function that predicts to which class of a document belongs (liked or not liked).

- *An attribute-based recommender system*- is a special instance of a content-based recommender system that recommends products to customers based on syntactic properties of the products. For instance, if customer does a search for a historical book, and the e-commerce site responds with a list of three recommended books, that is an example of an attribute-based recommendation. Attribute-based recommendations can be personal, depending on whether the e-commerce site remembers the attribute preferences for a customer. There are two sub-problems of content-based recommender systems. The first one is finding a representation of the service. The second one is to create a profile that allows the unseen services to be recommended. Once representation has been found for a document, a classification algorithm can learn a profile to distinguish the representation of highly rated services from others. Attribute-based recommender systems recommend items based on the matching of their attributes to the user profile. They have no new-user problem. They are very sensitive in changing preferences. They can map from a user's needs to items and work with categories (Singh and Hunhs 2005).

- *Case-based reasoning recommender systems*- assume that if a user likes a certain item, the user will probably also like similar items. It recommends new but similar items. This technique complements the recommendation strategy by adding an additional data source

for available learning activities. If insufficient data is available for collaborative filtering techniques, the recommendation strategy could switch to case-based reasoning (Drachster et al. 2007).

- *Context-awareness and recommender systems* -It is a given that user preferences in highly dynamic environments are dependent on the user context. An investigation of the work on context-based recommender systems identified three approaches to factoring context into the personalisation process. These are rule-based recommendation systems; item-filtering context-aware recommender systems; and context-indexed preference recommendation systems.

2.5.3.4 Rule-based context-aware recommender systems

Since the IoT is a dynamic environment, there is a need to look at context-aware recommender systems. This section discusses those recommender systems.

- *Rule-based recommender systems* –have rules to recommend other products based on user history. For example, a system may contain a rule that recommends the sequel to a book or movie to people who have purchased the early item in the series. Another rule might recommend a new compact disc (CD) by an artist to users who purchased earlier CDs by that artist. Rule-based systems may capture several common reasons for making recommendations, but they do not offer the same detailed personalised recommendations that are available with other recommendation systems (Pazzani and Billsus 2007).

- *Item filtering-based context-aware recommender systems* - are based on the premise that some items are only applicable in a given context. The context within which a service is applicable is captured in the item's description. Examples of such works are the mobile architecture (Aldinucci, Daneletto and Kilpatrick, 2008;Yap, Tanand Pang, 2007; and Buriano, 2005).In the mobile architecture, the system gets the current context from the context providers and uses the current context to filter the services that will be accessed using the mobile device. The personalisation process acts as a transparent sheet of paper

operating on top of the context-filtering process. The approach proposed in Yap et al. (2007) scores for every item against every possible high-level context item. The item with the highest weighted average score for the entire current context will be a candidate for recommendation.

- *Context-indexed-preference context-aware recommender systems* - Recommender systems in this category use the fact that since user preferences or needs are context-dependent, context data can be used to index user preferences. That is, before a request is passed for selection of the services, the user's preferences in the current framework should be captured as soft constraints to the request. Examples of such approaches include those by Jembere *et al.*(2006)and Averbakh, Krause and Skoutas (2006). In Schafer,Konstanand Riedl(2001) an algorithm for automatically extracting context-based user preferences from the user session data is developed. The user preferences are stored within the context under which they were discovered. In this case, only the preferences that satisfy the current user context are processed. The approach in Schafer *et al.*(2001) follows a slightly different approach

2.5.4 Classification of recommender systems

Recommender systems are classified as utility-based, knowledge-based, model-based and memory-based approaches, as shown in Figure 4 below.

The utility-based approach models a user's multi-attribute utility function and recommends items with the highest utility based on function (Shiu-li 2008). The k*nowledge-based approach* provides advice to users about services they might be interested to request (Shiu-li 2008).The *model-based approach* uses the database to estimate or learn a model which will be used for prediction(Li et al. 2011). This class of approach first builds a model from the given users and then uses the model for making predictions about the active user. After clustering the users ahead of time, the active user is placed in one of the clusters. Alternatively, one might build decision trees (Manikrao and Prabhakar 2005).

The *memory-based approach* considers the rating of all users directly, instead of via an intervening step of building a model (Manikrao and Prabhakar 2005). There are some limitations on applying recommender-system approaches to service selection. The shortcoming is the fact that the purchase of a product or the selection of a service does not imply any preference for a given product or service. The registry or broker does not provide the service that it is recommending and may have little to say about its quality and other features. Service registries do not have any control over service interaction, whereas an e-commerce site would know that a product was shipped. Another shortcoming of the recommender systems considered in this study is that services would be invoked multiple times and the registry would not even be aware of the repeat customers of a provider. This work is based on both memory and model. Therefore, the candidates remaining for further evaluation are content-based and collaborative filtering.

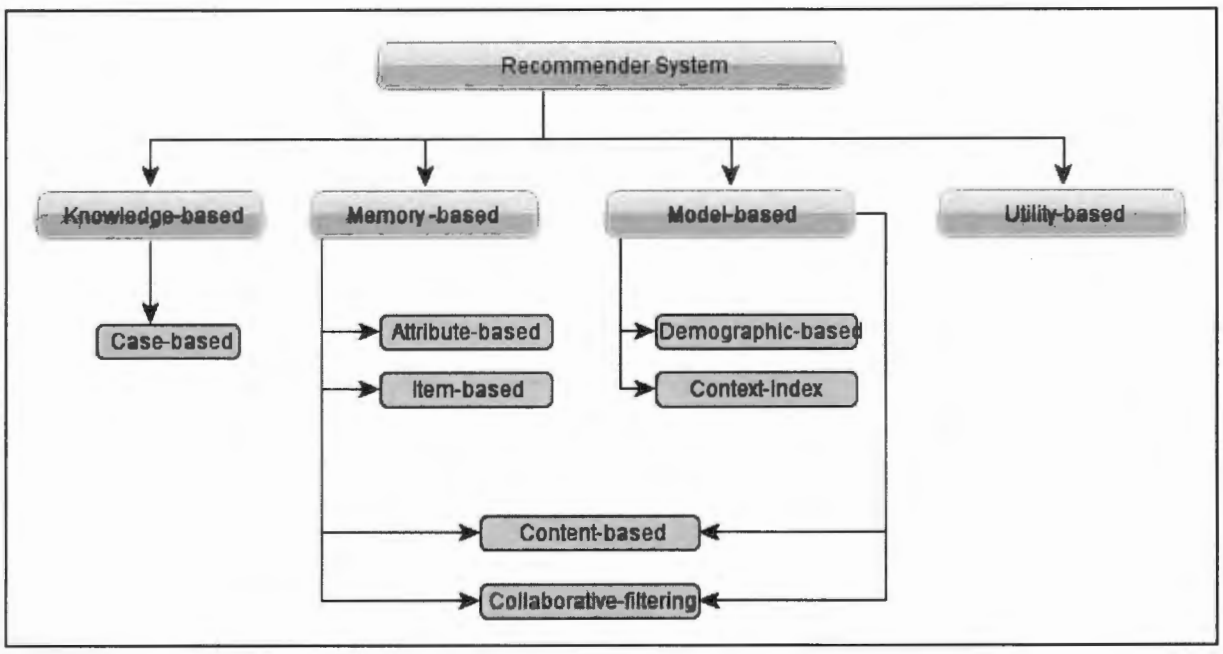


Figure 4: Classification of recommender systems (Manqele, Dlodlo, Adigunand Xulu, 2012)

Collaborative filtering - The underlying assumption of the collaborative-filtering approach is that those who agreed in the past tend to agree in the future. Collaborative filtering systems usually take two steps: they look for users who share the same selection patterns with the active user (the user whom the prediction is for) and use the ratings from those like-minded users found in step 1 to calculate a prediction for the active user. Collaborative filtering recommends items that were

used by similar users in the past. It uses this to predict the utility of items to a particular user based on a database of user utility from samples or popular votes by other users (Wei et al. 2012).

2.6 Summary

This chapter has presented a survey of the paradigms that were identified in the statement of the problem and the research questions. Against the background of the paradigms, the next chapter presents critical analyses of the existing service-selection architectures and approaches.

3 Literature Review

Thischapter analyses the existing scholarship in architectures that support user-preference-based service selection. The goal of this study is to develop a preference-based architecture that will enable the selection of smart-campus IoT services in the cloud using recommender systems for mobile devices. Using wisdom from the literature discussed in this chapter, a selection approach for this work will be proposed.

Section 3.1sets the context by giving a brief background to service selection and how a normal web service can be re-purposed for an IoT environment. Section 3.2 analyses the existing architectures. Section 3.3analyses the existing service selection approaches. Section 3.4 discusses the proposed service-selection approach and evaluates the existing preference-based recommender systems based on the IoT system consideration that will be gathered from the literature.

3.1 Context

In order to appreciate the significance of this chapter and the study more generally, it is necessary to put the discussion of service selection in context. To do this, it is necessary to draw from service-oriented computing (SOC) literature because this lays the foundation for the topic of service selection. Service Oriented Computing is discussed next.

Service-oriented computing is the computing paradigm that utilises services as fundamental elements for developing applications and business solutions (Papazoglou and Heuvel 2006).The set of concepts, principles and methods that represent computing in service-oriented architecture (SOA), in which software applications are constructed based on independent component services with standard interfaces, is referred to SOC (Chen Zhou, Liang-Tien Chiaand Bu-Sung Lee, 2004). SOC supports the development of rapid, low-cost and easy composition of distributed applications even in heterogeneous and highly dynamic environments. The emergence of the SOC paradigm promises to enable businesses and organisations to collaborate in an unprecedented manner by means of technologies such as web services (Ahsan 2006).

Kuropka, Lauresand Troger (2008) captured the provisioning of web services by means of a service-delivery lifecycle which is accomplished through three sub-cycles, namely the planning, binding and enactment sub-cycles. The service-delivery lifecycle presents a good view of what is necessary in order to provide electronic services of any kind (Kuropka et al. 2008).

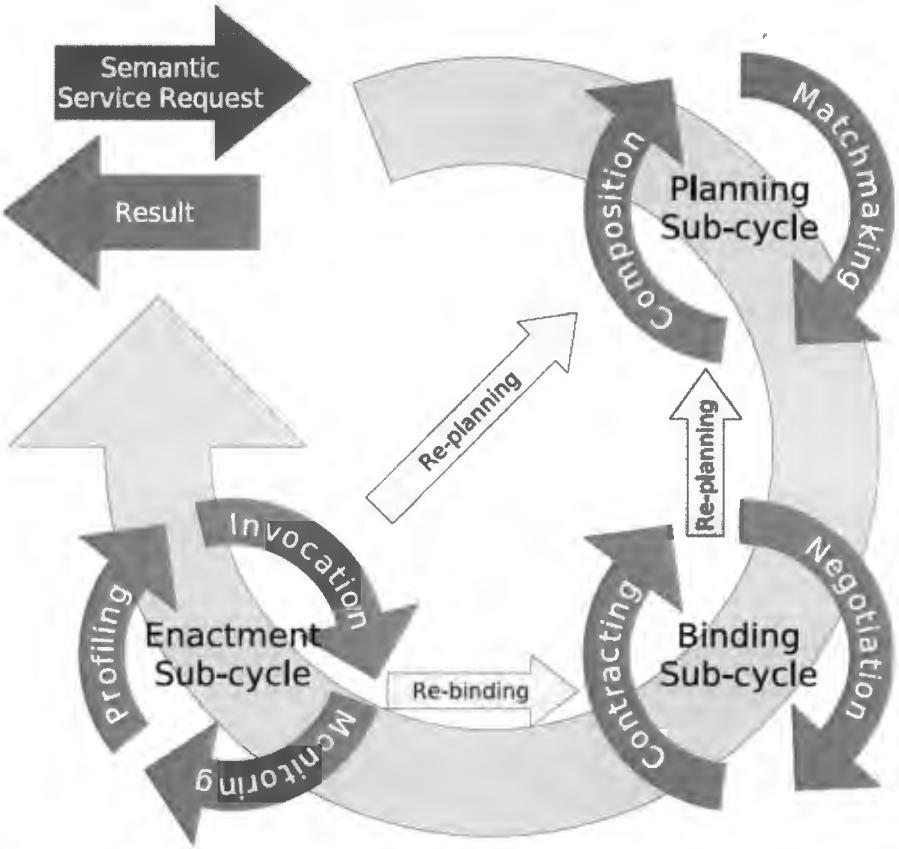


Figure 5: Service-delivery lifecycle (Kuropka et al. 2008)

When a user makes a request, the first step that a service-providing platform takes is to *find* a service that can satisfy the incoming request. If such a service is not readily available, the service space (e.g. a smart campus) searches for those services that can satisfy some aspects of the service request until the request can be satisfied in its entirety. This process is called planning and consists of two main activities: service selection or matchmaking, and service composition. *When a service is found that can satisfy the entire request or some aspect of it, such a service is*

selected for invocation. This is service selection. In the case where multiple services are needed to satisfy the request, when each of those services is found and selected dynamically or manually, this is the process of service composition and such services are selected for composition. Service composition becomes essential when a client's complex request cannot be satisfied by a single service. The focus of this chapter, and the study as a whole, is service selection that, according to Figure 3, is done during the planning sub-cycle of the service-delivery lifecycle.

When a service is found, the service provision platform needs to bind to this service – this entails contracting on some of the non-functional requirements of the request and, negotiating if such requirements cannot be readily satisfied.

After binding with a service, the bound service can now be enacted. Enactment of a service also involves invoking the service, profiling its execution and monitoring it. When errors occur, the service-provision platform may resort to re-binding or, if the error is severe, a complete re-assembling of a service is done by returning to the planning sub-cycle.

In the preceding description of the steps necessary to provide a service, no particular mention was made of the nature of services being provided. However, because the focus of this chapter is service selection, it is necessary to describe the nature of such services. In the case of this study, the services that are the subjects of selection are IoT services, which are discussed next.

3.1.1 Internet of Things services

In the IoT domain, any software component that provides information on an entity or enables controlling of an IoT device is called a "resource" (Barnaghi, Wang, Hensonand Taylor, 2013). A "service" provides a well-defined and standardised interface, offering all necessary functionalities for interacting with entities and related processes. An IoT service is modelled as a virtual concept that is exposed by an IoT resource. IoT services delivery devices mostly have limited computation capabilities, their exposed resources operate in dynamic environments, and

they are less reliable than the general web service (Kleinrock 2013). The model in Figure 6is created based on the SOAP/WSDL and RESTful service technologies (Barnaghi, Wang, Henson and Taylor, 2013). The SOAP/WSDL-based services have strong associations with business process modelling and have been widely adopted in the business world, while RESTful style services are data-centric and have been prevalent in web 2.0 applications due to their flexibility and simplicity.

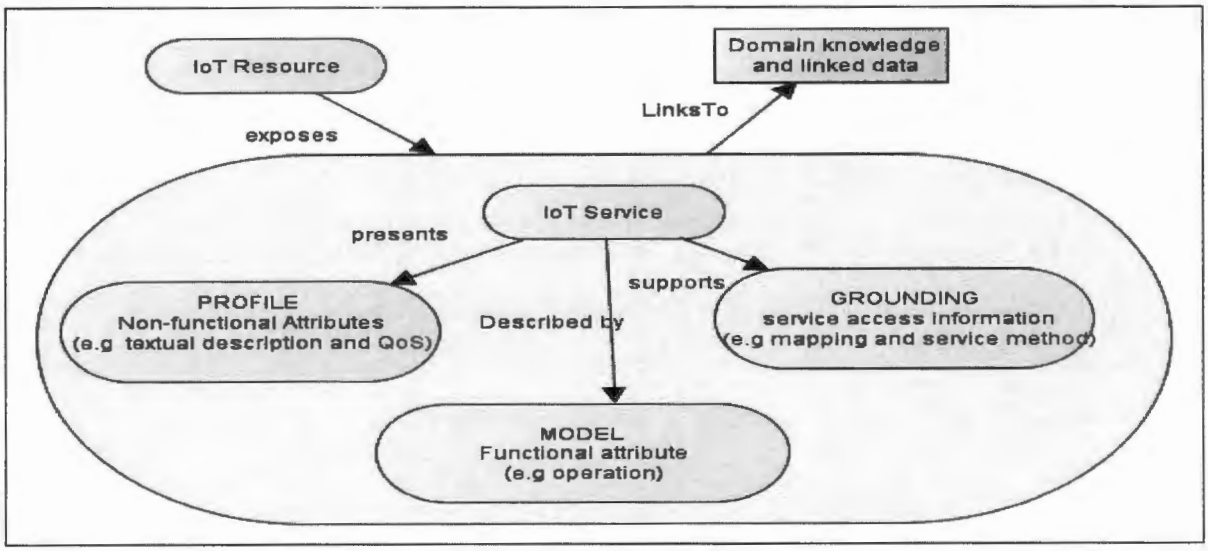


Figure 6: Overview of the IoT description model (Barnaghi *et al.*, 2013)

Figure 6gives an overview of the IoT description model. In this figure, the profile of a service defines the non-functional aspects of the service and it contains properties for linking to semantic concepts in the existing knowledge bases or taxonomies which are essential for service search and discovery. Barnaghi *et al.* (2013) adopted the OWL-S ontology as a semantic model for SOAP/WSDL services, and OWL-S is designed based on the so-called "Profile-Process-Grounding" pattern. In the literature, ontology is presented as a model-based approach, therefore the approach adopted by Wang (2013) solves the complexity that originates from process modelling. In contrast, RESTful services is a simple service ontology that excludes the profile and grounding modelling which is important for service discovery and access.

The model of a service defines the functional aspects of a service. The model is developed by identifying and analysing commonalities between different service technologies. It represents a trade-off between the SOAP/WSDL-based and RESTful services. The concept of grounding provides a mapping between the concepts defined in the semantic description ontology and those defined in the service documents such as WSDL (for SOAP-based services). The mapping concept is optional in IoT services, which usually do not present service documents.

The different approaches to the selection of services, including the IoT services, are discussed in Section 3.3.

3.2 Service-selection architectures

This section presents service-selection architectures, which select services in dynamic environments. To complement these approaches, the literature proposes architectures that represent knowledge and support reasoning during service selection. The architectures here are based on the following components: user preferences, domain specification, storage, scalability and evaluation. These components help to understand the nature of the IoT service environment and are briefly discussed as follows:

- *User preference-based* - users may have their opinions and reasons when choosing services. For that, there is a need for the well-defined interface for the user profile to capture the user preferences. The literature referred to in Section 3.3 shows that more of those systems focused on the non-functional properties. Therefore, we need a system that will define both functional and non-functional properties. User preference is one component that integrates human interaction with the system. The user profile should also capture user context. In this work, context is defined as location, time and duration. It helps to know what kind of a service should be selected and the application must be able to capture the service request in any computer-readable format and redefine it according to how the service is described on the database. IoT services provide well-defined and standardised interfaces offering all functionalities for interacting with entities and related processes (Barnaghi *et al.*, 2013). IoT services are less functional than web services; however, one may want to know about the nature and the state of a "Thing" in

order to continue with querying the service.

- *Domain-specification*- specifies the scope of data and gives awareness in order to visualise a dataset. Smart environments are based on ubiquitous computing, where environments interact with their inhabitants on a device layer (Silva et al. 2008). The data have to be stored intelligently and used for smart monitoring and actuation. This suggestion leaves one with the question as to whether the use of an artificial intelligence algorithm will satisfy users or will it give the users interfaces with which they can express themselves when submitting the job requested. In order to answer this question, there is a need to invoke the user profiling component that captures user preference on a well-defined interface. According to Silva *et al.* (2008), the other challenge faced by smart-environment technologies is to improve decision making, sense making, user experience, and cater for the convenient situations, saving energy, costs – and other concerns that may arise as needs increase and technology improves. Based on the nature of smart-environment technologies, it is hoped that it will improve the quality of life and reduce the ecological impact of humankind, since it eliminates human involvement. This discussion shows that each domain has its own expectations, requirements and challenges.

- *Storage* - UDDI enables businesses to publish and discover service listings and define how the services or software applications interact over the Internet. UDDI is the commonly used storage mechanism in SOA and web services. A smart campus has the distributed resources and services. However, according to Manikrao and Prabhakar (2005), there is a need to get a storage mechanism that will be able to integrate services logically to make the concept. Therefore, from the literature review conducted in Section 2.3, cloud storage is the centralised database where data is stored in virtualised pools of storage which are generally hosted by third parties (Sukhamrit, Kuljitand Dilbag, 2012). Cloud storage is a model of networked enterprise storage, where data is stored in virtualised pools of storage that are generally hosted by third parties. Physically, the resource may span multiple servers and multiple locations. The security of the services depends upon the hosting units, and on the applications that leverage the cloud storage.

- *Scalability* - refers to the ability of a system, network, or process to handle a growing amount of services in a capable manner, or its ability to be expanded to accommodate that growth (Lange 2008). Distributed systems have a limited memory and have sharing restrictions. Cloud service is capable of providing online object storage for files, media and functionalities, then deliver them globally or locally depending on the domain specification for each project. The IoT is a global network infrastructure linking the physical and the virtual objects through the exploitation of data capture and its communication capabilities (Lange 2008). Such an environment therefore demands large data-storage facilities and the sharing of resources and services.

- *Algorithm* - the importance of the algorithm is to support recommendation, since the selection of services will be done based on user preference for a specific need. Singh and Hunhs (2005) indicated that users save time by using the recommender system that helps them to choose from a variety of options. The purpose of recommender systems is to pre-select information a user might be interested in (Singh and Hunhs, 2005). Recommender systems make a prediction of user's needs or interests based on user preferences. The heuristic algorithm should support recommendation and dynamic environments.

- *Evaluation* - the evaluation provides proof of the concepts that whether an architectural or algorithm proposed as the solution works better than other solutions reported in the literature, it is worthwhile to check the relationship that exists between the architectural components and how they perform together. Mostly, it is important to select a relevant service that satisfies the user. In order to check this, evaluation should be carried out using metrics such as recall, precision and response time. Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. Precision and recall are usually presented as a percentage or decimal. Response time is the total time interval when a service is invoked until the service is recommended.

- *QoS - is* a non-functional property of a system. According to Yu and Lin (2006), non-functional properties include service tracking. Service tracking is a broker that has a service repository to record all feasible web services the system is aware of non-functional properties comprise three service criteria. Firstly, the *dynamic service composition* is a broker that maintains some predefined business process plans in the process repository. Dynamic service composition can also build new plans or update existing plans based on user requirements and newly discovered services. Secondly, *the dynamic service selection* is the key function of the service broker, as proposed by Yu and Lin (2006). The dynamic service selection QoS-based broker selects services to execute a business process so that the user-defined utility is maximised, and users' QoS requirements are satisfied. Lastly, *dynamic service adaptation* focuses on service failure during a business process execution whereby a broker needs to reconstruct a business process to ensure good performance.

All architectures studied in the literature will further be evaluated based on the architectural components consideration discussed in this section. The next sub-sections look at relevant architecture, discuss its strength, and identify challenges that are in line with the interest of this work.

3.2.1 A QoS broker-based architecture

According to D'Mello, Ananthanarayanaand Thilagam (2008), the architecture proposed in Figure 7 explores the different types of requester's QoS requirements (demands) with illustrations. The architecture proposes the QoS broker-based architecture for dynamic web-service selection that facilitates the requester to specify the user's QoS requirements along with functional requirements. This work presents the mechanism for web-service selection that selects the best and most suitable web service based on the requester's functional and qualitative requirements. From an architectural perspective, the QoS broker-based architecture is the middleware that can be implemented as a web service. The QoS broker-based architecture can register and edit QoS property values in the QoS registry. The QoS registry provides facilities to

search price, performance and the requester's response-sensitive QoS information on web services.

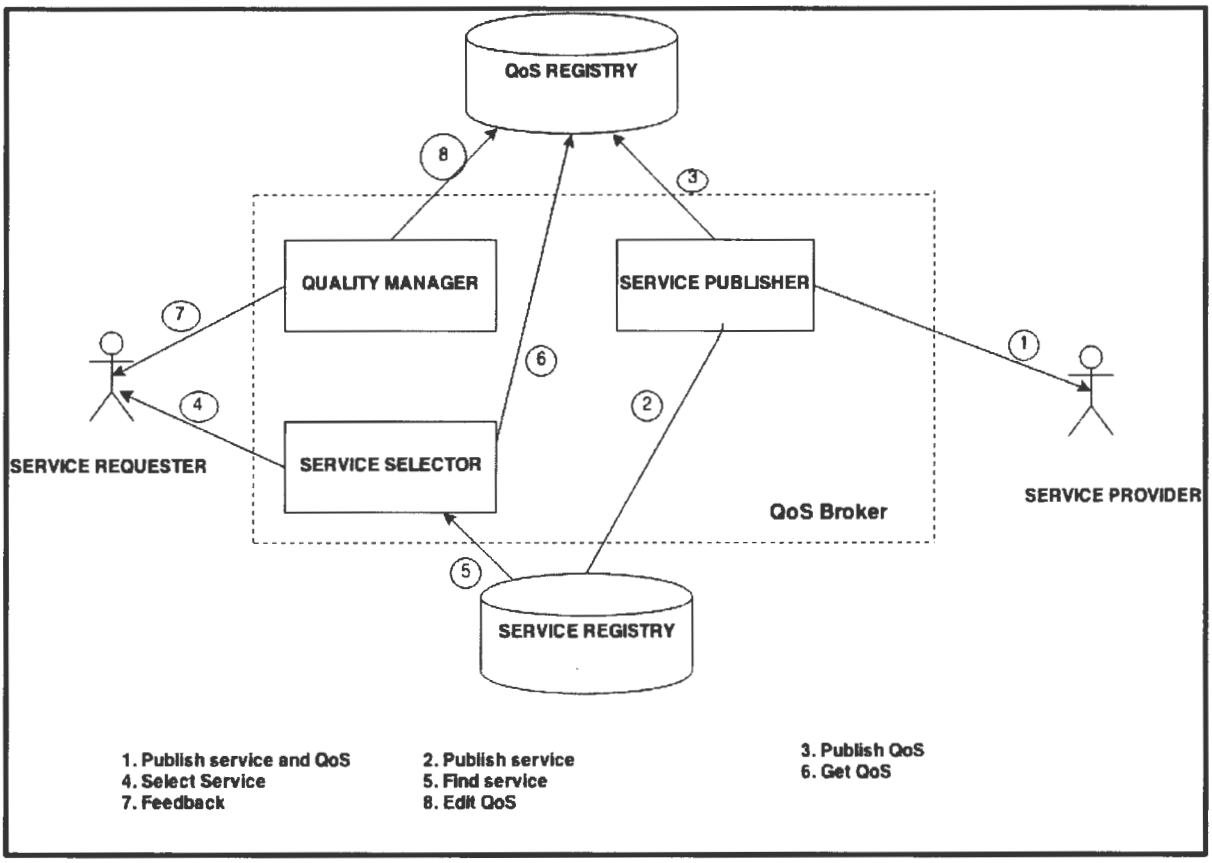


Figure 7: A QoS broker-based architecture (D'Mello et al. 2008)

This architecture is built on basic web-service architecture. The contribution is on the broker side; the architecture considers the QoS for both broker and registry when submitting a request. This broker encapsulates three components and these are the service selector, the service publisher and the quality manager. This architecture demonstrates the importance of QoS in distinguishing functionally similar web services.

3.2.2 The semantic broker-based web-service architecture

D'Mello and Ananthanarayana (2009) proposed a semantic mechanism for the web-service selection process as indicated in Figure 8. The semantic mechanism distinguishes the meaning of the web services that are considered similar; the differentiation is based on the QoS and business

offering (BO). The semantic broker-based web-service architecture recommends the best match for the requester, based on the requested functionality, quality and BO. D'Mello and Ananthanarayana (2009) designed the semantic broker that enables the provider to advertise the service. The semantic broker creates an OWL-S service profile consisting of information related to functionality, quality and BO. After the service advertisement, the broker computes and records matchmaking information. The matchmaking information improves the performance that includes service query time of the discovery and selection process. The broker reads requirements from the requester and finds the best profitable web service by matching and ranking the advertised service(s) based on functionality, capability, quality and BO.

The semantic broker system was implemented for the domain of shopping services in Figure 8to prove the importance of QoS and BO in the service-selection process for the service binding. This architecture does not specify the user's QoS and functional requirements.

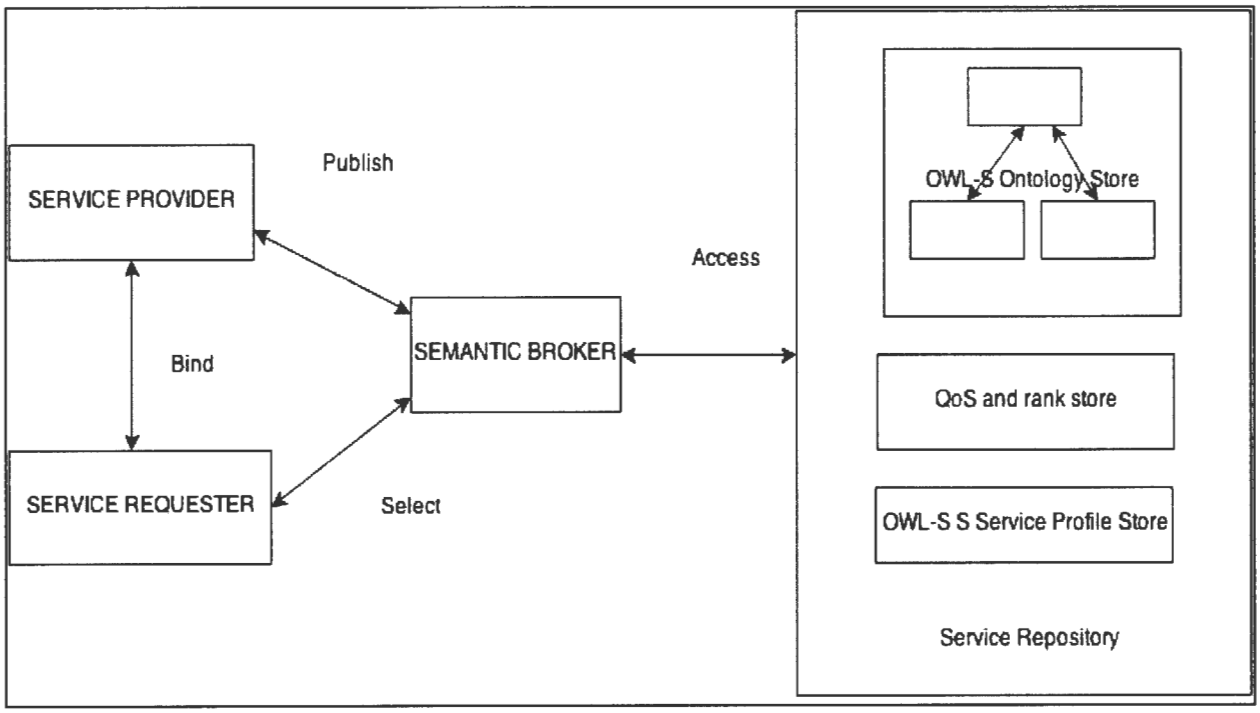


Figure 8: The semantic broker-based web-service architecture (D'Mello and Ananthanarayana 2009)

3.2.3 Architecture of web-service selection frameworks

Choosing one service among similar services that match the user's requirements, which include services providing similar properties, capabilities, interfaces, and effects, is not easy and necessitates the use of an intelligent decision-making framework. However, Manikrao and Prabhakar (2005) presented the design of a dynamic web-service selection framework that makes use of a semantic matcher to support matching and the composition of software services. The framework also uses a recommendation system that helps a user to select the best service that matches his requirements, as indicated in Figure 9. The recommendation system results in the evolution of the framework to adapt to users' requirements dynamically. The recommendation system is based on user-feedback and collaborative-filtering techniques. It helps the user in selecting a web service from a set of similar services. Collaborative filtering algorithms recommend all items that are similar to the given item. However, this framework needs to consider other recommender systems to improve on a well-defined user profile. A well-defined user profile helps with matching service descriptions.

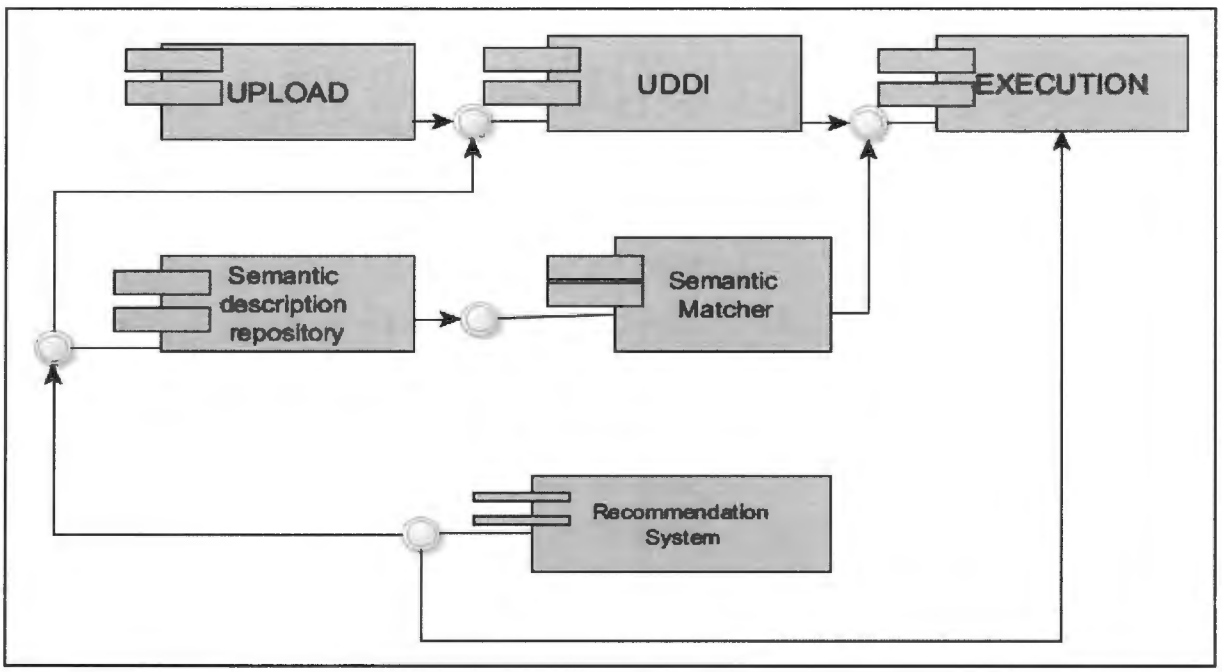


Figure 9: Architecture of web service selection framework (Manikrao and Prabhakar, 2005)

3.2.4 QoS Broker Architecture

Integration efficiency and flexibility are critical for services composition. For web services that provide similar functionality, QoS is the main factor to differentiate between the web services. The overall QoS of a business process must meet a user's requirements. Lin and Tao (2006) proposed a broker-based framework to facilitate dynamic integration and adaptation of QoS-aware web services with end-to-end QoS constraints, as indicated in Figure 10. The functions of a dynamic broker include service collection, selection, composition and adaptation. The motivation behind QoS broker architecture is to deploy a broker that can make selection decisions for the business process requestors and thus simplify enterprise system engineering. The focus is on broker components, and these are service info manager, selection manager, composition manager and adaptation manager, as discussed in the introduction to Section 3.1. The algorithm derived from the architecture for service selection can handle only one QoS constraint and a single point of failure. This work achieves collection, selection, composition and adaptation, but there is still a need to define the user profile when accessing the service.

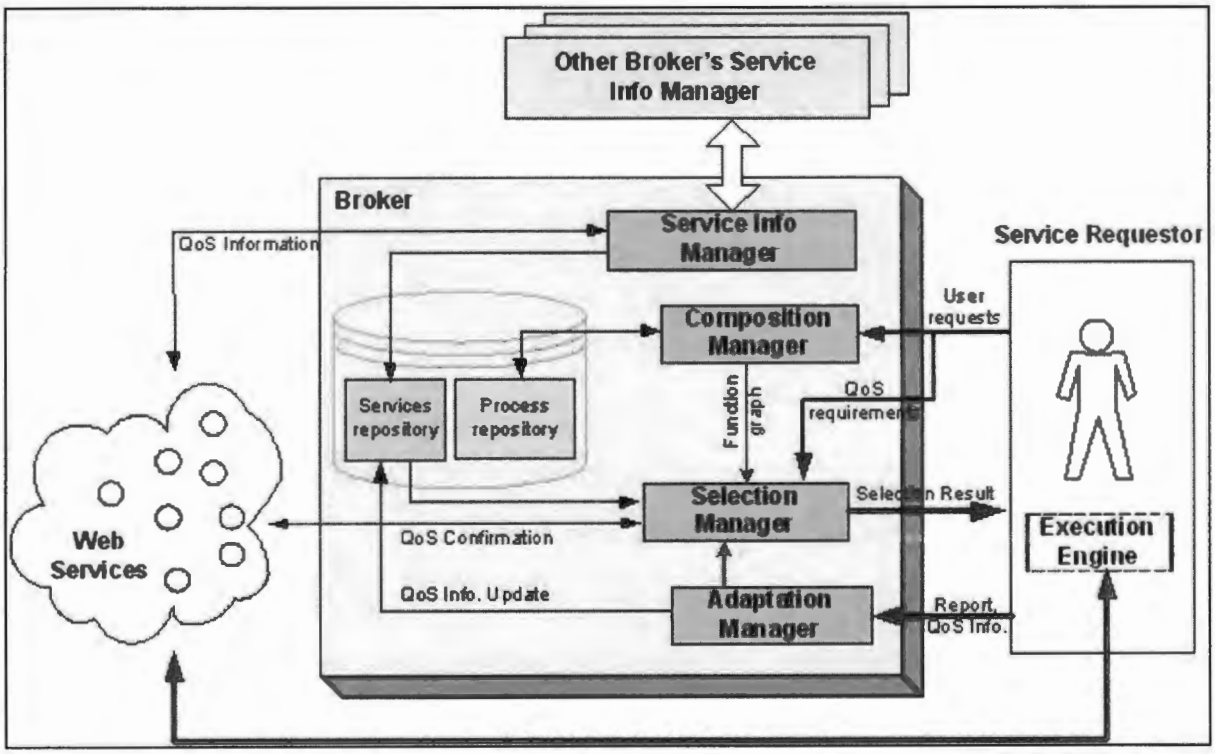


Figure 10: QoS broker architecture (Lin and Tao 2006)

3.2.5 Service-selection architecture

The challenge of service selection in a dynamic environment arises through the heterogeneity, distribution and sharing of the resources in different virtual organisations. The task of efficiently assigning requests to appropriate users also becomes a challenge (Ludwig and Reyhani 2006). The selection process should consider the volume of incoming requests. Owing to the advantages of virtual organisations, industries, scientists and engineers are rapidly deploying the cloud computing paradigm. Cloud computing gathers geographically dispersed users into a new generation of high-performance applications to be developed, to solve computational challenges and to have access to the facilities (Buyya and Sulistio 2008). In order to ensure that the available services are efficiently assigned to the high volume of incoming requests it is important to have a robust service-selection algorithm. The robust service-selection algorithm should not only increase access to the distributed services, promoting operational flexibility and collaboration, but should also allow service providers to efficiently scale access to meet a variety of demands while adhering to certain current QoS standards. Guha (2009) has proposed service-selection architecture, as indicated in Figure 11, to control the incoming volume of requests. This architecture uses a service registry to store a service along with its service identity, the link to locate the service, a brief description of the service and the QoS parameters of the service.

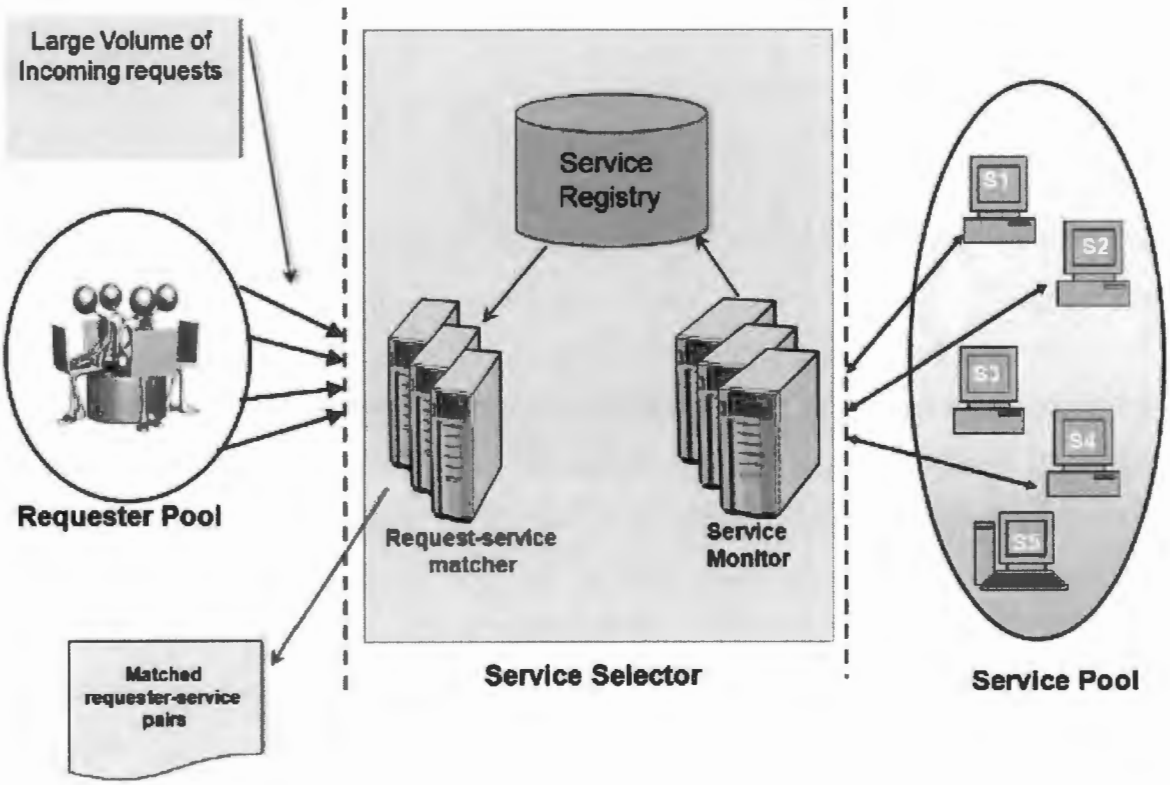


Figure 11: Service-selection architecture (Guha 2009)

3.2.6 Internet of Things system considerations

According to the definitions, characteristics, applications, challenges and architectures studied in this chapter, the following requirements should be taken into system consideration when selecting IoT services:

- Select a service that best satisfies the user as a candidate for IoT service composition.
- Get a user request in whatever acceptable form and break it down into required user tasks and preferences, which can then be easily mapped to the services existing on the cloud.

3.2.7 Summary of the architectural analyses

A summary of the service-selection architecture is shown in Table 2. The literature studied shows that service-selection architectures have covered selection based on QoS. QoS constraints cannot be neglected when selecting a service to specify non-functional requirements for the system. It is easy to tell if scalability is not a challenge when a domain is specified. Most of the architecture conducted on this study selects web services, and little work on the literature has been done in selecting IoT services. Cloud storage has been used by various platforms like Google, Dropbox and other platforms, and storage is never a problem except that it may be costly. To make virtual integration of the service possible, cloud storage must be adopted. It is important to specify the kind of services to be selected. Knowing what kind of services is to be selected helps in defining the interface and knowing how the services must be stored.

Table 2: Summary of service selection architectures

Author	User profile	Domain	Storage	Algorithm	Scalability	Evaluation	QoS
(D'Mello et al. 2008)	The functional properties are not the focus of this work; however the main focus was on non-functional properties	Does not specify domain	Uses UDDI	Uses WSDL	Web services are scalable depending on the domain	√	√
(D'Mello and Ananthanarayana 2009)	Uses QoS requirements specified by a user	√	Uses UDDI	Uses WSDL	√	√	√
(Manikrao and Prabhakar, 2005)	Uses collaborative filtering	Does not specify domain	Uses UDDI	Uses WSDL	Web services are scalable depending on the domain	√	√
(Lin and Tao 2006)	Uses QoS requirements	√	Uses service	√	√	√	√

			repository				
(Guha 2009)	Focuses on volume of incoming requests		Uses service registry	√	√	√	√

The other observation was on the user profile that was not the focus of the architectures. The architecture to be proposed should define the user profile because of the type of services that are selected.

3.3 Service-selection approaches

Service discovery finds or locates service implementations that meet a specified condition while service selection deals with choosing a service implementation from the located services. Service discovery is a prerequisite for the selection process, but selection is the main challenge that needs to be addressed for retrieving web services successfully (Sathya, Swarnamugi, Dhavachelvan and Sureshkumar, 2009).

Service selection can fall into either design-time (static) or run-time (dynamic) services. Simple service selection involves mapping a set of services to a service, while general service selections map a set of services to a ranking of the services in that set. Therefore, a semantic matcher matches the user request with the service description and locates available services that match requirements. The properties of messages include parameters passed (data type, language, unit and business role) and message types (serviceability, provider type, and purpose and consumer type).

Service-oriented architecture provides several competing services and allows users to select available services. Those services can be selected and integrated either statically or dynamically (Yu and Lin 2006). The complexity of web-service selection includes a large number of atomic services integration and performance requirements. Service consumers not only expect the

service to meet functional aspects but they also require services to meet non-functional aspects and properties – that is, QoS (such as service reliability), security, trust and execution cost.

Existing approaches in helping a user to compare and select infrastructure services in cloud computing involve manually reading the provider documentation to find out which services are most suitable for hosting an application. Therefore, selecting and composing the right services with which to meet the application's requirements is still a challenge. Web services are typically highly configurable and a service requester often has dynamic preferences for service configuration. Services composed need to be planned in an optimised way. There may be no single web service that directly offers the desired functionality. A combination of web services may need less investment or capabilities than a single service. In most cases, providers compose web services in order to offer the composite service as a new web service. The service has to be selected in an environment consisting of multiple functionally equivalent operators, but with possibly different implementations and time-varying resources (Lamparter, Ankolekar, Studerand Grimm, 2007).

Personalisation could be any information that can be used to adapt the interaction of a user with a system or service to the needs and preferences of the user or user group. Personalisation of a user's information must be well defined and made available for the context-aware system (Jembere et al., 2006). Most of the selection criteria in the literature are based on description, semantics, quality, rating, effectiveness, scalability of the service and other aspects. Several service-selection techniques and algorithms are proposed in the literature.

With a thorough study of service-selection processes in the literature, the following service selection approaches were identified:

- The multi-agent approach,
- The ontology-based approach,
- The QoS-based approach ,
- The functional-based approach, and
- The user-centred QoS-based approach.

3.3.1 Service selection based on the multi-agent approach

This approach, as proposed by (Maximilien 2004), uses the ontology for QoS and a new model of trust. The approach gathers and shares a service based on ratings; even its user preferences are based on ratings. This approach assumes that the system should give an empirical basis for the service for selection. These ratings are made to be quality-specific through monitoring and user input. Agents used in this approach show that they are able to dynamically adjust their trust assignment and select a service that best satisfies a user.

The challenge associated with this approach is that a user is forced to make an ad hoc decision about the service requested. The selection is based on how a given service behaved previously (as deduced from ratings). However, this approach depends on users sharing their experiences. The approach should be able to introduce and suggest a new service. The service that performed poorly previously is given a chance to improve in the next request.

The challenge identified in this approach according to the focus of this research is the design of a system that would be able to handle user preferences in a machine-readable format such that cloud workflow engines will be able to process them, regardless of whether the preferences are explicitly given by the user or leant from user-session data.

3.3.2 Service selection based on ontology

This approach is based on semantic matching of each service. Semantic matching focuses on meanings behind every service comprising the repository. Services may differ in their syntax but if they serve the same purpose or functionality, they will likely be recommended. Some ontologies use models and those models are supported by QoS properties like name, category, data type, relationship, priority, dynamic attributes and other properties. Maximilien (2004) proposed QoS ontology that let service agents match advertised quality levels for its consumers with specific QoS preferences. Much such ontology has been proposed in the literature.

DAML-QoS complements DAML-S by providing a better QoS metrics model (Zhou *et al.* 2004). This approach presents a matchmaking algorithm for QoS property constraints and describes different matching degrees. This work facilitates the QoS selection between similar semantic service advertisements. The metric ontologies may also provide a powerful solution for measurement organisation for monitoring and billing against the agreed upon SLAs. The development of QoS ontology allows for a new role of QoS designer who designs customised QoS properties, and selects or invents the suitable QoS metrics for the QoS properties (Zhou *et al.* 2005). It is the service QoS designer's task and the web services vendor's task to define the available property types, their domain constraints and range constraints. The basic profile contains a response time that is the total time needed by the service requester to invoke the service, the cost associated with the execution of the service, and reliability. This profile corresponds to the likelihood that the service will perform when the user demands it; and this function of the failure rate and throughput represents the number of web service requests served in a given time period. Semantic web technologies do not have the advantage of a good machine that is able to understand, be interoperable, be unambiguous, and have better extensibility because they are based on XML.

OWL-Q ontology is proposed by Kyriakos and Dimitris (2007). This approach solves a problem of web-service registries returning many functionally equivalent web services advertised for each user request. Therefore, the semantic description of QoS for web services was proposed. QoS is used for distinguishing between functionally equivalent web services.

QoS-based web services discovery algorithms fail to produce accurate results because they rely on either syntactic or semantically poor QoS metric descriptions. Hence, they cannot infer the equivalence of two QoS metrics based on descriptions provided by different parties. This approach does not incorporate evaluation of metric matching in order to show their performance and accuracy.

(Damiano, Giallonardoand Zimeo, 2009) proposed a query language that is used to define complex constrains and it is called onQoS, which captures QoS requirements. The language that was used before was based on SPARQL, whereas onQoS is able to select services based on QoS requirements and specifications. This approach focuses only on improving quality of service language, and presents how the user defines themselves on semantic webs.

According to Tran (2008), there is a need to distinguish and rank web services that have similar functionalities. QoS is often used as a main factor in distinguishing the quality of web services. Tran (2008) proposed a web-service QoS-based ontology (WS-QoS Onto) that can support describing QoS information in detail and facilitating various service participants expressing their QoS offers and demands at different levels of expectation. This approach needs to develop a ranking algorithm for web services, basing it on the QoS description specified by the QoS ontology.

The challenge identified in this approach, according to the focus of this research, is the need to find a mechanism to query for inferred preferences from preference repositories, and ask them for a personalised selection of services. Another requirement gap is finding a system that should support personalisation in dynamic environments, as well as both functional and non-functional requirements.

3.3.3 Service selection based onQoS

SOA enables a multitude of service providers to provide loosely coupled and interoperable services at various Quality of Service (QoS) and cost levels in a number of service domains. The QoS-based mechanism is a non-functional-based service selection approach that evolves trust computing and market-oriented computing development. Ahsan (2006) defines QoS as the collective effort of service performance that determines the degree of satisfaction from a user about the recommended service. QoS has non-functional constraint requirements that must be met during the process of selecting services. Those are constraints such as reliability, response time, throughput and integrity. The QoS value from the service consumer's perspective can be

positive, negative, close, or exact (Sathya et al. 2009). Some functional properties make use of domain ontology. To provide the consumer with the requested service, non-functional properties make use of QoS ontology. QoS has non-functional constraint requirements that must be met during the process of selecting the services. The QoS constrains are reflected in various parameters that the provider can monitor during service invocation and are used to evaluate the quality level. Considering the relevance of the study, some of those constraints are reliability, response time, cost, throughput, integrity and platform/API. The QoS constrains are briefly discussed as follows:

- *Integrity* - is a degree of trust that is expected from a service provider for reliability and availability purposes (Lee *et al.* 2001). However, the user can check whether the recommended service matches the job submitted when the system claims to be trusted.

- *Reliability* - is the ability of a system or component to perform its required functions under stated conditions for a specified period (Rosenberg *et al.* 1998). Reliability makes sure that the system has integrity.

- *Response time* - according to the focus of this research, response time is the total amount of time taken to recommend the service. It consists of execution time and network-transmission time. The job-execution time depends on the workload and system performance, and can be estimated using existing performance-estimation techniques. The network-transmission time depends on network latency and the size of input data. Response time can be predicted exactly and simply by using processing speed, representing the computing power of the service provider.

- *Throughput* - is the movement of inputs and outputs through a selection process (Cranor *et al.* 2000). It can be described as the rate at which a system generates services per unit of time. Throughput can be measured using a mathematical formula called Little's Law and it is measured in bits per second (bits or bps), and sometimes in data packets per second or data packets per *time* slot.

- *Cost* - is the total amount charged per successful execution (Mazzorana *et al.* 1994). Depending on the formula used, which is likely to be determined by the nature of environments where selection is processed, cost may include data volume transferred (which is currently charged for data space such as kilobytes, megabytes, gigabytes, terabytes, etc.), execution time, and other properties. Cost reasonableness attracts more people that would like to participate in the market.

- *Availability* - is important in presenting services during runtime (Schillinger 2011). The availability is not a problem in environments like the cloud since they provide on-demand services. Services are provided when they are needed, and providers understand the sense of emergency.

- *Platform/application Program Interface (API)* - a user may want to specify the API requirements, especially in the PaaS model. It is straightforward to deploy a Java-based application to Google's AppEngine (Zhao, Ren, Liand Sakurai, 2012).

Yau and Yin (2011) proposed a selection method based on the QoS ranking. This approach uses the results of the last phase to select relevant services according to the functional aspect. The approach has two phases. First, the classification of the data-mining algorithms copes with the web-service environment into the QoS level based on the QoS constraints. Secondly, it composes the best services by means of the services' semantic connections. Although the results showed the optimally heuristic algorithm to be used, there is still a challenge as to services available that may not be known during runtime, and the data mining algorithms assume that the user may like the services based on the QoS constraints.

The challenge with the QoS-based approach is that the service-selection system is not able to differentiate similar services based on their features and QoS parameters. The QoS description can be either semantic or syntactic. Semantic QoS is more concerned with the meaning and description of the service. The semantic approach makes the process of selection difficult for composite service (service composed of other services) while the syntactic approach is more

concerned with the language. The language used for search engines that uses key words to match the request with requested information is based on the QoS syntactic approach.

The system should be able to select a service that best satisfies the user as a candidate for IoT service composition. The challenge associated with the QoS-based approach is that QoS is most used during trial-and-error tests. The QoS-based approach does not address selection adequately for open environments like model-based or trust environments. QoS is based on requirements from the server and needs from the client that does not exhibit autonomic characteristics. QoS does not have enough support to help users define their QoS requirements.

Finding services that are relevant to a service request is the core function of service discovery. However, the way the results are presented to the client is also a matter of great importance. Presenting search results in a ranked order makes service selection easier for the client. There have been a number of works on web-service ranking. Wang (2013) uses the hybrid matchmaking process that works on the set of returned services; it aims to find the services most relevant to the query and rank them in order of relevance.

The other challenge identified in the QoS-based approach, according to the focus of this research, is that it has no consistent way for the consumer to select services because consumers perceive quality through the prism of their own experience, and evaluate those service maps to the specific quality parameters offered by a provider (Khan 2010). Therefore it is not effective enough to do selection based on QoS only. There is a need to support it with the functional approach in order to balance both the system's nature and selection criteria. With an ever-increasing number of functionally similar web services being made available on the Internet, there is a need to be able to distinguish web services using a set of well-defined QoS criteria (Wang 2013).

3.3.4 Service selection based on functional requirements

The functional-based approach provides information on how the system selects services. The functionalities lead to selection criteria that formulate the algorithm to be used during the service-selection process. A service-level agreement (SLA) is used in service selection, but most of the functional approaches are based on artificial intelligence (Moghaddam and Davis 2015). Owing to increasing agreement on the implementation and management of the functional aspects of services, interest is shifting towards non-functional attributes that describe the QoS.

3.3.5 Service selection based on user-centred QoS

The approach based on user-centred QoS is the one proposed by Mobedpour and Ding (2011), whereby experienced users are not the focal point. Instead, the proposal is more expressive and flexible for non-expert users to define their own QoS requirements. The QoS-based approach in this context is designed to help users to find their best matching services using their quality requirements. The system design guides the user through the selection process with sufficient information.

The service-selection mechanism proposed in the user-centred QoS-based approach is based on artificial intelligence and it gives a user an interface to browse to check the available service(s) in order to gain ideas and make a choice. This approach targets non-expert users and supports ranked, relaxed, preference and fuzzy results.

Two challenges have been identified in this approach. First, the challenge associated with this approach is that it assumes that a user is capable of formulating queries for the service-selection process. The other challenge is that there may be no service that matches the requested service from those that are available. Lastly, the challenge identified in this approach is to get a user to request in whatever an acceptable form and break it down into required user task and preferences, which can then be easily mapped to the services existing on the cloud.

3.3.6 Internet of Things system considerations

- Differentiate similar services based on their features and quality of service parameters. Get and use the quality of service parameters in the process of IoT service selection, no matter how and where they are defined.
- Handle user preferences in a machine-readable format such that the cloud's workflow engines will be able to process them, regardless of whether the preferences are explicitly given by the user or learnt from user-session data.
- The algorithm should be able to query or infer preferences from preference repositories and ask them to personalise the selection of IoT services.
- The system should support personalisation in dynamic environments.

3.3.7 Summary of existing approaches

This section presents a summary of the approaches studied in the literature. Table 3 presents the name, the description, the strength and the shortcoming of the approach.

Table 3: Summary of service-selection approaches

Approach	Description	Strength	Shortcoming
Non-functional-based service selection approach (QoS-based)	Approach based on user or system constraints and requirements	Classifies data mining algorithms to cope with web-service environment into the QoS level based on the QoS constraints	Does not have sufficient support to help the user to define the QoS requirements
Multi-agent service selection approach	Gathers and shares a service based on ratings; even its user preferences are based	Gives user an opportunity to gathers and share ratings; its user preferences are also	Depends on users sharing their experiences and does not give a service a chance to

	on ratings	based on ratings	improve
Ontology-based service selection approach	Based on service-selection model and supported by QoS properties	Uses semantic matching that focuses on meaning behind every service composed in the repository.	Does not query for preferences from the repository that stores user preference
User-centred QoS-based service-selection approach	It is an expressive and flexible way for non-expert users to define their own QoS requirements	Proposes an approach that is expressive and flexible for non-expert users to define their own QoS requirements	Does not give a user the chance to request a service in any acceptable format that can be further be broken down into required user task and preferences.
Functional-based service selection approach	An approach which answers the questions on how should service be selected, which forms part of selection criteria	The functionalities lead to selection criteria that formulate the algorithm to be used during the service-selection process	Can be limited by the nature of the environment and the problem intended to be solved by service selection

The approaches studies here show that there is still a need to address the concept of personalisation when selecting services in dynamic environments. It appears that a service-selection approach can be derived from the model (Chen Zhou et al. 2004) or can be derived from the architecture (Guha 2009). The next section presents the proposed approach.

3.4 Proposed approach

Based on the aim of this work, in order to select a mechanism that can be used to select the IoT services environment, the desired mechanism should support the following environmental considerations:

- *Integration* - in order to select services, there is a need to find a storage mechanism that will enable the integration of services from different departments of the smart campus.
- *Scalability* - the mechanism adopted should be able to support a large number of services integrated into a database.
- *User personalisation* - the service-selection system should be flexible to allow users to define the service that corresponds to their profile. User preference is captured automatically.
- *Algorithm* - the mechanism to be proposed should implement algorithm(s) that support(s) user preference and personalisation.
- *Domain* - the mechanism to be proposed should be able to support the smart-campus environment.

3.4.1.1 Evaluation of recommender systems in the IoT

This section evaluates recommender systems based on collaboration between the IoT and recommender system consideration. These considerations are derived from the literature after the discussion conducted for the IoT in Section 2.1, and the recommender systems in Section 2.5. The system of selecting the IoT service should be able meet the following requirement as per component.

A system should be able to do the following:

A. Select a service that best satisfies the user as a candidate for the IoT service composition.
B. Get a user request in whatever acceptable form and break it down into required user tasks and preferences, which can then be easily mapped to the services existing on the cloud.
C. Differentiate similar services based on their features and quality of service parameters, and get and use the quality of service parameters in the process of the IoT service selection no matter how and where they are defined.
D. Handle user preferences in a machine-readable format such that cloud workflow engines will be able to process them, regardless of whether the preferences are explicitly given

by the user or learnt from user session data.

E. The mechanism approach should be able to query for inferred preferences from preference repositories and ask them for the personalised selection of the IoT services.

F. The system should support personalisation in dynamic environments.

The evaluation of recommender systems is conducted in Table 4 and based on system considerations discussed in section 3.4.1.1. Each system consideration is represented by a symbol corresponding to it. A, B, C, D, E and F in the table correspond with each system consideration. At the end of every recommender system, a total number will be discussed further after the evaluation. Where there is a (√), it means the requirement is met, otherwise the challenge will be stated.

Table 4: Recommender-systems evaluation

	A	B	C	D	E	F	Total
Utility-based	√	Measure of howuseful or important an itemset is	If service found not useful will not be suggested	Does not handle user preferences in a machine readable format	The cluster most similar to the user's topic of interest is selected as the relevant subset	√	2
Knowledge-based	√	Suggests services based on conclusions reached on the basis of evidence and reasoning	√	Differentiate similar services based on reasoning (artificial intelligence)	It does not have to gather information about a particular user because its judgements are independent of individual tastes	√	3
Memory-based	√	√	√	Memory-based systems are not always as fast and scalable as we would like them to be, especially in the context of actual systems that	√	√	5

				generate real-time recommendations based on very large datasets			
Model-based	√	√	√	√	It is usually difficult to add data to model-based systems, making them inflexible	√	5

The evaluation results show that the remaining candidate is a hybrid of both model-based and memory-based approaches. According to the classification conducted in the literature the remaining candidate is a collaborative filtering and content-based recommender system. These two recommender systems will further be evaluated in the architecture.

Based on these requirements and the preceding analyses of existing approaches to service selection, this study proposes the use of recommender systems as the mechanisms for service selection. Recommender systems have been criticised for not being suitable for service selection because they do not allow a customised schema in terms of the qualities of interest for different users and data owned (Maximilien 2004). However, Balke and Wagner (2003) showed that selection not only depends on service parameters like execution costs or accuracy, but also on the usefulness of objects or information that a service offers. Furthermore, there is a need to improve the usability of services on the subsequent refinement and integration of presented ways of personalisation into standard interaction techniques. Recommender systems are computer-based intelligent techniques for dealing with the problem of finding appropriate services or products (Park, Kim H., Choiand Kim J., 2012). Recommender systems can be utilised to efficiently provide personalised services in most e-commerce domains with the purpose of benefiting both consumer and provider. Consumers are benefited by allowing them to define and make suggestions related to the service they want, and providers benefit by the increase in sales made possible when more consumers find value in the service presented to them.

Tserpes, Aisopos, Kyriazisand Varvarigou (2012) published a paper that uses the recommender system to select services in service-oriented environments. The paper describes an innovation

that allows the consumer to quantify and aggregate their Quality of Experience (QoE) against the provider's QoS once the service-level agreement (SLA) is concluded.

The IoT environment studied in the literature shows that there is a need to propose the key components of an architecture that will make the service-selection process possible. The architecture needs to be tested by select services using the recommender system(s). The recommender system(s) should be evaluated to prove their accuracy. The effectiveness and performance of the recommender algorithms are usually evaluated using response time, precision and recall metrics (Sukhamrit et al. 2012). Response time sums up time span from the user submitting a job until the results are received. Precision and recall are the basic parameters by which to evaluate the efficiency and effectiveness of recommender systems.

This work will make use of recommender systems to test the performance of the architecture. Chapter 4 proposes the architecture and recommender system from the literature studied in chapter three.

3.5 Summary

This chapter has analysed the existing service-selection mechanisms. The mechanisms identified were multi-agent based, ontology-based, QoS-based, user-centred QoS-based and functional-based approaches. Each approach serves its own purpose of service selection in different infrastructures. The architectures studied in this chapter were QoS broker-based architecture, semantic broker-based web-service architecture, architecture of web-service framework, QoS-broker architecture and service-selection architecture. Most of the approaches studied were based on QoS only. The solution approach indicates the need to find storage that will integrate services.

The next chapter will present the approach proposed for this study. It presents the development of the architecture.

4 Architectural development and design

This chapter presents the development of the architecture for user preference-based service selection. The challenge that is addressed in the research question in Section 1.3.1 requires a mechanism that can be used for dynamic service selection, using user preferences in a smart environment to support dynamic service composition. Section 4.1 presents the use case scenario. Based on the use case scenario, Section 4.2 proposes and discusses the key components of the architecture. Section 4.3 proposes the algorithm derived from the architectural components in Section 4.2. Section 4.4 and Section 4.5 compare the proposed architecture, approaches against existing architectures, and approaches respectively. Lastly, Section 4.6 gives the summary of the chapter.

4.1 Use case scenario

IoT services have "Things" and resource types. The "Things", such as light bulbs, cameras, actuators and doors consist of resource types such as light-switches, air-conditioner controls, door-strings and light-sensors. The "Things" and resources database is stored in the middleware registry. Based on the nature of the IoT services description, a scenario of service request is presented as follows:

The user is looking for a service that can do function Z (e.g. check if the blinds are closed or open). The user may request this service within or outside the premises of the smart campus. The user expects the application to be able to recommend the relevant service(s) to the user.

4.1.1 Assumptions

This section discussed the assumptions on which the architecture is developed. The assumptions are as follows:

- *Type of service* -the type of services to be selected is IoT services as presented in the IoT description model in Figure 6. The services consist of attributes such as thing, name, resource type, service type, hostAddress, parentUID, root, resource type, uuid, version

and description.

- *Storage* -the IoT services are stored in the middleware registry. The middleware registry can be accessed only if it is connected to the Internet.

    - *Scope of the database:* The scope of this work is to use the available data. Since the IoT is a newly introduced paradigm, the available IoT data are limited. In order to test the proposed solution, this work will use the middleware database as associated with the experimental smart environment as created by the CSIR Meraka Institute IoT research group[1]. The services are stored in the internal middleware database and published to the Internet, so that the application can select the updated services. This database setup addresses the dynamic environment.

- *Type of request* - the request should be automated based on dynamic input, such as date-time and current location. If the user preferences were well defined during user-account creation, then the request can be automated.

- *Number of available services* – the number of available services when the application was tested was 50. The number of IoT services is expected to grow each time there is a service added on the middleware.

- *Service-selection process* – the service-selection process should take place on the application layer which the user can view. If the user is within the smart campus, then the services that are close (by physical location) should be recommended to the user. If the user is outside the smart campus, services based on the user's room (user's registered office or room in the smart campus) location should be recommended.

---

[1] The middleware application and database used for experimentation is continuously being extended and enhanced.

- *Smart campus* - this refers to the CSIR campus where an experimental IoT environment is being created.

- *Connection* - the assumption is that there is no congestion on the network.

- *Devices* - the application must support any mobile devices that are compatible with the Android operating system, such as smart phones and tablets. Smart phones and tablets are handheld devices that have few challenges for use anywhere, compared to laptops and desktops.

4.1.2 Selection criteria

Based on the scenario presented in this chapter, here are the two questions that rise:

1. Which service-selection approach will return relevant services based on the incoming request looking at the concrete real-life scenario presented above?
2. Furthermore, will the requesting service within or outside the premises have a relevant effect on recommendations?

In order to address these questions, this section considers details scenario details.

Assume that the user looks for a service to perform Z, and the pseudo code will be as follows:

To do: Z // you want to do Z

Search for a service within Service composition that can do Z

1. *Service found → Do Z //if the service found then return service that can do Z*
2. *Service Z not found //if service is not found*
3. *But Z = {f, g, h, I, J} // Z can be accomplished by a combination of services (f,g,h,I,J)*
4. *Find Service (f, g, h, I, J) // search and combine services that together can accomplish Z*
5. *Service Found → Do Z //service is found and return the service*
6. *Service not Found → Abort ( ) //service not found abort the transaction*

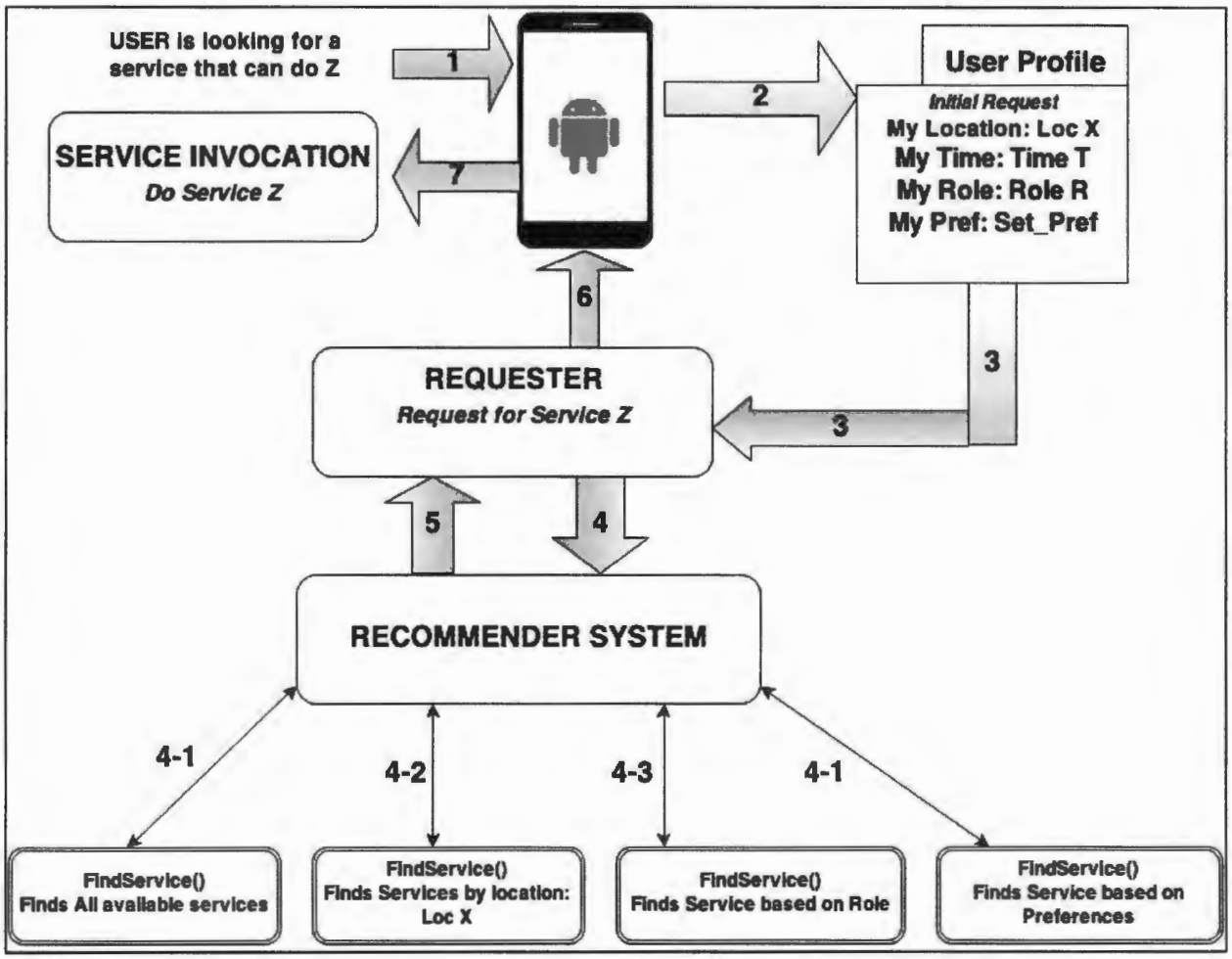


Figure 12: Selection criteria

1. User is looking for a service that can do function Z using a mobile phone.
2. The user sends an initial request for a service and specifies the operation (check if the blinds are open or closed). The current position of the user and the time should be presented as an automated input as part of attributes required for the user profile.
3. The system creates a user profile from all the inputs required and sends the request to the requester component.

4. The proxy receives the request and starts by firstly attempting to list all available services based on the following criteria:

   4-1 Query all available services on the database;

4-2 by availability, according to the current position of the user;

4-3 by availability, based on the user role;

4-4 by availability, based on preferences (all services that can do Z; check status of the "Things").

The application recommends service(s) received from each of the four sub-services (services at the user's current position, services available during the date-time the services were requested, service based on the role and services that were set as preferences to do operation Z (check if the blinds are open or closed)). Nevertheless, these lists are necessarily helpful in executing the operation; the service then computes the intersection of all these three sets of services.

5. The resulting set of services (i.e. those services that are able to do Z, that are available during the specified date-time, and are located at the specified geographical location) is recommended to do operation Z.
6. After receiving the list of services as candidates for the user's request, the operational service passes this list of services to the user, who can choose which service among the recommended services can meet that user's request.
7. The user selects and invokes the service.

The IoT is referred as a self-configuring network in which everyday objects are interconnected. The IoT gets a heightened awareness of real-time events by deploying sensors that capture contextual information about objects (e.g. user preferences) to achieve an enhanced situational awareness (Zhang, Zouand Xiong, 2011).

Figure 13 presents the SOA that captures the interaction pattern among the key components. The service consumer performs dynamic service locating by querying the service provider for a service that matches its criteria. The service provider defines service descriptions and publishes the descriptions of services so that services can be discovered. The interaction between service consumer and service provider is, however, transparent to the user. The contribution of this work is in the proxy component that acts as the mediator between service consumer and service provider. The proxy discovers, matches services and recommends the relevant services to the user.

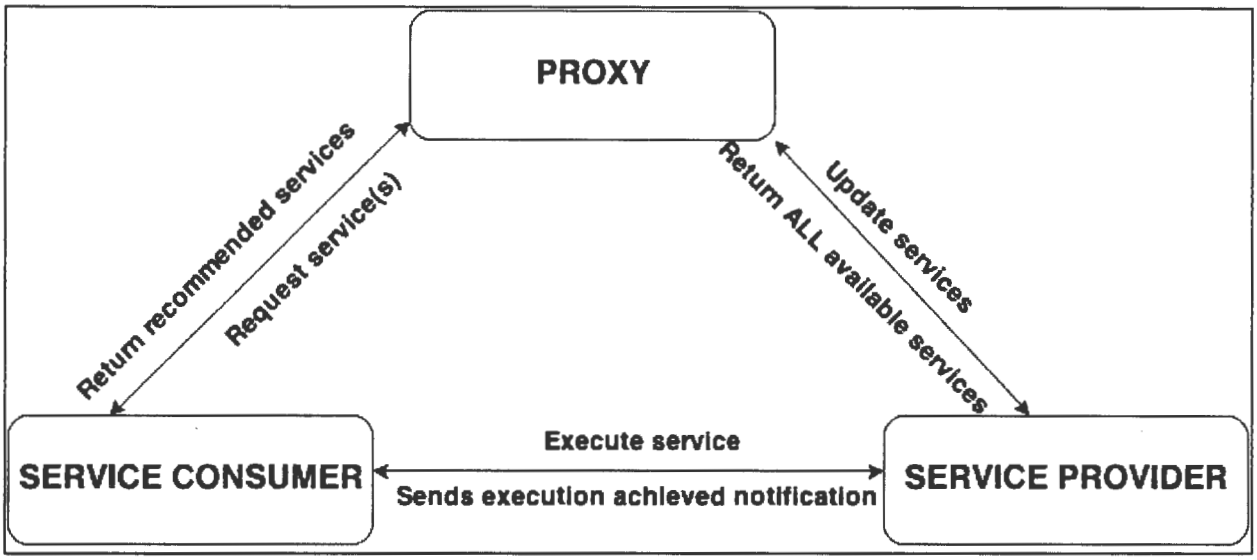


Figure 13: SOA-based Selection criteria

The registry (where services are stored) needs to be clear on how the context is defined and in what condition the services will be recommended. Figure 13 presents a mobile user and an application requesting a service on the proxy. The services requested should be published from a device layer and stored in the registry. The selection takes place in the proxy then returns a set of services to the requester. The requester receives a service and executes it. The execution binds the service requester and the service provider and does the operation.

The next section discusses the proxy component.

4.1.3 Proxy implements recommender systems

Assume there are a number of requests *(R) i.e.* R_1, R_2, R_3..... R_n submitted by a user (u) to the system. Each request has its own user profile. The recommender system makes the search possible for each request without making it transparent that the services requested are provided by different service providers ($P_1, P_2, P_3 \ldots\ldots P_n$).

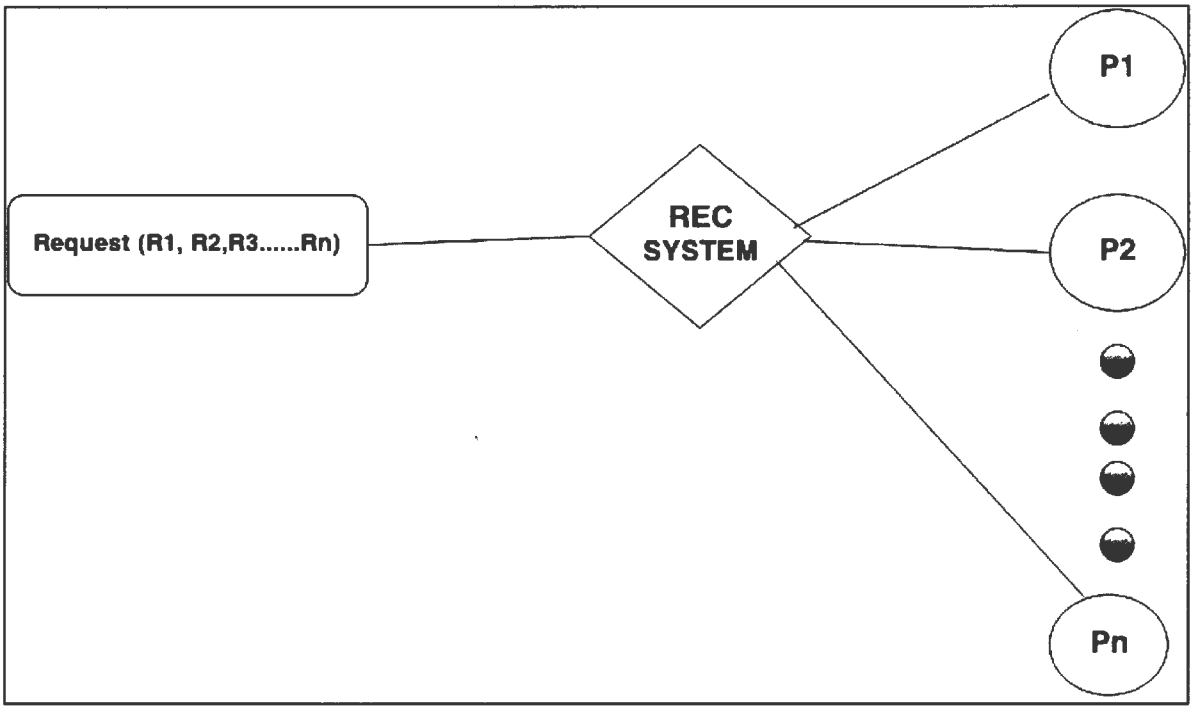


Figure 14: Service requester submitting jobs

The next section proposes the IoT service selection architecture.

4.2 Proposed architecture

This section presents the proposed architecture developed, based on the use scenario in Section 4.1 and system consideration proposed in Section 3.2.6 and Section 3.3.6.

The analyses of the scenario propose the following architectural considerations:

- If the search would be performed using an algorithm, therefore there is a need for a container that will enable the plug-in of an algorithm component within the architecture.
- The search should happen within the registry where all IoT services are stored. In order to make it possible for a user to select a service there must be an entry point for a user to present a user profile.

- A user profile comes with user preferences that contain the user's history of previous likes or what people with the same attributes liked before. In order to simplify processing, the usage history would have to be stored in a usage repository.
- However, if the user is new, the system should allow the user to create an account. Hence, there is a need for a component to store user-account details. Repositories for user profiles, domain knowledge or usage patterns and the generation of new patterns will have to be investigated.
- There is a need to consider attributes of the user profile and how the matchmaking is done when selecting services offered by different client devices. In the case of, for instance, mobile devices with limited capabilities such as battery (last for shorter time) and the Android operating system, the extension of our basic techniques could pave the way to improved usability enabling a user-specific and context-aware selection of appropriate services.
- However, the selection process would need to monitor services according to their description and other properties. Service monitoring cooperates with the service selector in building the recommendation based on the selection criteria.
- The storage layer focuses on how the services should be stored. Each department registers and stores the service in the middleware. The registry deployed to the cloud gets well-defined services to suggest for the user. Since the services are dynamically composed, the registry should be automatically updated and be able to suggest and advertise new services or improved services.

The system considerations are as follows:

- Select a service that best satisfies the user as a candidate for IoT service composition.
- Get a user request in whatever acceptable form and break it down into required user tasks and preferences, which can then be easily mapped to the services existing on the cloud.
- Differentiate similar services based on their features and quality of service parameters to get and use the quality of service parameter in the process of IoT

service selection, no matter how and where they are defined.

- Handle user preferences in a machine-readable format such that cloud workflow engines will be able to process them, regardless of whether the preferences are explicitly given by the user or learnt from user session data.
- The algorithm should be able to query for inferred preferences from preference repositories and ask them for a personalising selection of IoT services.
- The system should support personalisation in dynamic environments.

Based on the architectural and system considerations, the architectural key components are proposed. The proposed architecture consists of three layers, which are the application layer, the middleware layer and the registry layer.

4.2.1 Application layer

The application layer presents components that are transparent to the user. The application layer consists of three sub-components and those components are briefly discussed as follows.

- *User registration* - this is a user interface that registers the records for a user by completing required fields. This sub-component helps in identifying the user and traces a user's usage history and preferences. The assumption is that if the user prefers the service, the user might like the same service in future.
- *Service search* - search is a user web interface sub-component that allows a user to define a service the user (user may or may not be a person)is looking for. The job is submitted and that enables searching for a requested service.
- *Serviceview - is* a sub-component that gives the list of recommended service(s) based on the attributes used on resource search.
- *User View* - is a sub-component that allows a user to choose a service among recommended services and invoke the service.

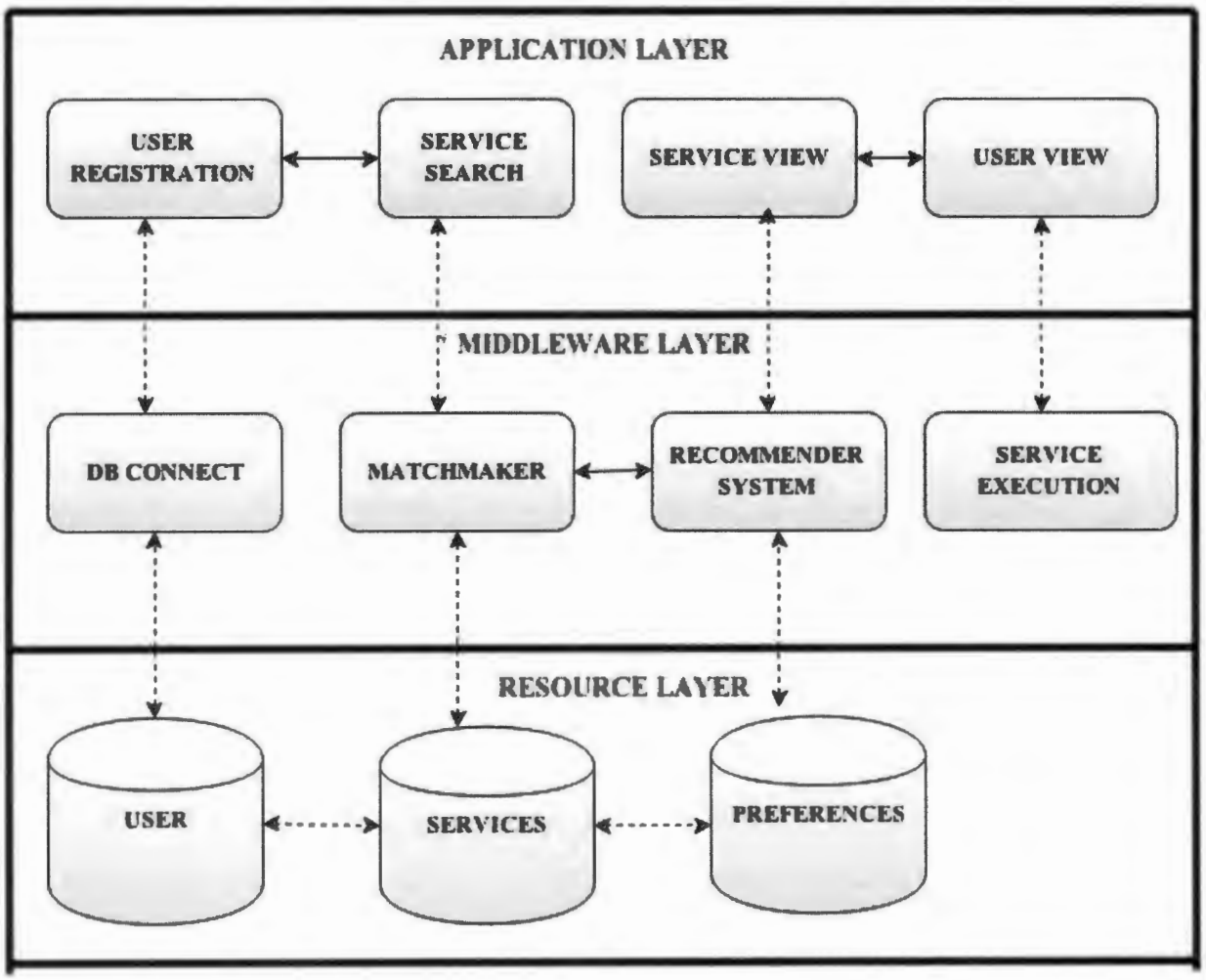


Figure 15:IoT service-selection architecture (ISSA)

4.2.2 Middleware layer

The middleware layer consists of four sub-components that orchestrate the selection of services. Middleware sub-components are briefly discussed as follows:

- *DB Connect - is* the sub-component that connects databases stored in registries to be utilised by the user.

- *Matchmaker* - matches available services using recommender systems techniques.
- *Recommender system* – this is a plug-in component; it allows the system to select services based on plugged-in recommender systems techniques.
- *Service execution* - is the sub-component that invokes the services that will be chosen by a user among the recommended services.

4.2.3 Resource layer

The resource layer consists of three sub-components that store the database of user, preferences and services.

- *User database* – this keeps the records about the user (forms part of user profile).
- *Resource database* - this keeps records about resources and Things. This sub-component keeps real-time records; it keeps records updated and ready to be utilised.
- *Preference database* - this keeps records about what a user likes and the user's usage history.

Having proposed the ISSA, the next section presents the ISSA algorithm.

4.3 Proposed ISSA algorithm

The architecture proposed in Section 4.2 needs to be proven in terms of its effectiveness and performance within a smart environment. Hence, the second research question from Section1.3.1 states:

"Having established the key components necessary for an architecture to support dynamic selection of IoT service, what preference based algorithm that integrates user preference works best?"

However, this chapter addresses the research question by deriving the recommender algorithm from the ISSA components and implementing the algorithm. The ISSA algorithm expands the *Matchmaker* component. The *Matchmaker* component processes matching, using the technique

that will be plugged into the *recommender system*. However, *Matchmaker* has its own skeleton service-selection criteria. The pseudo code of the selection criteria is as follows:

Given

P_s = {*current_loc, sname, uuid, stype, rtype, operation, physical_loc, thingtype, status, IP_address*}

P_u= {*Current-loc, uname, email, role, room, date-time, Set_Pref{stype}* }

1. List attributes to be matched

   Common attributes= $P_s \cap P_u$= {current_loc}

2. Matching the first set of attributes

   If P_u = {current_loc = "120$^{\theta}$:40$^{\theta}$"}

   Return $S_{pset= range}$ {current_loc = "100$^{\theta}$:80$^{\theta}$"}

   2.1 Are these services in a close proximity?

   Apply rules

   BEGIN FOR: All services in S_{pset}
   BEGIN WHILE: x<5m
   Distance threshold x is set
   BEGIN IF: results more than n
   Reduce threshold
   ELSE IF no results
   Increase threshold
   END IF;
   END WHILE;
   END FOR;

   This step returns a second set of services S_{pset2}

3. Preference-based filtering

This section will match service with an attribute specified on the set_pref static attribute that is stype.

Match set_pref with each service in S_{pset2}

BEGIN FOR: All attributes in set_pref
    BEGIN FOR: All services in S_{pset2}
        BEGIN IF: value of attribute=value of same S_{pset2} attributes
        Include S_{pset2}
    ELSE
        Exclude S_{pset2}
        END IF;
    END FOR;
END FOR;

Based on the use case scenario, phase three matches services based on preference filtering. Matching services based on preferences may be a challenge. The challenge rises when the *Matchmaker* component has to identify the attributes that are composed in the "set_pref" attribute. The "set_pref" attribute needs to be modelled in order to return the relevant services. The next section presents the modelling of both user and the service profile.

4.3.1 User profile and service profile modelling

The summary of the service-selection approaches in Section 3.2.7 showed that other architectures presented little research into user and service profile modelling. The use case scenario consists of the important attributes that need to be taken into consideration during the selection process.

The user (User Profile) is looking for a service (Service Profile) that can do function Z (can check if the blinds are closed or open). The user may request this service within or outside the premises of the campus. The user expects (User Preference(s)) the application (Attributes on User interface) to be able to predict and recommend (Real-time recommendation) services to the user.

This section, therefore, presents the modelling of the user and service profile during the service-selection process. The user and the service are considered to have both dynamic and static profile attributes. An example of user-profile static attributes includes name, email, room, role and set of preferences as indicated in Figure 16 while service-profile static attributes include service name, unique ID, service type, resources type, and other attributes as indicated in Figure 16. Dynamic user-profile attributes include current location and date-time, while dynamic service-profile attributes include current location, IP address and device status.

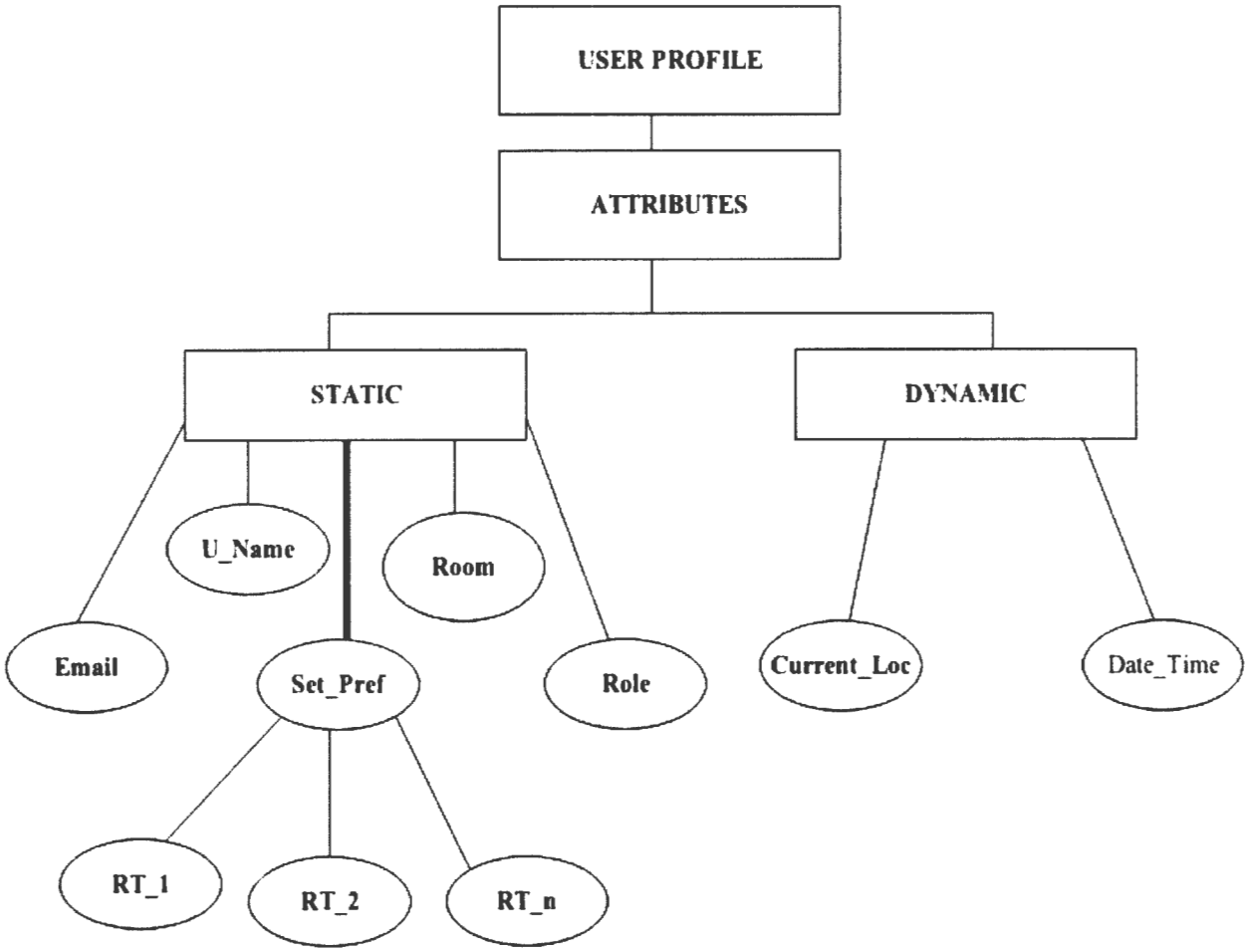


Figure 16: User profile

For discovery of service, the user profile has to be matched against IoT service profiles. The service profile is as represented in Figure 17.

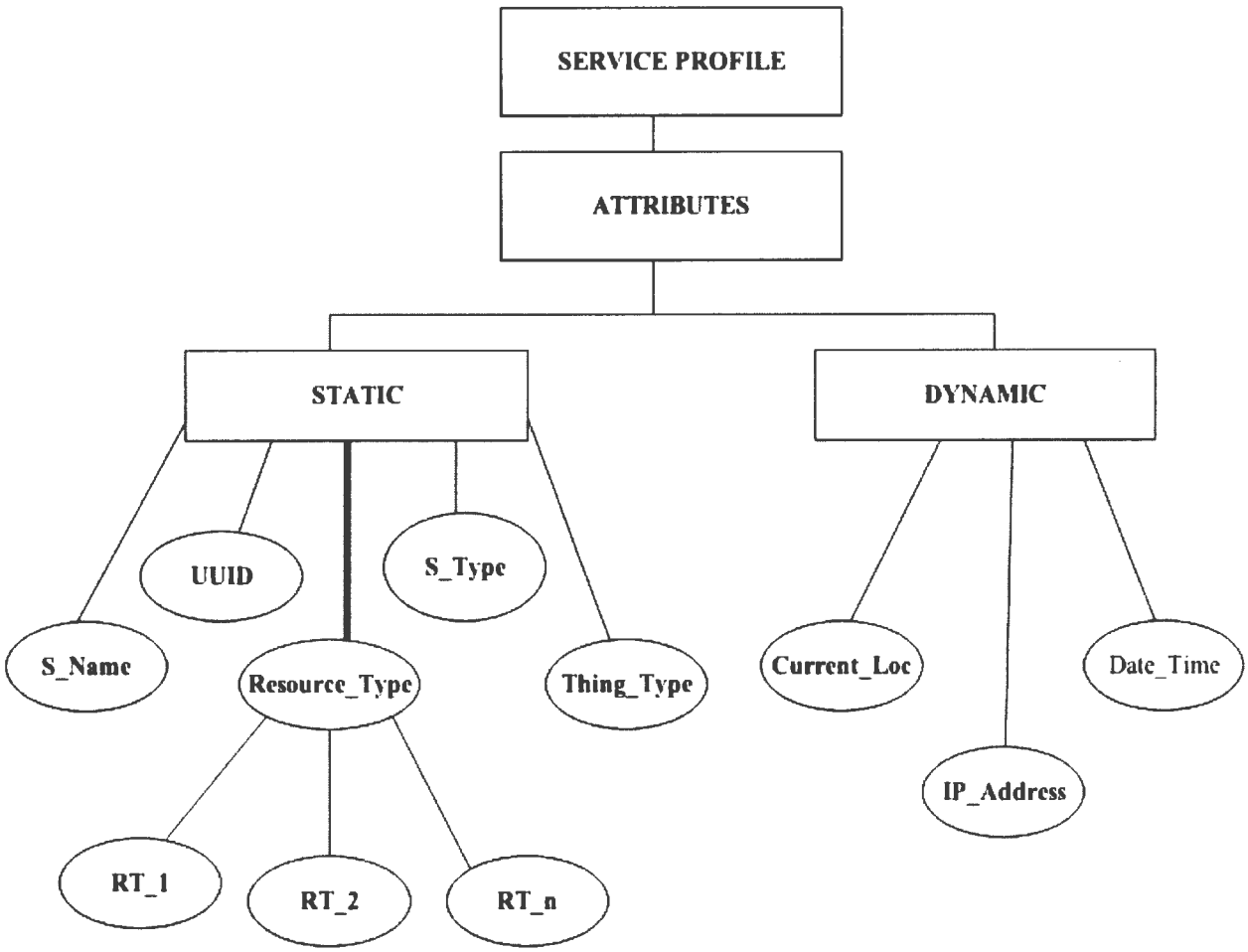


Figure 17: IoT Service profile

In order to generalise the user profile and the service profiles, this work makes use of mathematical models. Let the set of static attributes that can be used to represent a user profile beA_{stat}. Therefore,

$A_{stat} = \{x \mid x \text{ is a static user attribute}\}$	Eq. 1

Since the user profile also comprises dynamic attributes, the set of dynamic attributes is representing asA_{dyn}.

$A_{dyn} = \{y \mid y \ is\ a\ dynamic\ user\ attribute\}$	Eq. 2

The user profile is, therefore, made up of the two sets of attributes. i.e.

$P_u = A_{stat} \cup A_{dyn}$	Eq. 3

The service as well has both dynamic attributes and static attributes. In this paper, the IoT service is considered to be a web-service endpoint which has direct access to one or more IoT resources (Thoma, Braunand Magerkurth, 2014). Such a service has a set of dynamic attributes and static attributes which are represented as B_{dyn} and B_{stat} respectively. The two sets are then represented as follows;

$B_{dyn} = \{q \mid q\ is\ a\ dynamic\ service\ attribute\}$	Eq.4

And,

$B_{stat} = \{r \mid r\ is\ a\ static\ service\ attribute\}$	Eq. 5

The service profile comprises the union of the two sets, i.e.,

$Ps = B_{dyn} \cup B_{stat}$	Eq. 6

During the recommendation process, the matchmaking algorithm determines similarities between the user profile set, Set P_u in Equation 6 and service profile set, Set P_s. It is worth noting that in

this paper$x$, y, q and r are considered to be of both numerical and textual value types. The numerical values are further categorised as of continuous numerical value types and discrete value types -for example distance if a continuous value type is valid, while a number of resources on a "Thing" are of a discrete nature.

For example if

$x = \{\{x_i \mid x_i discrete\}, \{x_j \mid x_j\ continuous\}\}$	Eq. 6

Then the equation becomes,

$P_u = \{\{\{x_{i,k}\}, \{x_{j,l}\}\}, \{y\}\}$	Eq. 7

Similarly, the service profile in Eq. 6 can be given by

$P_s = \{\{\{q_{i,k}\}, \{q_{j,l}\}\}, \{r\}\}$	Eq. 8

It should be noted that as part of the user profile there are preferences that are specified by the user during the enrolment or registration. The preferences are service attribute name-value pairs that a user specifies. They are therefore a subset of attributes that are prioritised during the matchmaking process. In the next section how this subset is taken into consideration during the matchmaking process will be discussed in detail.

The matchmaking algorithm that enables the recommendation process will be discussed in the next section.

4.3.2 Matchmaking

This section presents the matchmaking of the user and the service profile using selection criteria that emanate from recommender systems. Recommender systems are computer-based intelligent techniques to deal with challenges in finding appropriate services from large numbers of

available services. Therefore, recommender systems can be the function that takes P_u and P_s as inputs and produces a list of matching score. The matching score is what is used to rank services for returning to the user. In this section, we discuss how these profiles are matched for recommendation of the services based on user profiles and preferences.

In essence, matchmaking utilises a function that aggregates the matching scores of each attribute's values from both the service and the user profile. There is a unique function for each attribute, which determines similarity of the specific attribute values from both the service profile and the user profile. As an example, a function that determines the vicinity of a service to the user (distance) may not be used to compare resource types specified in the user preferences set against the resource type of the service being matched. A different function that compares the resource types is therefore required. The matching function therefore would be

$$F(P_u, P_s) = \sum_{i=0}^{n} w_i f_i(P_u, P_s) \qquad \text{Eq. 9}$$

Where$n$, is the total number of attributes to be matched, w_i is the assigned weight of attribute a_i and f_i is the function used to compare values of attribute a_i.

Step 1. The user preferences set is considered. This set of name-value pair attributes needs to be compared against the attribute name-value pairs of the published IoT services. Often, one of the attributes that will be specified during enrolment is the resource types set. The output from this step will be the set of attribute name-value pairs. With this set of attribute name-value pairs, we proceed to step 2.

Step 2. The second phase of the matchmaking algorithm involves matching the user profile and the service profile based on the attribute name-value pairs that were not specified in the user preferences. Retrieve user profile attribute name-value pairs not specified in preferences.

Step 3. Match each service profile against the user profile.

Step 3.1. In this step, the attributes in both profiles are compared and the ones that are common among the profiles are matched. For example, if the user has his current location specified in his profile and the service which may be mobile has the current location specified, the current location attribute is included in the set of attributes against which the service profiles and the user profile will be matched.

This involves selecting a set of functions $\{f_i\}$ with each function corresponding to an attribute to be matched.

An output from this step is a set of attribute names. With this output, proceed to Step 1.1.Otherwise, proceed to Step 3.2.

Step 3.2. Match non-common attributes between the user profile and the service profile based on semantics - e.g. attributes "permanent location" and "home location" are semantically similar.

Step 3.2.1. Get semantically similar attribute names from service profiles and the user profile.

Step 3.3. If the list of services to match is not empty proceed to Step 3.3.1. Otherwise, proceed to Step 3.3.1. else proceed to Step 4.

Step 3.3.1. For each attribute, the name-value pair from the user profile is matched against the attribute name-value pair of the service profile. The output from this step will be a set of services that are matched based on these attributes. This is where Eq. 10 is applied. A service is considered to have some matching degree if, and only if,

$$F(P_u, P_s) > 0$$

Step 3.3.2. If results are not returned and the attribute is of a dynamic and numerical nature and not at maximum value, the threshold is increased and the algorithm returns to Step 3.3.1.

Step 3.3.3. If results were found and are more than the maximum desired list length, and the attribute is of a dynamic and numerical nature and not at minimum value, the threshold is reduced and the algorithm returns to Step 3.3.1.

Step 3.3.4. If results are not found, all dynamic attributes are at minimum, and non-preferred attributes are not considered, the algorithm proceeds to Step 2.

Step 3.4. Rank services. The services are sorted based on their computed value of the function $F(P_u, P_s)$.

Step 3.5. Return recommended list

Step 4. Notify user and Exit

The next section re-compares the existing approaches and architectures against the proposed ISSA and ISSA algorithm

4.4 Comparison of ISSA vs architectures in the literature

This section presents the comparison of the ISSA with other existing architectures studied in the literature. The comparison is based on components that describe IoT natural environments such as domain, storage, algorithm, scalability, evaluation and user profile.

Domain: Domain specifies that the scope of data the sample IoT services used embraces the smart campus services offered by the experimental environment.

Storage: The services are stored in the internal middleware, and published by creating a proxy on DMZ through the API. The services are integrated and deployed on the cloud database.

Algorithm: The ISSA algorithm allows different types of user-preference recommender systems techniques to be applied using the same selection criteria proposed in Section 4.1.2.

Scalability: The ISSA is developed such that the system data are able to network or process the service selection and handle a growing amount of services in the ISSA's ability to be expanded to accommodate that growth.

Evaluation: The ISSA will be evaluated based on recall, precision and response time.

QoS: The non-functional properties are presented in the ISSA; only the *Matchmaker* component will be expanded. There is a consistent way for the user to select services because users perceive quality through the prism of their own experience and rate those services maps to the specific quality of parameters offered by a provider.

User Profile: The "set_pref" composite service is extracted to atomic services through modelling.

Table 5: ISSA comparison

Author	**User profile**	**Domain**	**Storage**	**Algorithm**	**Scalability**	**Evaluation**	**QoS**
(D'Mello et al. 2008)	The functional properties are not the focus of this work; however the main focus was on non-functional properties	Does not specify domain	Uses UDDI	Uses WSDL	Web services are scalable depending on the domain	√	√
(D'Mello and Ananthanarayana 2009)	Uses QoS requirements specified by a user	√	Uses UDDI	Uses WSDL	√	√	√
(Manikrao and Prabhakar, 2005)	Uses collaborative filtering	Does not specify domain	Uses UDDI	Uses WSDL	Web services are scalable depending on domain	√	√
(Lin and Tao 2006)	Uses QoS requirements	√	Uses service repository	√	√	√	√
(Guha 2009)	Focuses on volume of incoming requests	√	Uses service registry	√	√	√	√
ISSA	√	√	√	√	√	√	√

		composed in the repository	
User-centred QoS-based service-selection approach	It is an expressive and flexible way for non-expert users to define their own QoS requirements	Proposes an approach that is expressive and flexible for non-expert users to define their own QoS requirements	Does not give a user a chance to request a service in any acceptable format that can be further broken down to required user tasks and preferences.
Functional-based service-selection approach	An approach which answers questions on how service should be selected to form part of selection criteria	The functionalities lead to selection criteria that formulate the algorithm to be used during the service-selection process	Can be limited by the nature of the environment and the problem intended to be solved by service-selection
User-preference-based IoT service selection	**The approach is based on a user-preference approach that emanates from recommender systems**	**Selects real-time services, allows users to define their preferences, selects services based on automated attributes such as current location, current device status, and date-time**	**The shortcoming is that as the collaborative filtering is based on the role. If the services are defined using the array. The order is likely to shrink.**

4.6 Summary

This chapter has proposed the IoT service-selection architecture (ISSA) and the ISSA algorithm. The ISSA algorithm showed the need to re-define other attributes. Re-defining attributes was done through user and service modelling. The modelling made it easy to match services using re-defined user preference attributes. Chapter five designs and implements the ISSA.

5 ISSA experimental design and implementation

The implementation of the prototype of the ISSA that captures the pertinent components needed for IoT service selection is described in this chapter. Section 5.1 presents the ISSA experimental design, Section 5.2 the ISSA implementation and Section 5.3 the summary of the chapter.

5.1 ISSA experimental design

This section presents the implementation design of the ISSA proposed in Section 4.2. The design comprises use case diagrams and tables in Section 5.1.1, a sequence diagram in Section 5.1.2 and an activity diagram in Section 5.1.3, an entity relation diagram in Section5.1.4 and Section 5.1.5 presents the summary of the experimental design. The next section presents use case diagram.

5.1.1 ISSA use cases

An interesting system interacts with human or automated actors that use that system for a certain purpose, and those actors expect that system to behave in predictable ways. The use cases for the architecture to be implemented in this work are illustrated in Figure 18. A use case specifies the behaviour of a system or a part of a system and is a description of a set of sequences of actions, including variants that a system performs to yield an observable result of value to an actor. Use cases are applied to capture the intended behaviour of the system one is developing, without having to specify how that behaviour is implemented. Use cases provide a way for the developers to come to a common understanding with a system's end-users and domain experts. In addition, use cases serve to help validate architecture and to verify the system as it evolves during development. Each use case has its own specific task, presents the use case with two actors - the service requester and service provider.

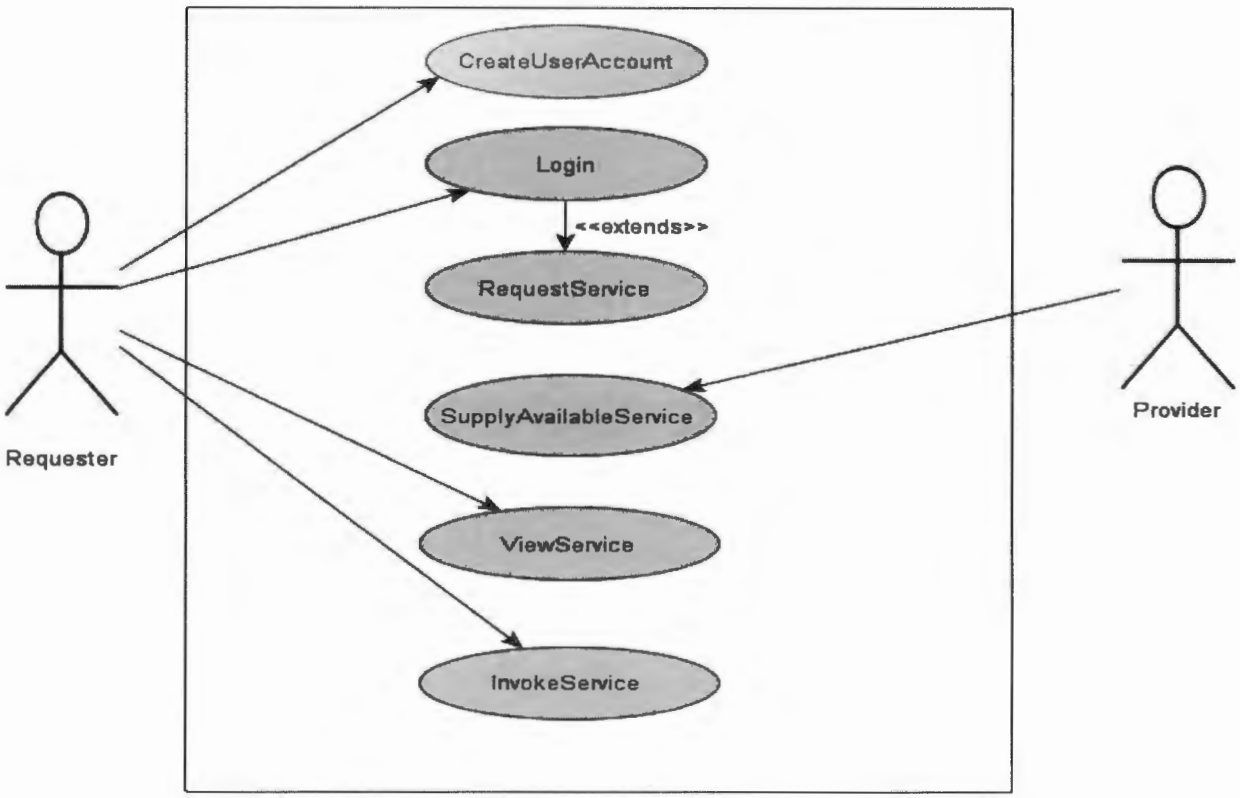


Figure 18: Use case diagram

As one implements the system, the use cases are realised by collaborations, whose elements work together to carry out each use case. Details about each use case are given in the tables.

Table 7: Create account

Description	This use case describes the procedure that the user has to follow in order to log into the system.
Primary actor	Service requester
Secondary actors	
Preconditions	The user is searching for a service and does not have appropriate rights to use the system.

Minimal and success guarantees	User successfully registered.
Trigger	User chooses to create an account.
Normal course	1. User chooses to create an account. 2. Account is successfully created. 3. User chooses to log in. 4. System displays the Home Page with search interface.
Notes:	When user chooses to login, the login process is referred to the login description.

Table 8: Use case: Login

Description	This use case describes the procedure that the user has to follow in order to log into the system.
Primary actor	A registered user.
Secondary actors	
Preconditions	User is registered and has appropriate rights to use the system.
Minimal and success guarantees	User successfully logged in.
Trigger	User chooses to log in.
Normal course	1. User chooses to log in. 2. System requests a username and password.

	3. User types in the username and password and submits. 4. System authenticates the user based on the username and password. 5. System displays the Home Page with the limited access menu
Notes	If the user has registered and is visiting the site for the first time, at this stage the user has a right to request any service immediately.

Table 9: Request service

Description	This use case describes the procedure that the user has to follow in order to request the service.
Primary actor	A registered user.
Secondary actors	
Preconditions	User is registered and has appropriate rights to use the system.
Minimal and success guarantees	User successfully submitted service search.
Trigger	User chooses to request a service.
Normal course	1. User chooses to request a service. 2. System gives proper interface. 3. System searches for the service. 4. System requests available services in order to make selection possible.
Notes	This process is performed by the system automatically because the system stored the user preferences when the user created an account.

Table 10: Supply available service

Description	This use case describes the procedure that the system has to follow in order to supply available services.
Primary actor	A registered user.
Secondary actors	
Preconditions	User submitted a service request.
Minimal and success guarantees	User successfully submitted service request.
Trigger	Provider supplies available service.
Normal course	1. System updates services. 2. System gets all available services. 3. System supplies available services.
Notes	This process recommends services based on user preferences.

Table 11: View service

Description	This use case describes the procedure that the user has to follow in order to view the service.
Primary actor	A registered user.
Secondary actors	
Preconditions	System supplies available services, matched services and compiles recommended list.
Minimal and success guarantees	System has successfully matched the services.

Trigger	User views services.
Normal course	1. User views recommended services. 2. User chooses the service of interest and invokes the service.
Notes	The user usage is stored on the system; when the user visits the application again, the service the user invoked before becomes the priority to be recommended to the user.

Table 12: Use case: invoke service

Description	This use case describes the procedure that the user has to follow in order to invoke service.
Primary actor	A registered user.
Secondary actors	
Preconditions	System has supplied available services, matched services and compiled recommended list.
Minimal and success guarantees	System has successfully matched the services.
Trigger	User invokes services of choice.
Normal course	1. User chooses a service among recommended services. 2. User invokes a service of interest.
Notes	The system is then expected to rate the selected service as the priority for the next recommendation.

5.1.2 ISSA sequence diagram

The sequence diagram shows the sequential interaction between the ISSA components. The sequence diagram in Figure 19 consists of three layers: application layer, middleware layer and registry. Application layer has the sub-component *user*; the user can be a human being or an application. The middleware layer consists of three sub-components: DB Connect, Matchmaker and the recommender system (Rec System). The last section is the registry, which keeps the database of the users, services and preferences. The user first registers with the system by creating an account. DB connect publishes the user's records to the registry. The system then automatically sends a request for a service. The matchmaker requests all available services on the registry based on user search attributes, and available services will be listed. The recommender system gets the user profile, service profile and user preference to compare and predict recommendation.

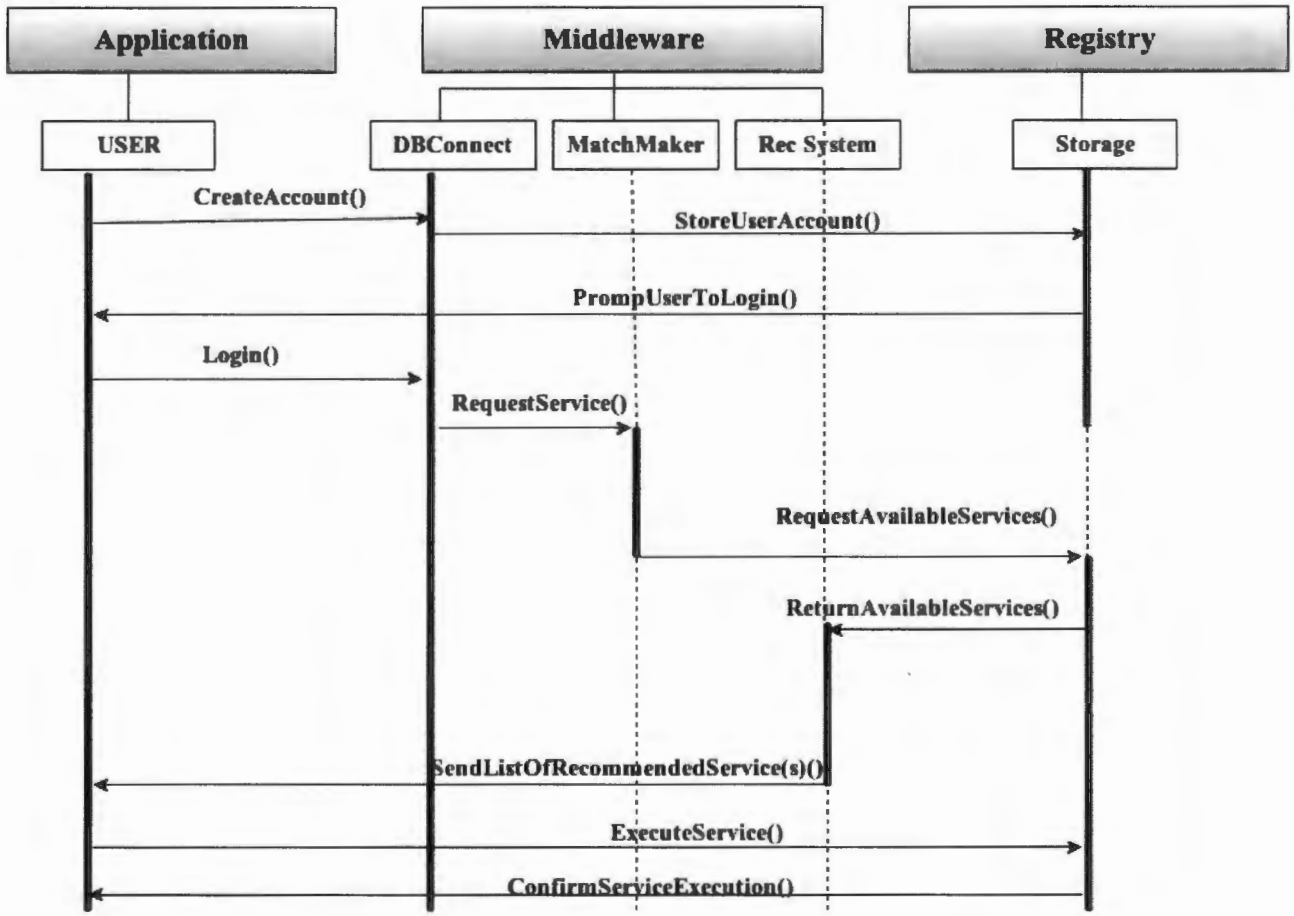


Figure 19: Sequence diagram

The recommender system sends the list of the most relevant service(s) and allows the user to choose one service from the list. The user will then invoke the service and do what the user wanted to achieve. Based on the scenario, the user will then check if the blinds are open or closed. The system will then send the confirmation of service invocation and allow the user to quit or log out.

5.1.3 ISSA activity diagram

The sequence diagram in Figure 19 showed the flow of messages among the components and actors in the ISSA. Figure 20 shows the activities among the entities. The user initiates the flow of messages by requesting the service. If the service request is automated, the user is expected to only log in. The user account creates a user profile with attributes that can match the user profile. The requester initiates the selection process, in which the user is expected to have registered with the system. If the user is not recognised then the user should create an account. If the user has created an account and specified preferences, the system will then send the service request. Matchmaker requests the latest updates on available services. The registry supplies all available services for selection. The recommender system sub-component needs user profile, service profile and preferences to predict the recommendation. If the service(s) are found the services would be recommended to the user or the system would compose services again. If the matchmaking is done for the second time and there is no match between the requested service and the available service, the selection will be aborted.

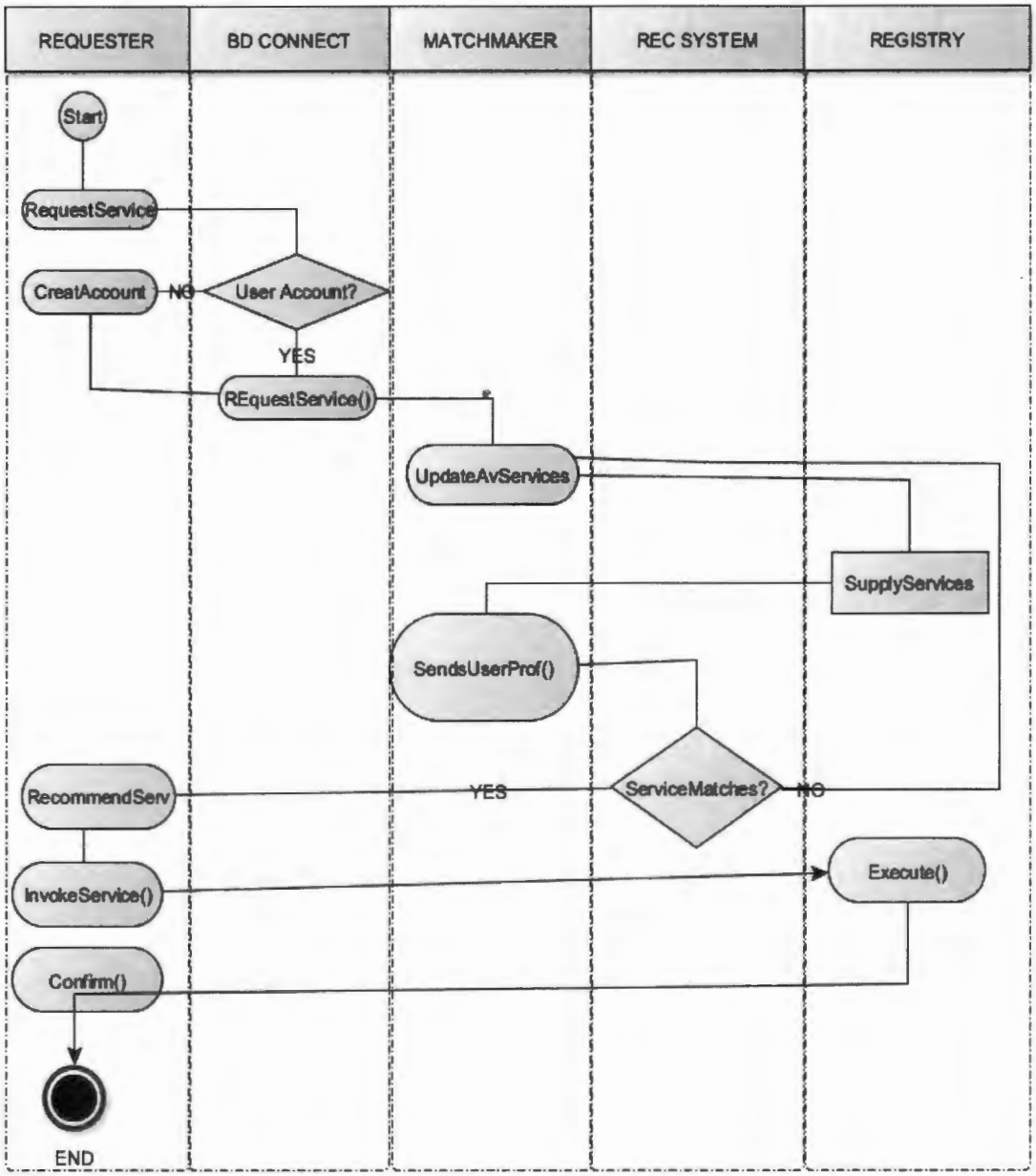


Figure 20: Activity diagram

5.1.4 ISSA entity relation diagram

The entity relation diagram (ERD) presents the relationship between entities. The entities, according to this work, are user and service. The relationship is presented as follows:

User requests a *service*

Figure 21 shows the relationship using the foreign key attributes.

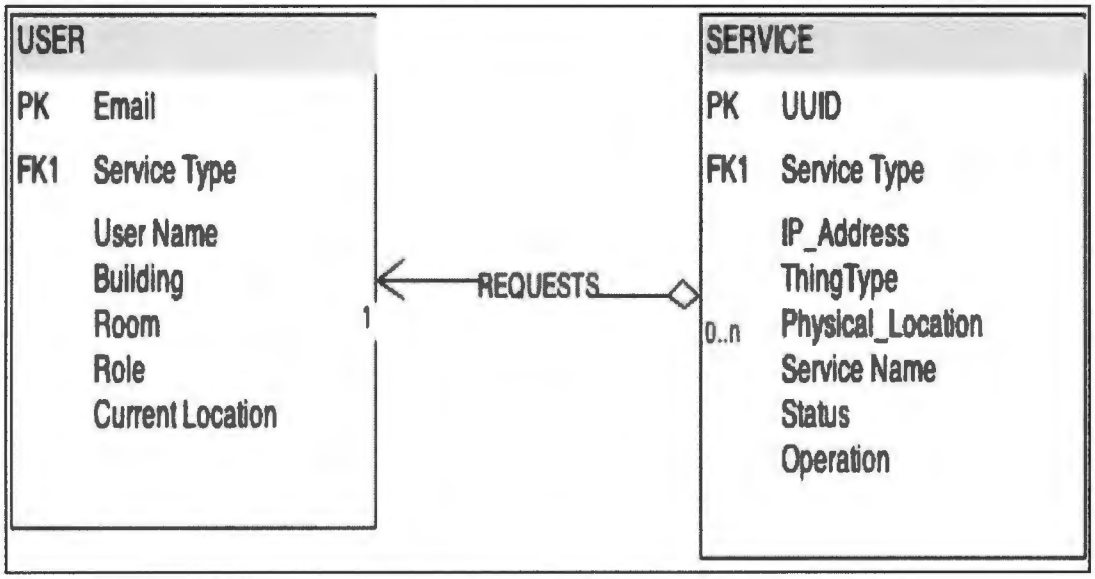


Figure 21: Entity relation diagram

The relationship between two entities plays a role during matchmaking and when compiling a profile and creating a database.

5.1.5 Summary

The ISSA experimental design shows the technical interaction between the ISSA components and entities. Each component has its own specific functionality to make the selection process successful. Use cases, the sequence diagram, the activity diagram and the ERD illustrated the architectural implementation design.

The next chapter focuses on the implementation of the ISSA.

5.2 ISSA implementation

This section presents the implementation of the ISSA. The ISSA was implemented on the Ubuntu 12.04 environment with 6 GB of RAM and an Intel(R) Core (TM) i7 CPU M620 @ 2.67 GHZ processor. The tool used in the development process was an Android Studio version 0.8.2. The Android device utilises SQLite where the application is installed for database administration. The application was released and installed on a mobile phone. The following sections present the screenshots of the application on an Android S4 phone.

5.2.1 IoT service application

The application was released and installed on the S4 Android phone and it appears titled "IoT Service" as shown in Figure 22. The use of this application requires a connection to the Internet. A click on the application takes a user to the "Login" page in Figure 23.

Figure 22: IoT Service Application

5.2.2 Welcome page

Figure 23 serves as a welcome page. Every user must create an account before accessing the services on the application. If the user is not registered with the system, the user gets options to register or quit the application.The second image in Figure 23 allows the user to register with the system.A "click" on "Register", proceedsto the "create account" page in Figure 24.

Figure 23: Login page

5.2.3 Create account

This page allows the user to create an account. The information about the user is stored on SQLite, the Android device database. On the "Prefered Things" attribute in Figure 24, the selection of "Prefered Things" becomes possible when the device is connected to the Internet. If the device is not connected to the Internet, there will be no preferred "Things" stored in the database.

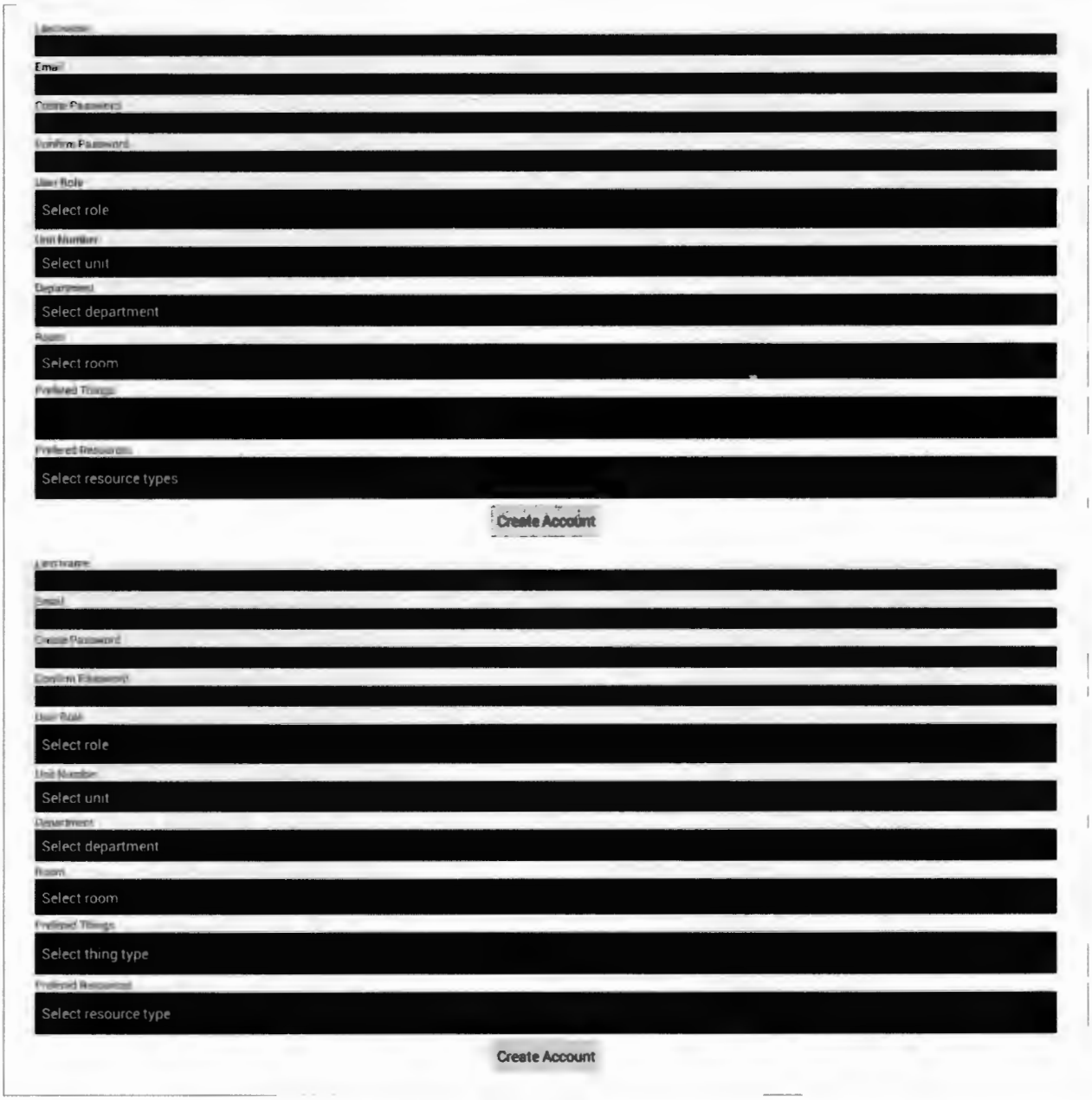


Figure 24: Create account

If the device is connected, the "Things" in the middleware will be shown as presented in Figure 25. The first image shows the list of preferred things while the second image shows the list of preferred resources. The user is expected to select the list of Things the user prefers to access as shown with a tick [√] in Figure 25. Success with the account creation proceeds to "Login" page

in Figure 27. Success with the user account forms a user profile that is further used to match the service profile for recommendation.

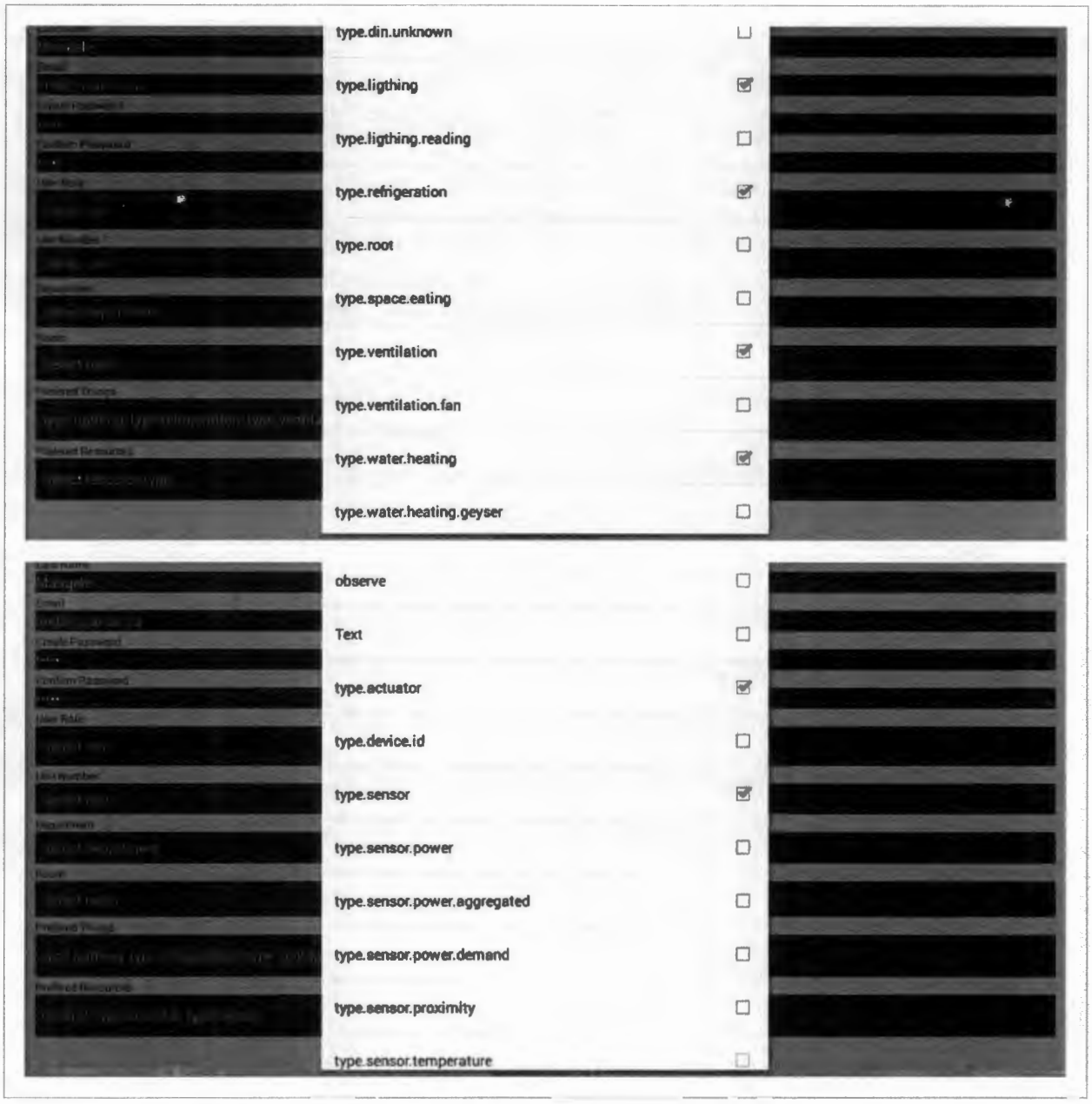


Figure 25: Preferred things

5.2.4 Login

This page serves to aunthenticate every user that accesses the application. If the credentials are incorrect, the system asks the user to try again later or fill in an email address in order to be reminded as shown in Figure 26.

Figure 26: Validation

The second image of Figure 26 allows the user to fill in an email address in order to receive a reminder of the credentials.If the credentials are correct, the "Signin" click in Figure 27proceeds to the list of recommended services in Figure 28.

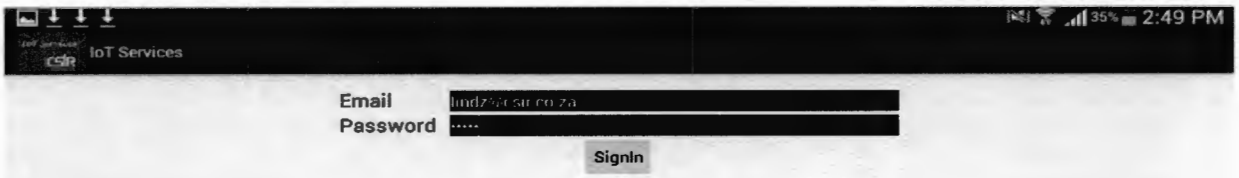


Figure 27: Login

5.2.5 List of recommended services

The "Login" page is linked to the matchmaking class that is transparent to the user. The recommendation of this application becomes automated after receiving the user profile. The selection process is automated because after the user has specified preferences, no more input is expected from the user. The matchmaking takes place within the system. The user can only see the list of recommended services as the result of matchmaking that took place in the within the system. The application gives the user an option to utilise the service or "logout" from the application, as indicated in Figure 28. If the user decides to utilise the service, the application takes the user to the page in Figure 29 or, if the user decides to log out, the application stops.

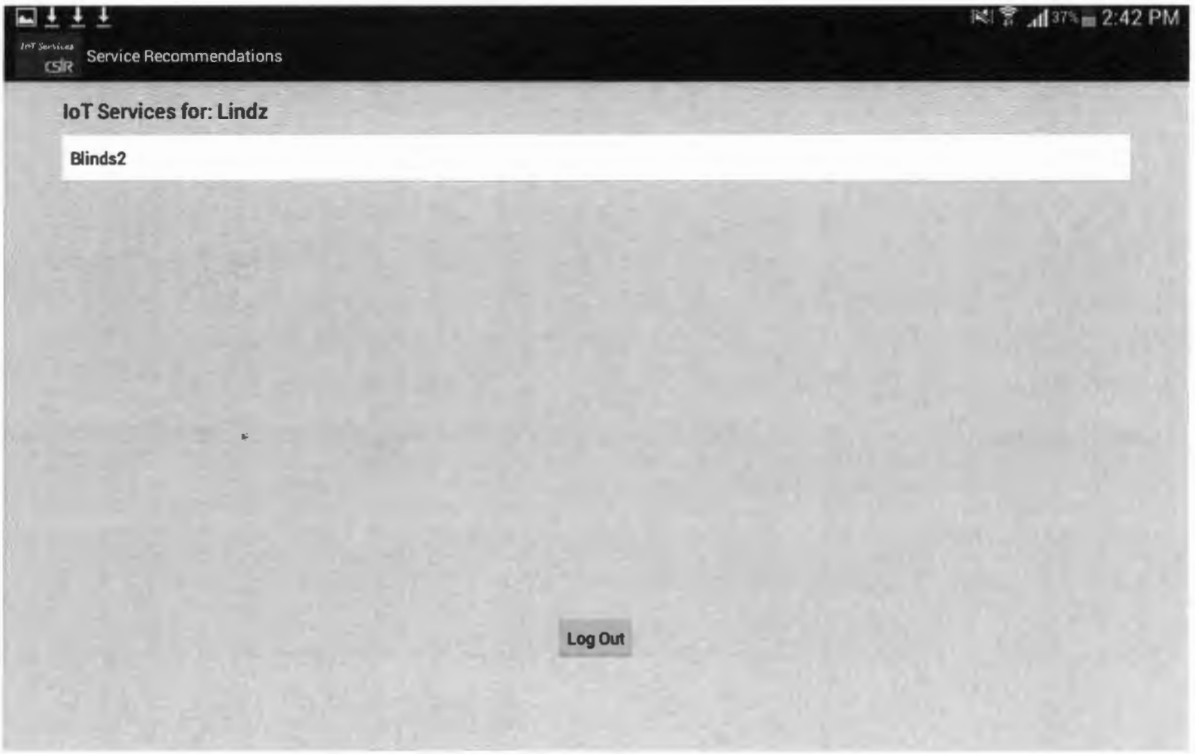


Figure 28: List of recommended services

5.2.6 Service invocation

This page appears when the user decides to utilise the service. The red arrow increases/turns on/opens, depending on the Thing type. The blue arrow decreases/turns off/closes the Thing type. The green button allows the user to submit changes on the Thing type. Otherwise, the user can quit the application or log out from the application.

Figure 29: Service Invocation

5.3 Summary

The chapter showed that the application works in line with the proposed architecture. Furthermore, the algorithm needs to be evaluated. Section 1.5 mentioned that the performance of the recommender system would be evaluated using precision, recall and response-time metrics. The next chapter presents the evaluation of the ISSA and the results analyses.

6 ISSA evaluation and results analyses

This chapter presents the evaluation of the ISSA. The evaluation is based on performance and the effectiveness of the algorithms. Parameters to be considered include response time, recall and precision to be used in this work. Section 6.1 presents the evaluation of the ISSA, Section 6.2analyses the results and Section 6.3 summarises the chapter.

6.1 ISSA evaluation of the algorithm

The evaluation of the algorithm implemented in this work is twofold. First, in Section6.1.1, the effectiveness of the user preference-based algorithm derived from the architecture is evaluated using some simulated users. Section 6.1.2 evaluates the performance of the same algorithm proposed in this work.

6.1.1 Effectiveness of the algorithm

In order to evaluate the accuracy of the architecture, this work has chosen the remaining candidates of recommender algorithms from the evaluation conducted in Section 3.4.1.1. The recommender systems were first classified based on their approach when recommending services or products. Manqele *et al.* (2012) conducted the evaluation of recommender algorithms that work in smart environments. The remaining candidates were content-based (CB) and collaborative filtering (CF) recommender systems. However, the evaluation will be based on two algorithms. The aim is to check the algorithm that works best for such an environment among user preference-based algorithms.

The evaluation of effectiveness will be conducted using recall and precision metrics. Precision metrics measure how good the framework is in reducing irrelevant recommendations (Davis and Goadrich 2006).A good preference model is expected to optimise the recall and precision parameters. The precision metrics are presented as follows:

$$\textbf{Precision} = \frac{number\ of\ correctly\ detected\ services\ for\ preference\ of\ user\ i}{number\ of\ ALL\ detected\ services\ for\ preference\ of\ user\ i}$$

Recall measures how well the architecture has been developed and makes sure that there is no missing relevant recommendations (Davis and Goadrich 2006). The recall metric is presented as follows:

$$\textbf{Recall} = \frac{number\ of\ recemmended\ services\ for\ preference\ of\ user\ i}{number\ of\ ALL\ services\ \ for\ preference\ of\ user\ i}$$

Table 13 presents the evaluation data. The table consists of values for the content-based recall and precision values for the collaborative recall and precision.

6.1.1.1 Test design

The experimental framework in Figure 30 was designed to present the evaluation procedure. The framework consists of five users per preference, meaning the first user had one preference (user1/Preference 1), the second user had two preferences (user2/Preference2), up to the fifth user. The user queries the service stored in the middleware. In the IoT middleware, there are other components, but the component used in this work is the registry that stores services, although the matching of services was happening in the application but the services were residing in the middleware. The services were invoked from the middleware and matched on the application layer using user profile, service profile, physical location and rules. The application layer consists of components that interact with the user. The result displays the recommended services, the value of recall, precision and response time. The query was repeated five times with the same number of preferences and average response times were compiled.

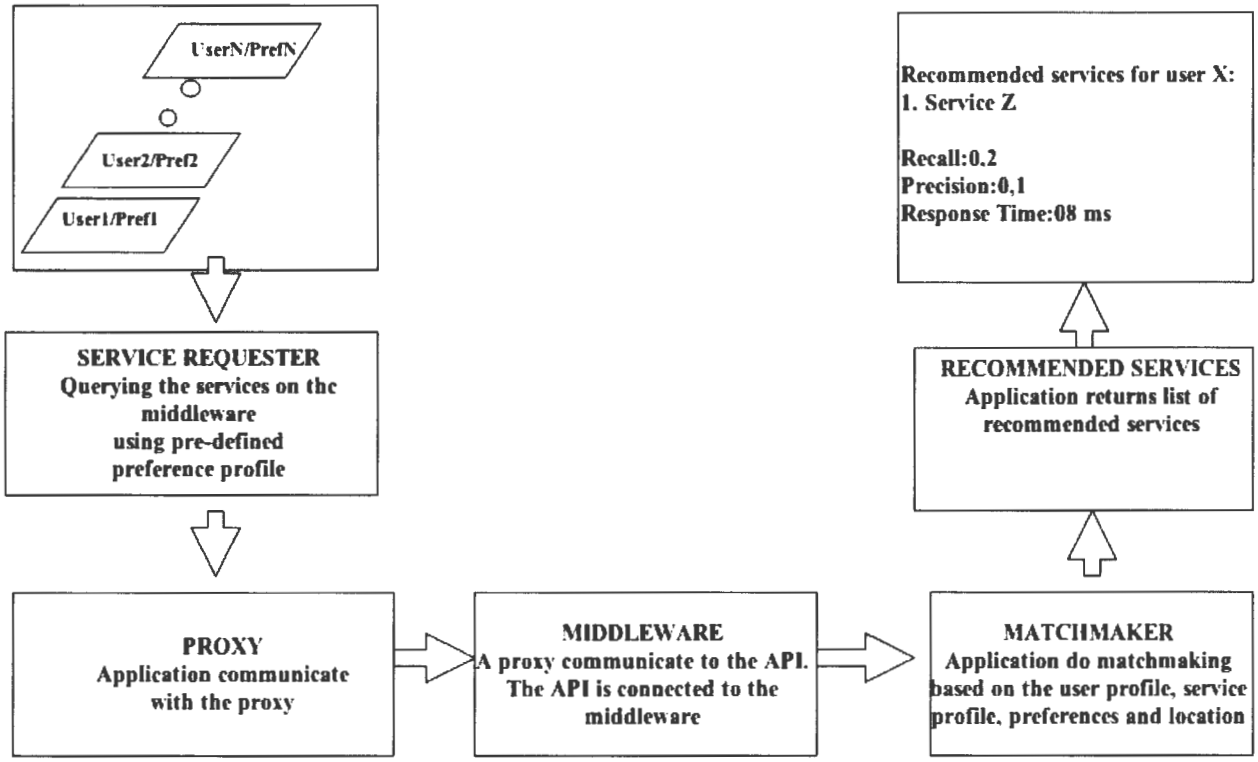


Figure 30: Experimental framework

Table 13: Recall vs Precision data

	RecallCB	PrecisionCB	RecallCF	PrecisionCF
1	0,02	1	0,02	0,01
2	0,02	0,5	0,06	0,38
3	0,04	0,67	0,08	0,44
4	0,06	0,75	0,08	0,5
5	0,08	0,8	0,1	0,63

Based on the values in Table 13, the graph was plotted in Figure 31. The graph presents the results for recall vs precision for the content-based and for the collaborative filtering as indicated in Figure 31.

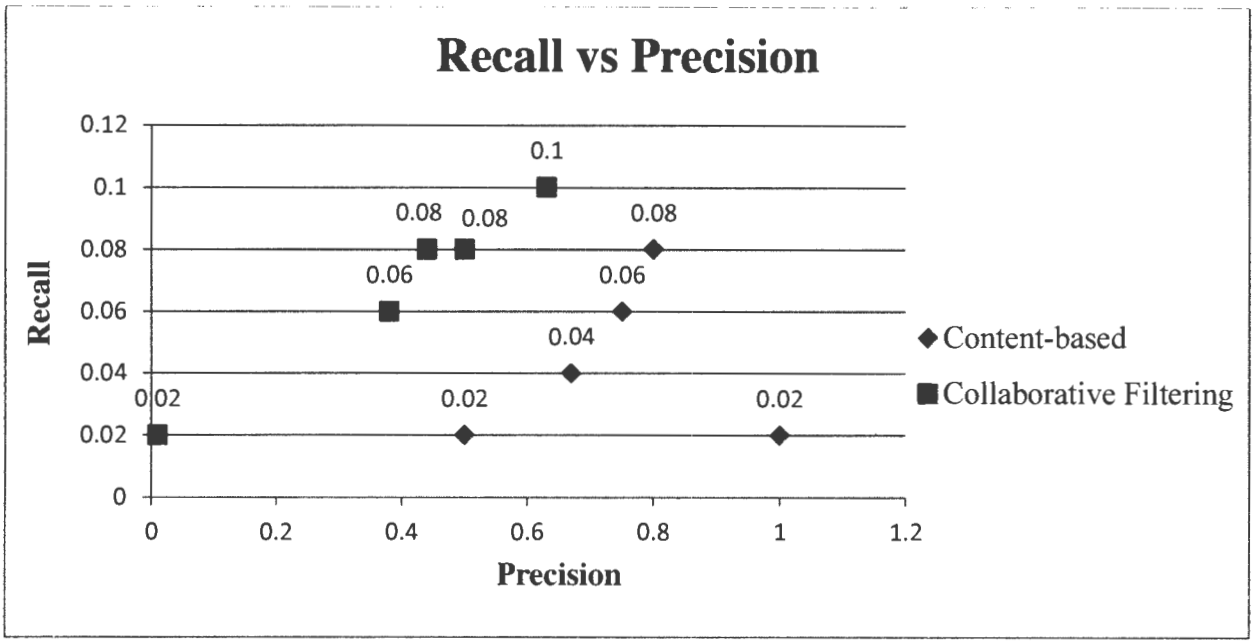


Figure 31: Recall vs precision for CB and CF

6.1.2 Performance of the algorithms

The performance evaluates the quality of the user preference-based algorithm implemented in this work. This work uses response time as the metric to measure the performance of the algorithm.

Response time = *Request time + query time + recommendation time*

Request time is the time taken by the system to send requests.

Query time is time taken by the system to do matchmaking.

Recommendation time is the time taken by the system to return a list of recommendations.

The results obtained using the response time are presented in Table 14. The response time was measured in microseconds. The time presented in the table is the average of the request repeated five times.

Table 14: Response time data

	Content-based	**Collaborative filtering**

1	2	21
2	3	15
3	4	32
4	7	11
5	10	20

Based on the values in Table 14, the graph was plotted in Figure 32. The graph in Figure 32 presents the response time for content-based and collaborative filtering recommender algorithms.

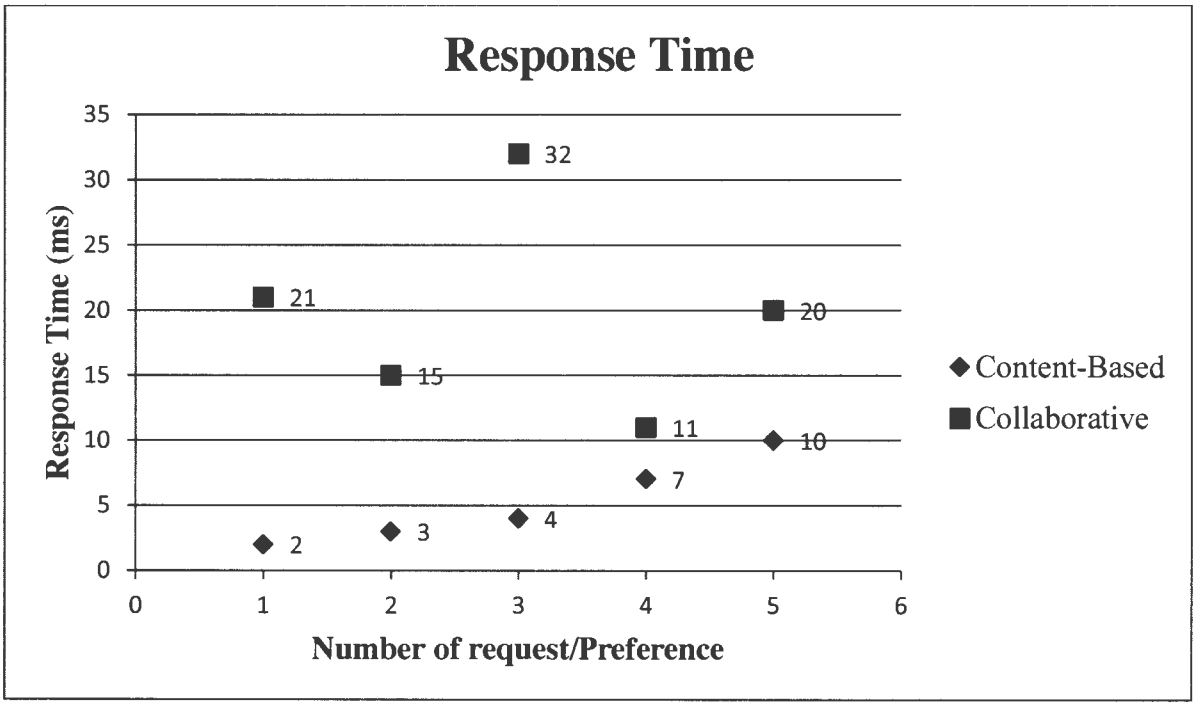


Figure 32 : Response-time evaluation

6.2 ISSA results analysis

The comparison of the two techniques -content-based and collaborative filtering recommender systems -was carried out through experiments. These recommender systems are based on both memory and model. The memory-based approach can either be based on user or service memory, depending on which technique is used. The model-based approach is based on the components of the architecture or model developed. The intended goal of the application implemented is to receive the list of recommended services per query.

6.2.1 Discussion

Recall presents the effect of the services detected from the overall number of services in the database. *Precision* presents the effect of the services needed from the overall number of services detected. Table 13: Recall vs Precision data shows that collaborative filtering can detect up to 10% of the available services while content-based detects up to 8%. *Precision* presents the effect of relevance in the algorithm. Table 13 shows that content-based filtering returned between 50% and 100% relevant services, while collaborative filtering returned between 10% and 63%, at most, of relevant services.

The response time presents the quality of the system's performance. This can reflect the challenges associated with the device or the challenges associated with the effectiveness of the algorithm. However, collaborative filtering took almost double the time of the content-based recommender algorithm. Based on the interpretation of the graphs in Figure 31and Figure 32, the next section presents the advantages of using the content-based and the collaborative filtering recommender systems.

6.2.2 Advantages

This section presents the advantages of using content-based and collaborative filtering recommender systems. The advantages discussed in this section are based on the experience of attempting the ISSA approaches in this work.

6.2.2.1 Advantages of the content-based recommender system

This technique focused on the ISSA selection criteria for learning user preferences and filtering a stream of new services for those that most closely match user preferences. This approach built on the fundamental assumption that users are not able to formulate queries that express their interests. This approach could be applied to services by indexing the text description of service based on the words that occur in them. Content-based involved formal structure description of

services that was modelled and supported the dynamic composition of the services. Hence, the IoT service content-based took less time to return recommended service compare using collaborative filtering The most relevant services were returned using content-based technique. The application gives an option to clear the history of usage on the application so that the App does not rely on the experience but considers the current context and preferences.

6.2.2.2 Advantages of Collaborative filtering recommender system

Collaborative filtering recommender system recommends services that people with similar tastes or role and preference utilised in the past. The IoT service filtering is context-aware recommendation system based on the premise that some other web services are only applicable in given context. The context under which a service is applicable was captured in the IoT service description and modelling. Using collaborative filtering saves time of defining and re-defining the services. The role of the employees determines the access restriction for each user. The use of the collaborative filtering bridged the cold start in collaborative filtering. Cold start happens when the system does not consider the newly added service. This work selects services dynamic, newly added services are always considered on the algorithm.

The next section presents the disadvantages of using the content-based and the collaborative filtering recommender systems.

6.2.3 Disadvantages

This section presents the disadvantages of using content-based and collaborative filtering recommender systems. The disadvantages discussed in this section are based on the experience of attempting the ISSA approaches in this work.

6.2.3.1 Disadvantages of the content-based recommender system

Users can indicate preferences among the services they are restricted to utilise. Other services should be recommended, based on the role of the user.

6.2.3.2 Disadvantages of the collaborative filtering recommender system

It predicts the user's interest in services based on other people's opinions (roles) instead of performing selection based on the defined preferences. Defining restrictions based on indexes can be challenging when the new service is added.

The next section presents the challenges of the content-based and collaborative filtering recommender systems.

6.2.4 Challenges

The challenge of using collaborative filtering was defining the role using indexes. The indexes are likely to shift when the new service is added on the server. This challenge includes security as one of the IoT challenges, as indicated in Figure 1. It would be difficult to mention the number of services that were on the middleware because the number of services kept increasing.

6.3 Summary

This chapter evaluated the ISSA architecture and algorithm based on recall, precision and response time metrics. The ISSA algorithm was implemented using both content-based and collaborative filtering techniques. Based on these techniques, the evaluation to check the effectiveness and performance of the algorithm was presented. The results analyses using the graphs were presented. The results showed that content-based was more efficient and effective than the collaborative filtering algorithm. The next chapter presents the conclusion and the future recommendation.

7 Conclusion

Working in the environment where "things", both physical and virtual, have identities, attributes and intelligent interfaces that enable them to be "smart", able to sense, analyse, connect, and communicate, encourages us about intelligent and integrated systems that can "think" ahead of their developers and create opportunities in addressing other research issues. IoT is an emerging paradigm that brings together people, machines, inanimate or animated objects, facilitating their interactions over the Internet communication infrastructure, using established Internet standards and technologies to make human beings behave intelligently by providing applications that eliminate human labour using mobile devices.

The goal of this study was to develop a preference-based architecture that will enable the selection of smart campus IoT services in the cloud through recommender system techniques using mobile devices. The emerging paradigms, such as IoT, smart campus, cloud computing, SOA and recommender systems, were surveyed and integrated to achieve the goal of this work. The survey led to the synthesis of the definition for each paradigm as IoT was the main paradigm of this research. The sampling services were services provided by the research environment associated with the smart campus. These services were stored and published on a private cloud. The SOA, which consists of service consumer, service broker and service provider components, was adapted to this work. The contribution of this research was on the broker component. The broker component implemented the recommender systems. This proposed the ISSA, implemented it and evaluated it based on recall, precision and response-time metrics. These metrics were comparing the effectiveness and the performance of each recommender system technique's content-based and collaborative filtering. The ISSA proposal is discussed in the next paragraph.

The challenge in IoT was to find the key components of an architecture that could be used for integrated IoT service selection, using user preferences in a smart-campus environment to support dynamic service composition. Identifying, integrating and selecting an IoT service is a heterogeneity challenge, as indicated in Figure 1. The solution towards addressing the challenges of service selection was based on this scenario. The scenario was generic in order to cover any aspect in the scenario and be meaningful from the user's point of view. Based on this scenario and literature, IoT system considerations were identified. The literature revealed that not much work has been done in modelling user profiles when selecting services based on user and service

profiles. To this end, the IoT system considerations were used to compare user-preference algorithms from a theoretical point of view. The comparative analyses left us with two user preference techniques that were further tested on the application. The IoT system considerations were also used to develop the ISSA. The functioning of the ISSA was tested by selecting the services. Having established the key components necessary for an architecture to support dynamic selection of IoT services was not enough; there was a need to find the preference-based algorithm selection criteria, the ISSA algorithm. The ISSA algorithm was implemented and further evaluated based on the recall, precision and response time.

The evaluation showed that the content-based technique is more effective than collaborative filtering. Collaborative filtering algorithms recommended the items that matched based on the role of the user. Collaborative filtering recommended far fewer services than the services that were requested. Content-based algorithms recommended relevant services that were requested by the user. The performance showed that collaborative filtering was taking longer to return the list of recommended services, while content-based took less time. Therefore, the results showed that content-based works better than collaborative filtering in such IoT environments.

7.1 Future recommendation

The researcher needs to consider the nature of the environment before proposing the architecture or algorithm for service selection. This work considered the nature of the environment by defining the user profile, domain, storage, algorithm, scalability, evaluation and QoS. We recommend that these categories should be considered, especially when the selection of services is specific for an organisation. It becomes easier for the other researchers to relate their approach if the environment is well defined. The architecture needs to consider semantic matching to improve on the user profile. The semantic matching not only checks the syntax of the attributes, but also other similarities in meaning. The roles of users were presented in indexes. It would be better if the access restrictions could be done on the database itself, not on the matchmaking code. One can make recommendations by learning the behaviour of the user and make recommendations based on the user preference(s). In future, other researchers may consider doing the selection of services in smart cities and integrate services in the community or public or hybrid cloud. The change of domain can influence the increase in number of services on the database.

Reference

Ahsan, S.M., 2006. A framework for QoS computation in web service and technology selection. *Computer Standards & Interfaces*, 28(6), pp.714–720. Available at: http://www.sciencedirect.com/science/article/pii/S0920548905000966 [Accessed June 11, 2014].

Aldinucci, M., Daneletto, M. & Kilpatrick, P., 2008. A framework for prototyping and reasoning about distributed Sytems. In *Parallell computing and architectures algorithm and application*. pp. 219–226.

ASSOCIATION, S.N.I., 2010. *Implementing, Serving, and Using Cloud Storage*,

Atzori, L., Iera, A. & Morabito, G., 2010. The Internet of Things: A survey. *Computer Networks*, 54(15), pp.2787–2805. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1389128610001568 [Accessed April 28, 2014].

Averbakh, A., Krause, D. & Skoutas, D., Recommend me a Service : Personalized Semantic Web Service Matchmaking. , pp.2–4.

Badkas, K.C., Shahade, P.M.R. & Ramteke, U.S., 2010. Review of Internet of Things (IoT): A vision and future directions. , 1 Special(1), pp.129–134.

Balke, W.-T. & Wagner, M., 2003. Towards Personalized Selection of Web Services. In *WWW (Alternate Paper Tracks)*. Available at: http://dblp.uni-trier.de/db/conf/www/www2003at.html#BalkeW03.

Bardsiri, A.K., Hashemi, S.M. & Branch, B., 2012. A Review of Workflow Scheduling in Cloud Computing Environment. , 1(3), pp.348–351.

Barnaghi, P. et al., Semantics for the Internet of Things : early progress and back to the future.

Bianco, P. et al., 2011. Architecting Service-Oriented Systems. , (August).

Bieberstein, N. et al., 2006. No Title. *Service-Oriented Architecture (SOA) Compass-Business Value. Planning, and Enterprise Roadmap, IBM developerWorks*.

Black, P. et al., 2009. *Digital Britain-Final report*.

Bobby, S., 2005. A NEW SERVICE-ORIENTED ARCHITECTURE (SOA).

Buriano, L., 2005. Exploiting Social Context Information in Context-Aware Mobile Tourism Guides 2 FOAF for Social Context Representation. In *W#C Recommendation*.

Buyya, R. & Sulistio, A., 2008. Service and Utility Oriented Distributed Computing Systems : Challenges and Opportunities for Modeling and Simulation Communities Utility-Oriented Computing Systems. *Symposium A Quarterly Journal In Modern Foreign Literatures*, pp.68–81.

Chen Zhou, Liang-Tien Chia & Bu-Sung Lee, 2004. DAML-QoS ontology for Web services. In *Proceedings. IEEE International Conference on Web Services, 2004.* IEEE, pp. 472–479. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1314772 [Accessed June 11, 2014].

Computer, N. & Division, S., 2011. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology.

Corporation, D., 2010. Introduction to Cloud Computing Introduction to Cloud Computing.

Cranor, C.D., Gopalakrishnan, R. & Onufryk, P.Z., 2000. Architectural considerations for CPU and network interface integration. *IEEE Micro*, 20(1), pp.18–26.

Culler, D., Deborab, E. & Mani, S., 2004. Overview of Sensor Networks. , (August), pp.41–49.

D'Mello, D.A. & Ananthanarayana, V., 2009. Semantic Web Service Selection Based on Service Provider ' s Business Offerings. , 10(2), pp.25–37.

D'Mello, D.A., Ananthanarayana, V. & Thilagam, S., 2008. A QoS Broker Based Architecture for Dynamic Web Service Selection. *2008 Second Asia International Conference on Modelling & Simulation (AMS)*, pp.101–106. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4530459 [Accessed April 16, 2014].

Damiano, G., Giallonardo, E. & Zimeo, E., 2009. onQoS-QL: A Query Language for QoS-Based Service Selection and Ranking. In E. Di Nitto & M. Ripeanu, eds. *Service-Oriented Computing - ICSOC 2007 Workshops SE - 12*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 115–127. Available at: http://dx.doi.org/10.1007/978-3-540-93851-4_12.

Davis, J. & Goadrich, M., 2006. The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning - ICML '06*. New York, New York, USA: ACM Press, pp. 233–240. Available at: http://portal.acm.org/citation.cfm?doid=1143844.1143874.

Dey, A.K., Abowd, G.D. & Salber, D., 2000. A Context-Based Infrastructure for Smart Environments. In pp. 1–15.

Dlodlo, N. et al., 2012. The State of Affairs in Internet of Things Research. , 15(3), pp.244–258.

Drachster, HG, H. & R, K., 2007. Recommendations for learners are different: Applying memory-based recommender system techniques to lifelong learning. In Nertherlands. Available at: http://infolab-dev.ava.grl/sirte/2007/papers/Drachster_et_al.pdf.

Dutta, S. & Bilbao-Osorio, B., 2012. *The Global Information Technology Report 2012 Living in a Hyperconnected World.*

Dutton, W.H., 2005. The Internet of Things. *ITU INTERNET REPORT 2005.*

E, J. et al., 2006. Conceptual model for supporting advanced personalisation in M-Services. In *In proceedings of The 10th IASTED International Conference on SOFTWARE ENGINEERING AND APPLICATIONS SEA Dallas*. Texas, pp. 567–573.

Evans, D., 2011. The Internet of Things How the Next Evolution of the Internet The Internet of Things How the Next Evolution of the Internet Is Changing Everything. , (April).

FEDERAL IFORMATION PRPCESSING STANDARDS, 201, 2006. *FIPS PUB 201-1 Personal Identity Verification (PIV) of Federal Employees and Contractors*,

Fensel, D. et al., 2002. The Web Service Modeling Framework WSMF. *Electronic Commerce Research and Applications*, 1(2), pp.113–137. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.194.6486&rep=rep1&type=pdf\n http://ac.els-cdn.com/S1567422302000157/1-s2.0-S1567422302000157-main.pdf?_tid=b94db106-b038-11e3-a2ea-00000aab0f6b&acdnat=1395324522_066bdba7d81c246cd52bec55646af337.

Gubbi, J. et al., 2013. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7), pp.1645–1660. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0167739X13000241.

Guha, 2009. Investigation of Service Selection Algorithms for Grid Services A Thesis Submitted to the College of Graduate Studies and Research in Partial Fulfillment of the Requirements for the degree of Master of Science in the Division of Computer Science Universit. , (October).

Haller, S., Karnouskos, S. & Schroth, C., The Internet of Things in an Enterprise Context. , 2009, pp.1–15.

Hauck, M. et al., 2010. Challenges and Opportunities of Cloud Computing.

Huth, A. & Cebula, J., 2011. The Basics of Cloud Computing. , pp.1–4.

JB, S., JA, K. & J, R., 2001. *E-commerce Recommender Applications*, Kluwer Academic publisher.

Joachim, N., 2012. ARCHITECTURE (SOA):

Jørstad, I. & Thanh, D. van, 2006. Service Personalisation in Mobile Heterogeneous Environments.

Kevin, A., 2010. That ' Internet of Things ' Thing. , p.4986.

Khan, M.U., 2010. *Unanticipated Dynamic Adaptation of Mobile Applications.*

Kimpim, H., 2014. *Enterprise Architecture in Practice : From IT Concept towards Enterprise Architecture Leadership.*

Kleinrock, L., 2013. Cloud computing at a glance. In pp. 1–22.

Kontogiannis, K., Lewis, G. a. & Smith, D.B., 2008. A research agenda for service-oriented architecture. *Proceedings of the 2nd international workshop on Systems development in SOA environments - SDSOA '08*, pp.1–6. Available at: http://portal.acm.org/citation.cfm?doid=1370916.1370917.

Kulkarni, G., Sutar, R. & Gambhir, J., 2012. " CLOUD COMPUTING-INFRASTRUCTURE AS SERVICE- AMAZON EC2 ." , 2(1), pp.117–125.

Kuropka, D., Laures, G. & Troger, P., 2008. Semantic Service Provisioning. In K. Dominik et al., eds. Springer, pp. 5–18.

Kyriakos, K. & Dimitris, P., 2007. OWL-Q for Semantic QoS-based Web Service Description and Discovery. In *Service Matchmaking and Resource Retrieval in the Semantic Web*. pp. 123–138.

Laine, M., 2012. RESTful Web Services for the Internet of Things. , pp.2–4. Available at: http://media.tkk.fi/webservices/personnel/markku_laine/restful_web_services_for_the_internet_of_things.pdf.

Lamparter, S. et al., 2007. Preference-based selection of highly configurable web services. In *Proceedings of the 16th international conference on World Wide Web - WWW '07*. New York, New York, USA: ACM Press, p. 1013. Available at: http://dl.acm.org/citation.cfm?id=1242572.1242709 [Accessed June 11, 2014].

Lange, S., 2008. *Internet of Things in 2020, A ROADMAP FOR THE FUTURE.*

Lee, G.M. & Park, J., 2013. The Internet of Things - concept and problem statement draft. , pp.1–20.

Lee, M.K.O. et al., 2001. A Trust Model for Consumer Internet Shopping. *International Journal of Electronic Commerce*, 6(1), pp.75–91. Available at: http://ejournals.ebsco.com/direct.asp?ArticleID=JYN6W9T3CDGPCMJ8RTGQ.

Li, W. et al., 2011. Design and evaluation of a command recommendation system for software applications. *ACM Transactions on Computer-Human Interaction*, 18(2), pp.1–35. Available at: http://portal.acm.org/citation.cfm?doid=1970378.1970380 [Accessed June 9, 2014].

Lin, K.-J. & Tao, Y., 2006. Quality of service (qos) in web services: model, architecture and algorithms. *Quality of service (qos) in web services: model, architecture and algorithms*, (978-0-542-79027-0).

Ludwig, S. & Reyhani, S., 2006. Semantic approach to service discovery in a Grid environment. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1), pp.1–13.

Lundberg, M.J., 2009. Developing Advanced Technologies for the Imaging of Cultural Heritage Objects. , p.52.

Mabrouk, M., 2008. SOA fundamentals in a nutshell Prepare to become an IBM Certified SOA Associate. , pp.1–34.

Mahmoud, A.M., 2014. Moving Virtual University from Traditional Web to Cloud Computing. , 4(10), pp.139–144.

Manikrao, U.S. & Prabhakar, T. V, 2005. Dynamic Selection of Web Services with Recommendation System. In *In: Proceedings of the International Conference on Next Generation Web Services Practices (NWESP), IEEE Computer Society*. Press, p. 117.

Manouselis, N., Costopoulou, C. & Sideridis, E.B., 2006. Metadata for Web Portals: Developing an e-Markets ' Directory. *International Conference on (HAICTA 2006)Volos*.

Manqele, S. et al., 2012. ZAWWW 14 th ANNUAL CONFERENCE. , (November).

Maximilien, M., 2004. Toward Autonomic Web Services Trust and Selection.

Mell, P. & Grance, T., 2011. The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology.

Michael, C., Markus, L. & Roger, R., 2010. *The Internet of Things*.

Mobedpour, D. & Ding, C., 2011. User-centered design of a QoS-based web service selection system. *Service Oriented Computing and Applications*, 7(2), pp.117–127. Available at: http://link.springer.com/10.1007/s11761-011-0091-x [Accessed June 11, 2014].

Moghaddam, M. & Davis, J.G., 2015. Service Selection in Web Service Composition : A Comparative Review of Existing Approaches.

MP, S. & MN, H., 2005. *Service-oriented computing semantics, processes, Agents*", England: John Wiley & Sons LTD.

Papazoglou, M.P. & Heuvel, W.-J. Van Den, 2006. Service-oriented design and development methodology. *International Journal of Web Engineering and Technology*, 2(4), p.412. Available at: http://www.inderscience.com/link.php?id=10423.

Paridel, K. et al., 2010. Electronic Communications of the EASST Third International DisCoTec Workshop on Context-Aware Adaptation Mechanisms for Pervasive and Ubiquitous Services (CAMPUS 2010). , 28, pp.1–7.

Park, D.H. et al., 2012. A literature review and classification of recommender systems research. *Expert Systems with Applications*, 39(11), pp.10059–10072. Available at: http://www.sciencedirect.com/science/article/pii/S0957417412002825 [Accessed May 26, 2014].

Pavlik, G., Dan, H. & Tugdual, G., 2007. Next-Generation SOA Infrastructure: An Oracle White Paper. *Oracle White Paper*.

Pazzani, M.J. & Billsus, D., 2007. Content-based recommendation systems. , 13, pp.325–341. Available at: http://dl.acm.org/citation.cfm?id=1768197.1768209.

Prabhakar T & Manikrao U, 2006. Dynamic selection of web services with Recommendation system. In Kanpur: Indian Institute of Technology. Available at: http://www.cse.iitk.ac.in/users/tvp/papers/Umardand-Dynamic.pdf Last accessed April 2012.

Rader, D., 2012. How cloud computing maximizes growth opportunities for a firm challenging established rivals. *Strategy & Leadership*, 40(3), pp.36–43. Available at: http://www.emeraldinsight.com/journals.htm?issn=1087-8572&volume=40&issue=3&articleid=17031140&show=html [Accessed July 1, 2014].

Revell, S., 2013. IoT Special Interest Group. *Internet of Things (IoT) and Machine to Machine Communications (M2M) Challenges and opportunities: Final paper May 2013*, (May).

Rosenberg, L., Hammer, T. & Shaw, J., 1998. Software metrics and reliability. *Proceedings of the 9th International Symposium on Software Reliability Engineering*, pp.1–8.

Samreen, S.U. and N.A. and A., 2011. No Title. *Dynamic Service Composition in SOA and QoS Related Issues*.

Santos, N., Gummadi, K.P. & Rodrigues, R., Towards Trusted Cloud Computing.

Sathya, M. et al., 2009. Evaluation of QoS Based Web- Service Selection Techniques for Service Composition for selecting required web services according to their non-functional characteristics or quality of service (QoS). The objective of this paper presents the exploration of. , (1), pp.73–90.

Schelp, J. & Aier, S., 2009. SOA and EA - Sustainable Contributions for Increasing Corporate Agility. In *2009 42nd Hawaii International Conference on System Sciences*. IEEE, pp. 1–8. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4755468 [Accessed July 1, 2014].

Schillinger, R., 2011. Semantic Service Oriented Architectures in Research and Practice. Available at: https://books.google.co.za/books?id=z4yVZIliNykC&pg=PA218&lpg=PA218&dq=Availability+-is+important+in+presenting+services+during+runtime.&source=bl&ots=786QMzn63e&sig=fuP8S5HPG92r6k7KX9uunTNtgRQ&hl=en&sa=X&ei=MgVbVcLrH8zQ7AaM0IG4AQ&ved=0CDMQ6AEwBQ#v=onepag.

Serbanati, A., Maria, M.C. & Biader, C.U., 2011. Building Blocks of the Internet of Things : State of the Art and Beyond.

Seung-hee, B., 2012. SCALABLE HIGH PERFORMANCE MULTIDIMENSIONAL. , (February).

Shiu-li, H., 2008. COMPARISON OF UTILITY-BASED RECOMMENDATION. In *PACIS 2008 Preeding, paper 21*. Available at: http://www.pacis-net.org/file/2008/PACIS2008_Camera_Ready_Paper_012.pdf.

Silva, L.C. et al., 2008. On the Control of Adaptation in Ubiquitous Computing. In pp. 2228–2229.

Singh, M.P. & Hunhs, M.N., 2005. *Service-oriented computing semantics, processes, Agents*, England: John Wiley & Sons LTD.

Spiess, P. et al., 2009. Soa-based integration of the internet of things in enterprise services. *2009 IEEE International Conference on Web Services, ICWS 2009*, pp.968–975.

Steen, M., Pierre, G. & Voulgaris, S., 2011. Challenges in very large distributed systems. *Journal of Internet Services and Applications*, 3(1), pp.59–66. Available at: http://www.springerlink.com/index/10.1007/s13174-011-0043-x [Accessed May 20, 2014].

Stonebraker, M., Brown, P. & Martin, H., 1998. Interoperability , Distributed Applications and Distributed Databases : The Virtual Table Interface. , pp.25–33.

Su, X. et al., 2011. Towards an integrated solution to Internet of Things - A technical and economical proposal. *2011 15th International Conference on Intelligence in Next Generation Networks, ICIN 2011*, pp.46–51.

Sukhamrit, K., Kuljit, K. & Dilbag, S., 2012. Evaluating Performance of Web Services in Cloud Computing Environment withHigh Availability. *Global Journal of Computer Science and Technology: B Cloud & Distributed*, 12(11).

Teixeira, T., Hachem, S. & Georgantas, N., 2013. Service Oriented Middleware for the Internet of Things : (Invited Paper). , 257178(257178), pp.220–229.

Thoma, M., Braun, T. & Magerkurth, C., 2014. Enterprise Integration of Smart Objects using Semantic Service Descriptions.

Tran, V.X., 2008. WS-QoSOnto: A QoS Ontology for Web Services. In *2008 IEEE International Symposium on Service-Oriented System Engineering*. IEEE, pp. 233–238. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4730492 [Accessed May 27, 2014].

Tsaimos, D. et al., 2008. *Internet-of-Things Architecture*,

Tserpes, K. et al., 2012. A recommender mechanism for service selection in service-oriented environments. *Future Generation Computer Systems*, 28(8), pp.1285–1294. Available at: http://dl.acm.org/citation.cfm?id=2263483.2264532 [Accessed June 11, 2014].

Uckelmann, D., Harrison, M. & Michahelles, F., 2011. Architecting the Internet of Things D. Uckelmann, M. Harrison, & F. Michahelles, eds. *An Architecture Approach Towards the Future Internet of Things*, pp.1–25. Available at: http://link.springer.com/10.1007/978-3-642-19157-2 [Accessed May 20, 2014].

V1, Mazzorana. et al., 1994. Limited utility of emergency department thoracotomy. , pp.516–520.

Vermesan, O. & Friess, P., 2012. *The Internet of Things Converging Technologies for Smart Environments and Integrated Ecosystems*, River Publishers.

Vouk, M. a., 2008. Cloud computing — Issues, research and implementations. *ITI 2008 - 30th International Conference on Information Technology Interfaces*, pp.31–40. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4588381.

W, B. & M, W., 2003. Towards Personalized selection of web services. In Berkely. Available at: www.I3s.de/~balke/paper/www03.pdf.

Wakamatsu-cho, 2011. *A Guide to the Statistics Bureau, the Director-General for Policy Planning (statisticak standards) and the Statistical Research and Training Institute*, Available at: http://www.stat.go.jp/english/index.htm.

Wang, H., 2013. Toward a Green Campus with the Internet of Things – the Application of Lab Management. , II, pp.1–5.

Wei, L. et al., 2012. Collaborative Web Service QoS Prediction with Location-Based Regularization. *2012 IEEE 19th International Conference on Web Services*, pp.464–471. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6257841 [Accessed June 10, 2014].

Yang, X., Steck, H. & Hill, M., 2012. Circle-based Recommendation in Online Social Networks. , pp.1267–1275.

Yap, G.-E., Tan, A.-H. & Pang, H.-H., 2007. Discovering and Exploiting Causal Dependencies for Robust Mobile Context-Aware Recommenders. , 19(7), pp.1–16.

Yau, S.S. & Yin, Y., 2011. QoS-Based Service Ranking and Selection for Service-Based Systems. In *2011 IEEE International Conference on Services Computing*. IEEE, pp. 56–63. Available at: http://dl.acm.org/citation.cfm?id=2061042.2062033 [Accessed June 11, 2014].

Yu, T. & Lin, K., 2006. A Broker-Based Framework for QoS-Aware Web Service Composition. *2005 IEEE International Conference on e-Technology, e-Commerce and e-Service*, pp.22–29. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1402263.

Zhang, D., Zou, Q. & Xiong, H., 2011. CRUC : Cold-start Recommendations Using Collaborative Filtering in Internet of Things. *Energy Procedia*, (December 2011), pp.9–10.

Zhang, M. et al., 2012. A Declarative Recommender System for Cloud Infrastructure Services Selection. In p. pp 102–113.

Zhao, L. et al., 2012. Flexible service selection with user-specific QoS support in service-oriented architecture. *Journal of Network and Computer Applications*, 35(3), pp.962–973. Available at: http://www.sciencedirect.com/science/article/pii/S1084804511000671 [Accessed May 26, 2014].

Zhou, C., Chia, L.-T. & Lee, B.-S., 2005. Web Services Discovery with DAML-QoS Ontology. *International Journal of Web Services Research*, 2(2), pp.43–66.

Appendix

This section presents the user manual for the application. The name of the application is "IoT Services". The application was released from the Android studio and named "App-release-6.apk". The application is 1.74MB in size. One can receive the application via email or release it from the code. This work assumes that one can receive an application via email.

ISSA class diagram

This section presents a class diagram that describes and defines the purpose of each architectural component's functional requirement. Figure 33, presents five components and each component's function is described as follows:

- *User* - this contains three attributes; it verifies credentials for each user accessing the application.
- *User account* - this component contains user attributes and compiles a user profile. The resulting user profile created is then used to query the service.
- *Selector* - this component calls up user profile and service profile attributes, matches them and makes recommendations.
- *Registry* - this component stores the service database. It adds, deletes, updates and discovers services.
- *Administrator* - this component allows only the administrator to have access to make any changes to the application.

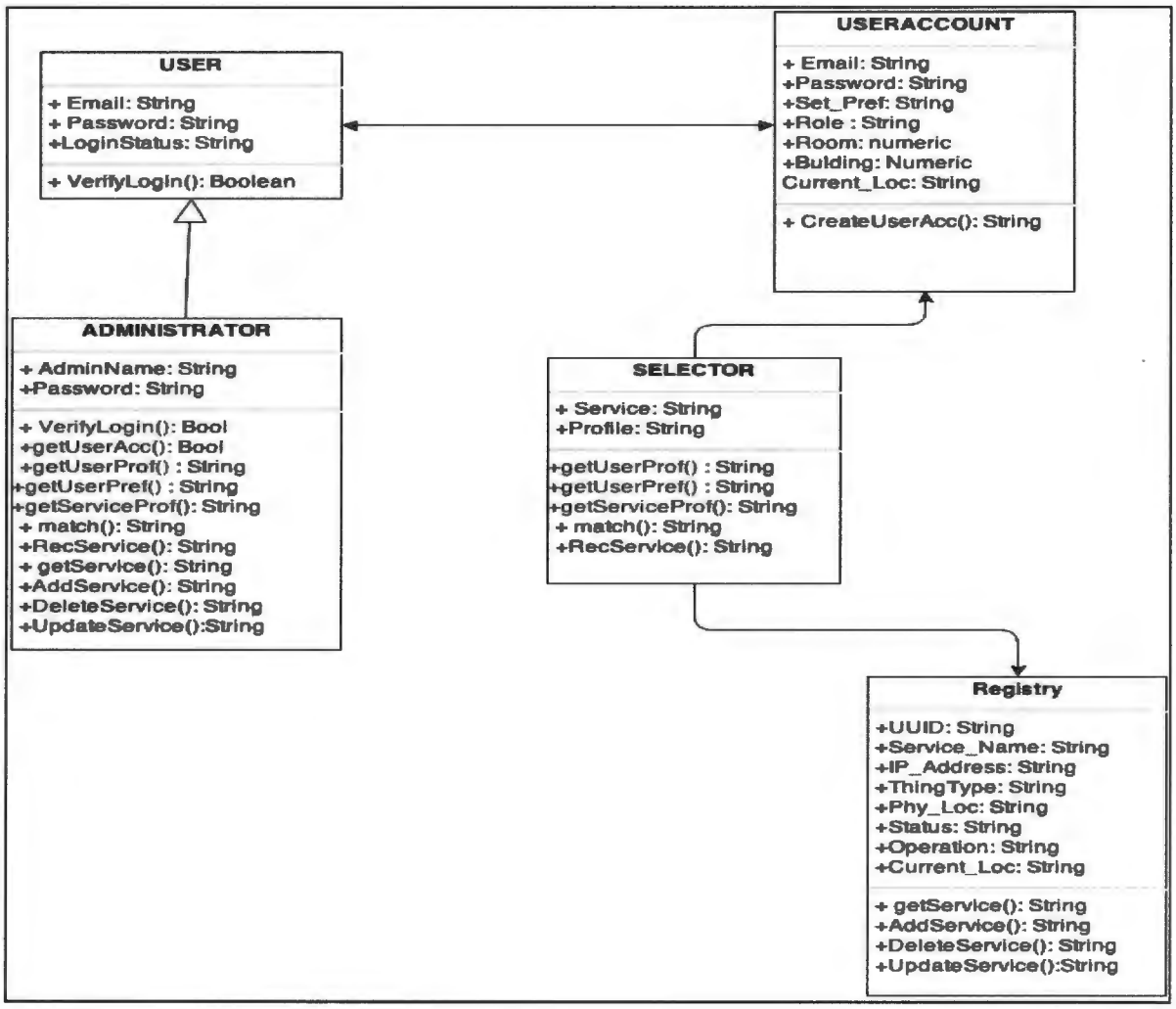


Figure 33: Class diagram

1. Installation

This section presents the steps to follow when installing the IoT Service application.

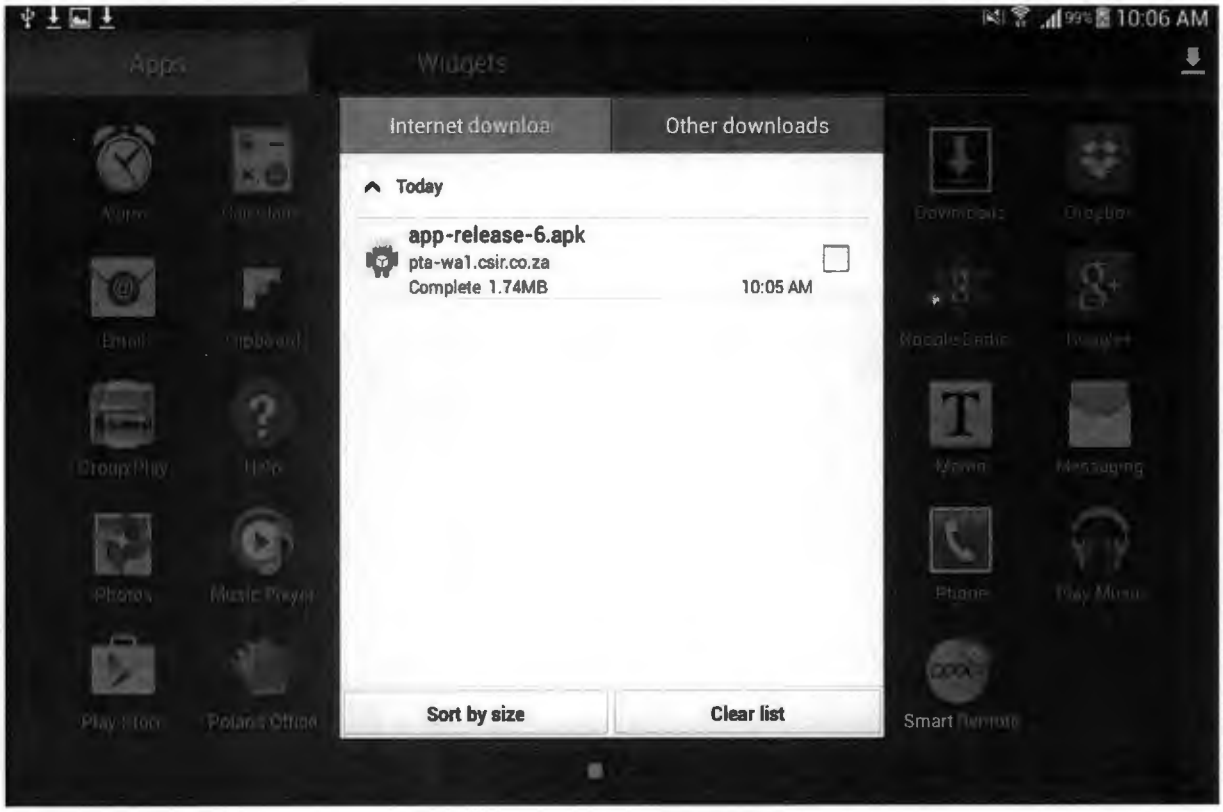


Figure 34: App install

The window in Figure 34 presents the downloaded IoT service application. The user must click on the check box. The click will lead to Figure 35.

Figure 35: Install app

Figure 35 gives the user an option to install the application or not. If yes, then the user must click "next". The "next" click leads to Figure 36.

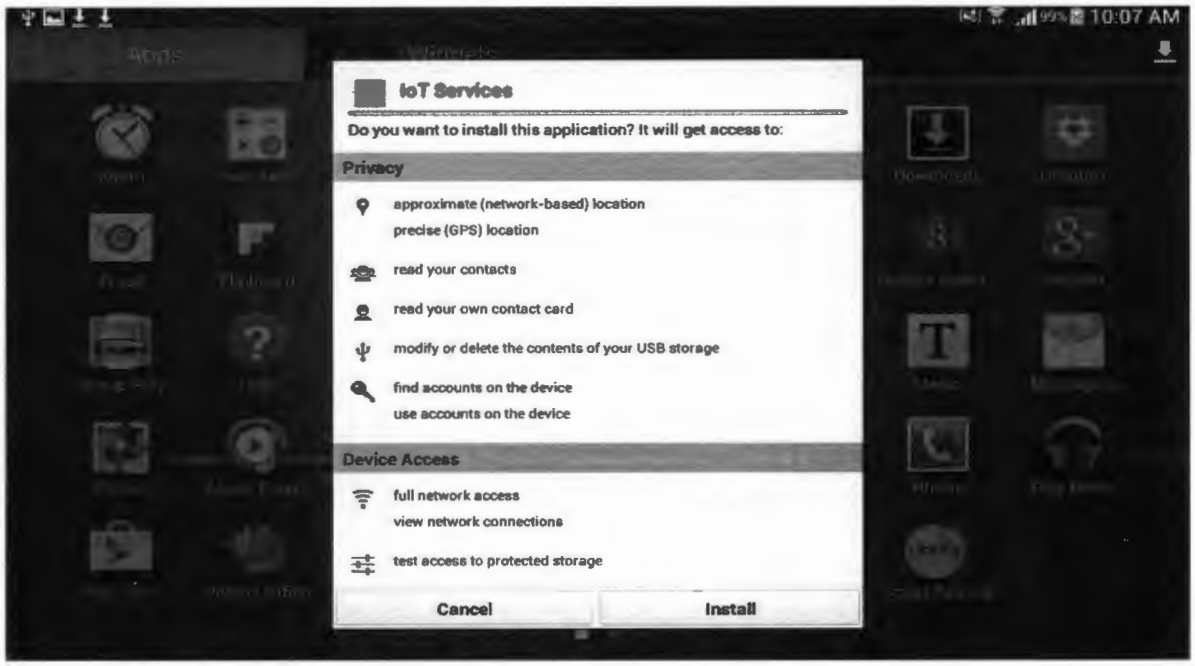


Figure 36: Install

If the user clicks "install", the app will lead to Figure 37

Figure 37: Installing

When the application is done with installation, it will lead to Figure 38

Figure 38: Open app

Figure 38 gives the user an option to "open" the application or click "done". If the user clicks "done", the application will appear on the user's home screen. If the user clicks "open", the application will automatically open itself. Now, the user is on the application.

2. Application appearance

The home screen appearance is indicated in Figure 39. The application is labelled "IoT Services". The application has a blue colour and is labelled "IoT Service".

Figure 39: IoT services app

When the user clicks on the "IoT Services" application, the application leads the user to Figure 40. If the user is not registered with the application, the user will click "Register" or the user logs into the application. Assume that the user has not yet registered, and the user will proceed to Figure 42.

Figure 40: App welcoming page

3. Registration

This page allows the user to register with the application. It becomes impossible to access this application if the user is not connected. If the field "Prefered Things" appears empty, it means the device is not connected to the Internet as indicated in Figure 41.

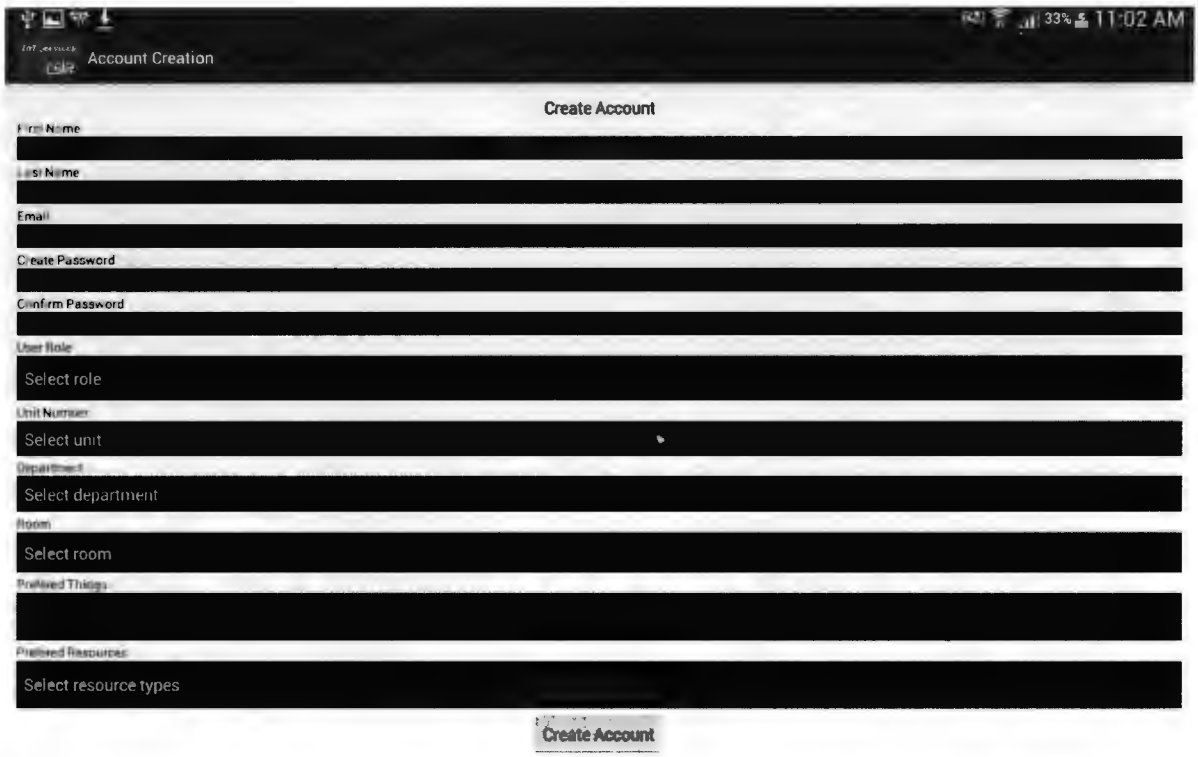


Figure 41: App connection

If the user is connected, the application will appear as indicated in Figure 42

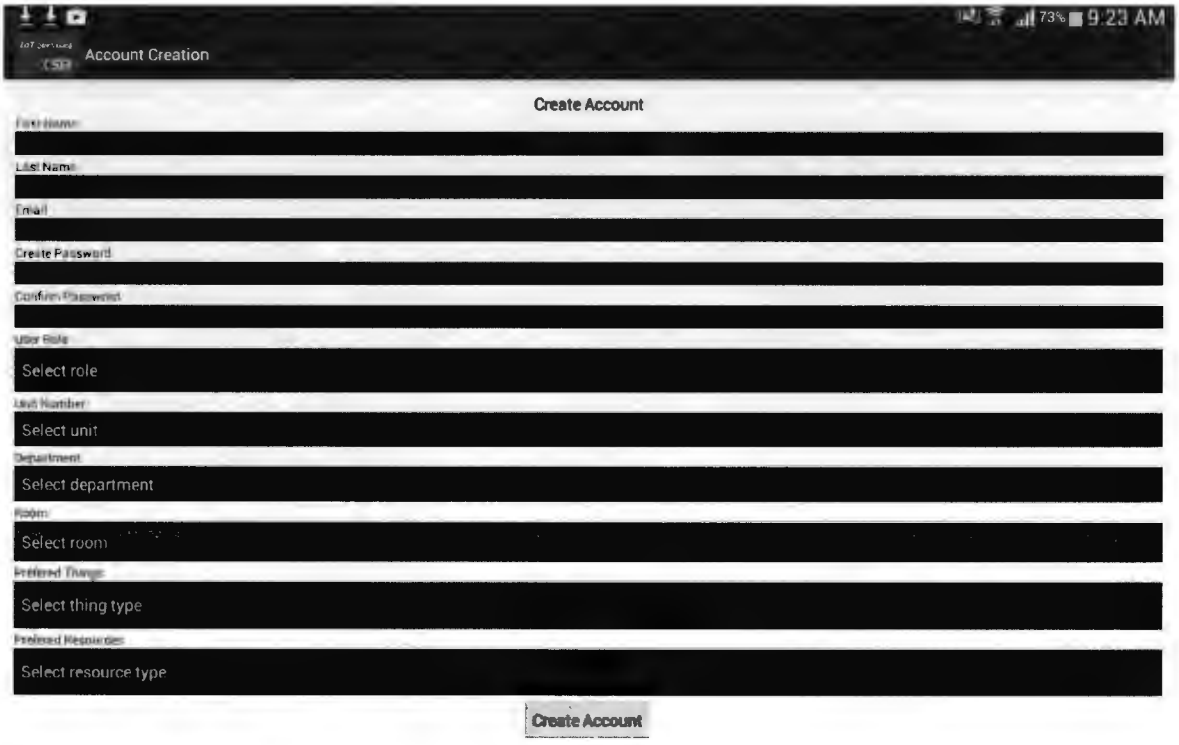


Figure 42: Create account page

Please note that the registration gives two options. The first option is to register using the "default mode" (content-based) or register using the "Role Based mode" (collaborative filtering). The modes appear in Figure 43.

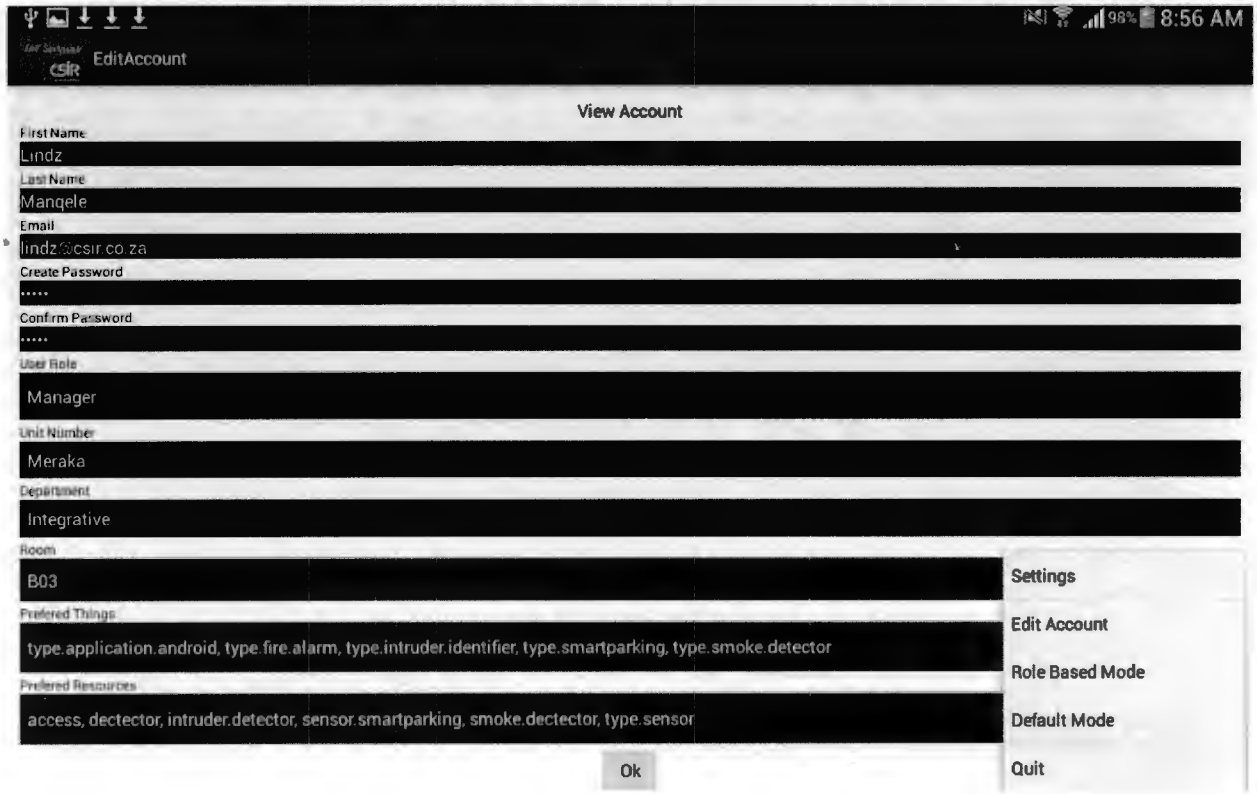


Figure 43: Registration modes

Assume the user uses the default mode. The registration window will appear as indicated in Figure 42. In the field of "Preferred Things", the application connects to the middleware and lists the available Things. The user is expected to define his user preferences as indicated in Figure 44. The user only defines preferences by selecting things, but also selects the resources of those things. Please note that, even if the user defines the user preferences, there are other steps that need to be followed in order to further filter the long list of services. Figure 44 and Figure 45 show how the user defines preferences using the "Things" and "Resource" descriptions.

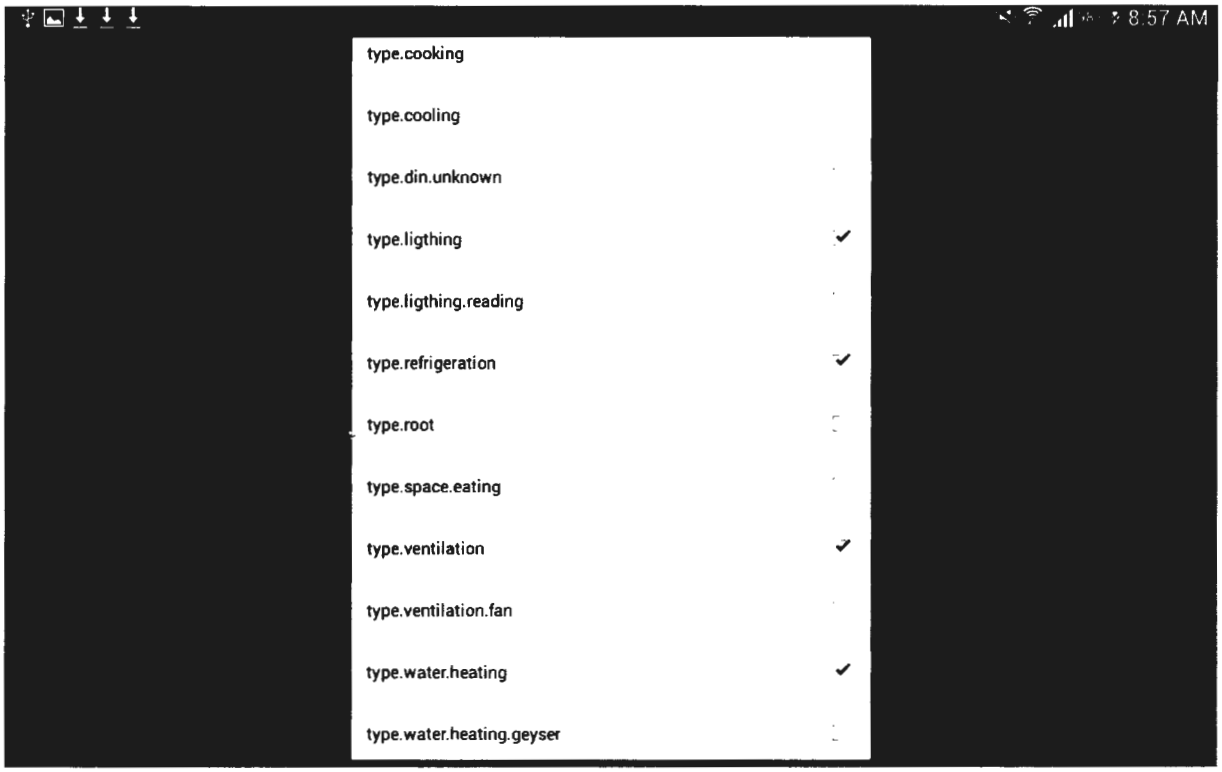


Figure 44:Things

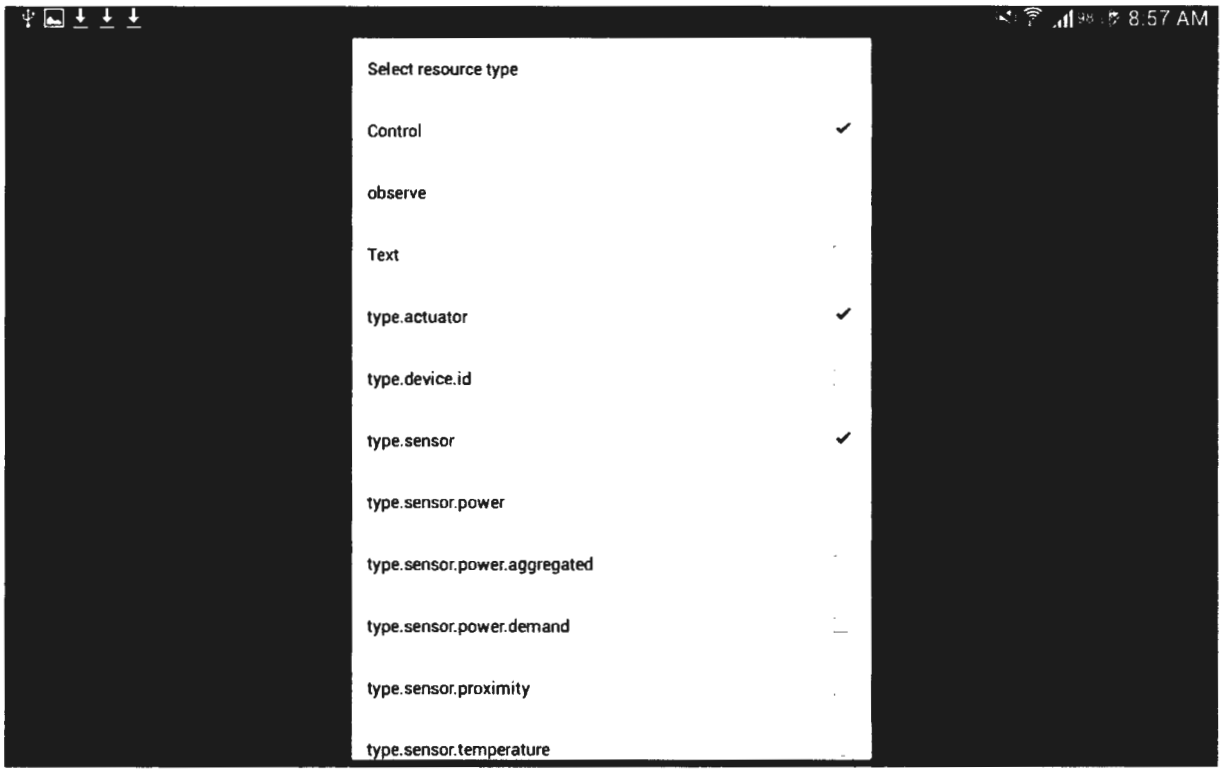


Figure 45: Resources

When the user has selected the "Things" and "Resources", the user clicks "ok" as indicated in Figure 43.

The "Role based mode" allows the user to register based on the role. This mode does not require the user to select "Things" or "Resources", as indicated in Figure 46.

Figure 46: Role-based mode

4. Authentication

If the user has registered on the application, the user proceeds to Figure 47. The user will be required to enter the credentials.

Figure 47: Login page in app

If the user has entered the wrong credentials, the application will give the user an option to be reminded about the password by clicking on "Forgot password", as indicated in Figure 48.

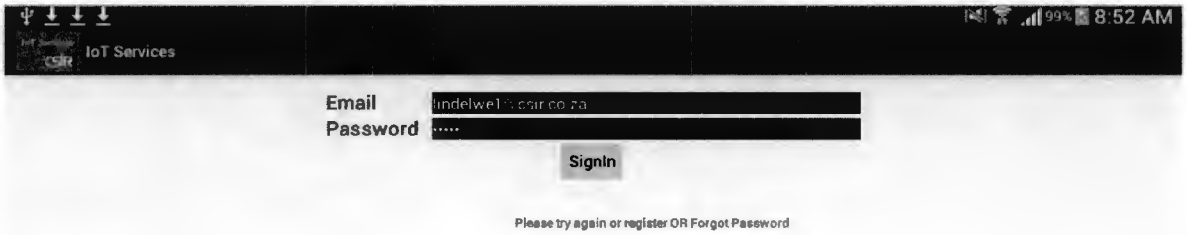


Figure 48: Wrong password

A click on "forgot password" will proceed to Figure 49.

Figure 49: Password reminder

Figure 50: Correct password

If the user entered the correct password as the credentials appear in Figure 50, the user will proceed to the next step of recommendation, as indicated in Figure 52.

5. Recommendation

This section recommends the services that match user preferences. Figure 51 shows the list of recommended services with the value of the recall. The value of recall will be discussed later in experimental evaluation. Figure 52 shows the recommended list for "Lindz". The same figure shows the options on the recommended list page. The user can "Refresh" the list. Refreshing the list makes the services the user once utilised the priority. The user also has an option to clear the history.

Figure 51: Recommended list with recall

Figure 52: Recommended services

6. Update account

Factors like change of the role, change of the office, change of preference, etc., can lead the user to update the profile. The user simply goes to "Update Profile" as indicated in Figure 52. A click on "Update Profile" leads to Figure 53. In Figure 53, the user selects "Edit account", chooses mode, edits the account and saves the changes.

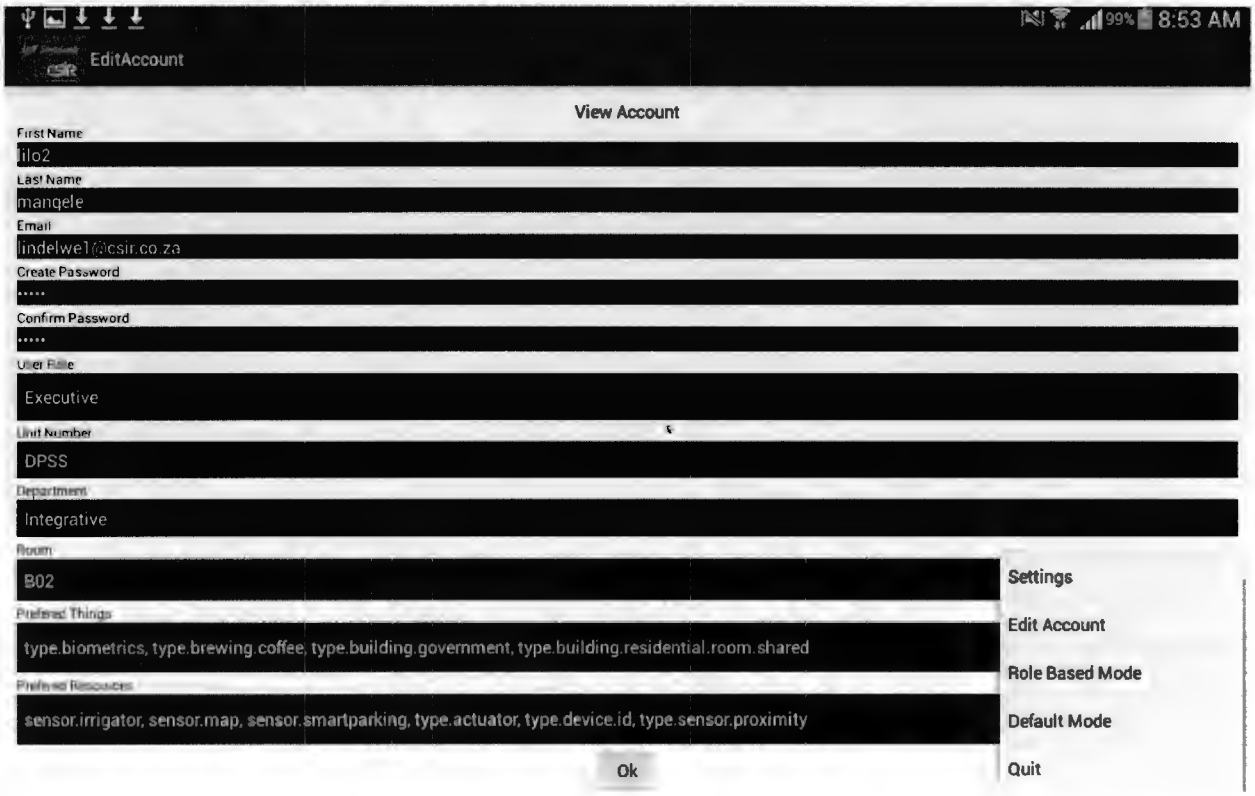


Figure 53: Edit account

7. Service invocation

This section presents the interface of service invocation. Figure 54 indicates that this window allows the user to submit changes the user wants to make on the service. The red arrow is normally used to decrease/switch off the "Thing". The blue arrow is used to increase/ switch off the "Thing". The green button submits the changes to be made on the "Thing" to the server.

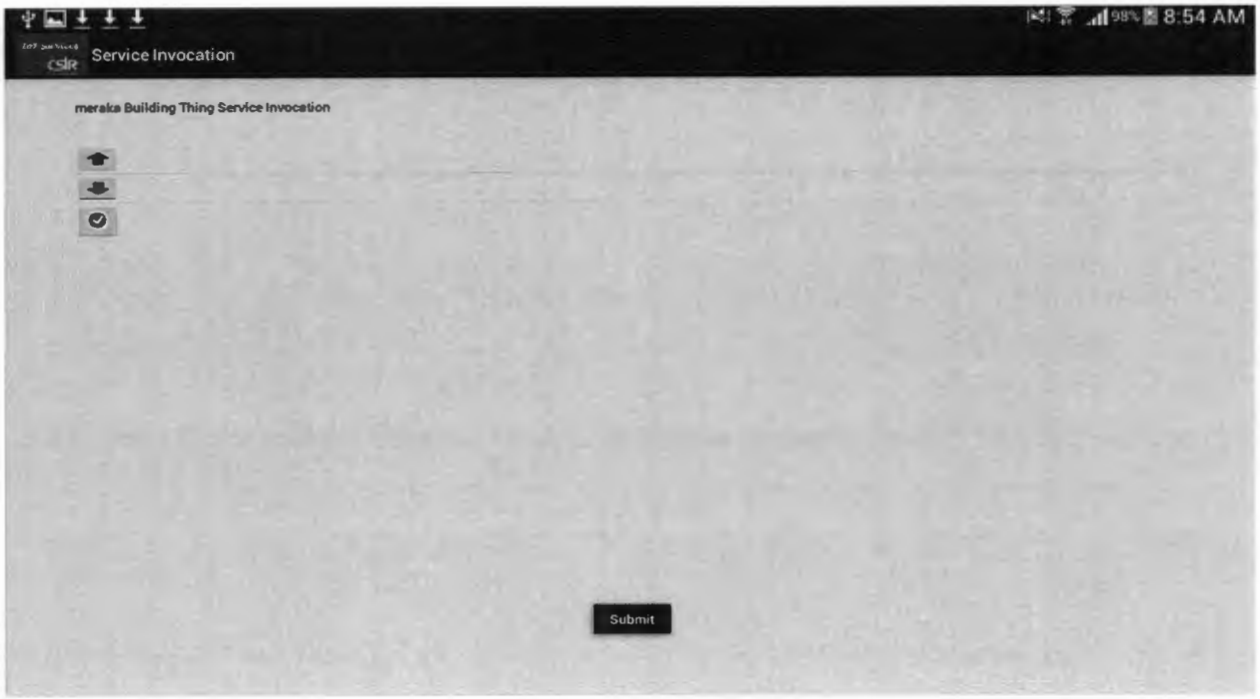


Figure 54: Service invocation

8. Quit application or logout

Figure 53 indicates the option to quit the application. Figure 52 indicates the options to logout from the application.

9. Experimental evaluation

This section presents the evaluation conducted to check the performance and the effectiveness of each technique (content-based or collaborative filtering). The values of the recall and response time are presented under the folder "My Files". Then go to the "Experiments" folder as indicated in Figure 55. A click on "experimentaldata.txt" proceeds to Figure 56.

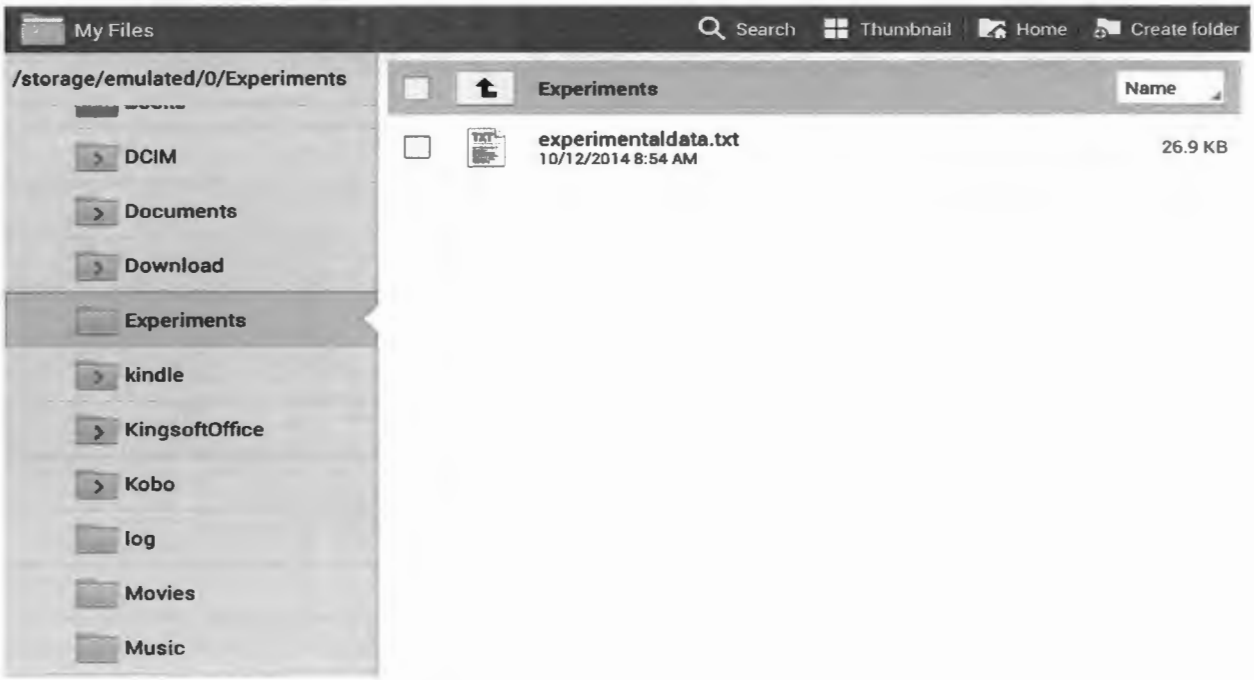


Figure 55: Experiments

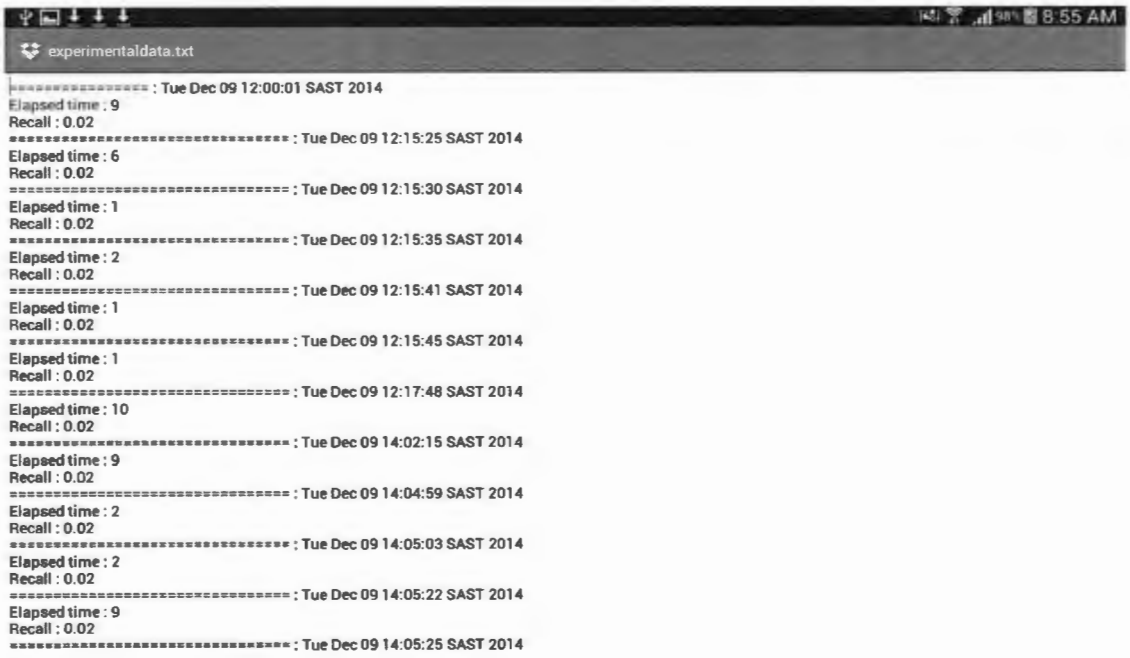


Figure 56: Values of recall and response time

Figure 56 shows the value of the recall. Recall is calculated using the number of detected services over the number of available services. The figure also shows the values the response time.

10. Uninstall the application

This section presents the steps to follow to uninstall the application. Go to "Settings", then to "Application manager". There is an "uninstall" option on the window as indicated in Figure 57.

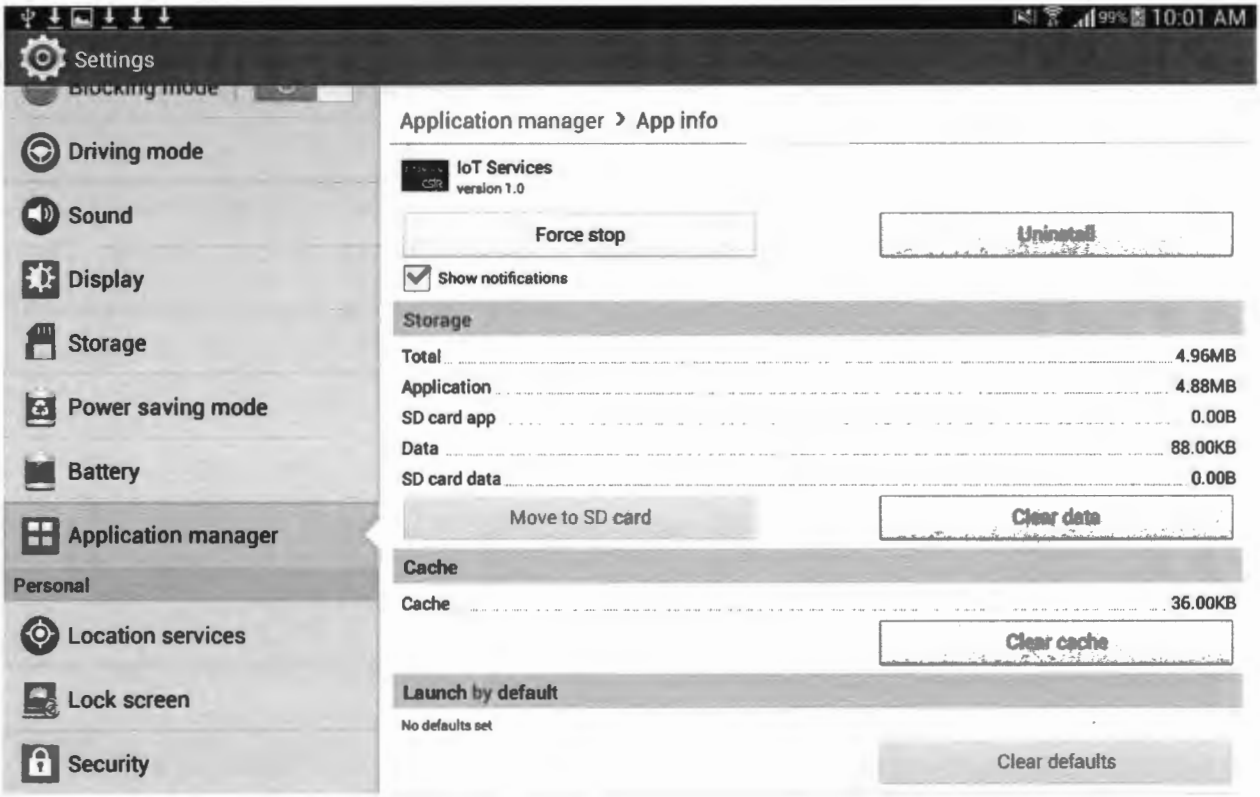


Figure 57: Uninstall 1

If the user chooses the "Uninstall" button, the application will lead to Figure 58. Figure 58 gives the option to confirm whether the user really wants to uninstall or it was a mistake. If the user intended to uninstall the application, the user clicks "ok", and the application will be uninstalled.

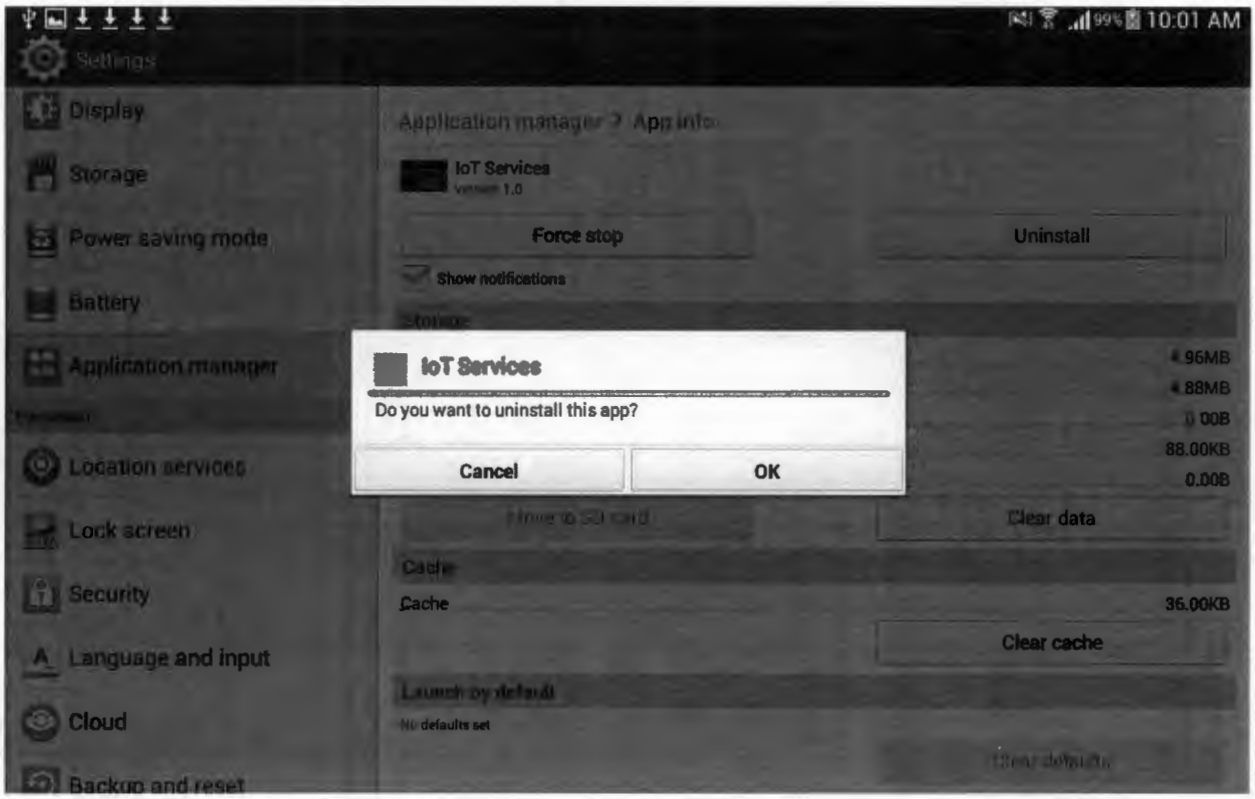


Figure 58: Uninstall 2