

THE UNIVERSITY OF HUDDERSFIELD

Investigation of a Novel Formal Model for Mobile User Interface Design

by

Ragab Basher Ihnissi

A thesis submitted in partial fulfilment for the
degree of Doctor of Philosophy

in the

School of Computing and Engineering

November 2017

Copyright Statement

- The author of this thesis (including any appendices and/or schedules to this thesis) owns any copyright in it (the “Copyright”) and s/he has given The University of Huddersfield the right to use such copyright for any administrative, promotional, educational and/or teaching purposes.
- Copies of this thesis, either in full or in extracts, may be made only in accordance with the regulations of the University Library. Details of these regulations may be obtained from the Librarian. This page must form part of any such copies made.
- The ownership of any patents, designs, trademarks and any and all other intellectual property rights except for the Copyright (the “Intellectual Property Rights”) and any reproductions of copyright works, for example graphs and tables (“Reproductions”), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property Rights and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property Rights and/or Reproductions.

Abstract

Mobile user interfaces are becoming increasingly complex due to the expanding range of functionalities that they incorporate, which poses significant difficulties in software development. Formal methods are beneficial for highly complex software systems, as they enable the designed behaviour of a mobile user interface (UI) to be modelled and tested for accuracy before implementation. Indeed, assessing the compatibility between the software specification and user requirements and verifying the implementation in relation to the specification are essential procedures in the development process of any type of UI. To ensure that UIs meet users' requirements and competences, approaches that are based on interaction between humans and computers employ a variety of methods to address key issues.

The development of underlying system functionality and UIs benefit from formal methods as well as from user-interface design specifications. Therefore, both approaches are incorporated into the software development process in this thesis. However, this integration is not an easy task due to the discrepancies between the two approaches. It also includes a method, which can be applied for both simple and complex UI applications. To overcome the issue of integrating both approaches, the thesis proposes a new formal model called the *Formal Model of Mobile User Interface Design (FMMUID)*. This model is devised to characterise the composition of the UI design based on hierarchical structure and a set theory language.

To determine its applicability and validity, the FMMUID is implemented in two real-world case studies: the quiz game *iPlayCode* and the social media application *Social Communication (SC)*. A comparative analysis is undertaken between two case studies, where each case study has three existing applications with similar functionality in terms of structure and numbers of elements, functions and colours. Furthermore, the case studies are also assessed from a human viewpoint, which reveals that they possess better usability.

The assessment supports the viability of the proposed model as a guiding tool for software development. The efficiency of the proposed model is confirmed by the result that the two case studies are less complex than the other UI applications in terms of hierarchical structure and numbers of elements, functions and colours, whilst also presenting acceptable usability in terms of the four examined dimensions: usefulness, information quality, interface quality, and overall satisfaction. Hence, the proposed model can facilitate the development process of mobile UI applications.

Acknowledgements

Firstly and most importantly, I would like to thank Allah the Almighty from the depths of my heart for the guidance and inspiration. Everything great that has happened to me in my life have resulted from his help, mercy and love.

I wish to send my deepest gratitude and appreciation Professor Joan Lu (my first supervisor) for her exemplary supervision, help, feedback and for pushing me with encouragement through the difficult times. I thank her particularly for her patience and valuable efforts to encourage me to conduct an affluent research.

I also wish to express thanks to my second supervisor, Dr. Gary Allen, and my third supervisor Dr. Qiang Xu, for their continuous friendly support, encouragement and feedback. I am sincerely thankful to my parents, and wish to give them huge acknowledgement for their undying love. I would not have accomplished this goal without their continuous support, patience and prayers.

I express my deepest thanks to my siblings for their endless love, support, prayers and encouragement throughout my studies.

I would like to express my gratitude and thanks to my beloved wife. I cannot put into words how important her inspiration and support have been throughout the course of my PhD. Her consistent support and encouragement has meant so much to me. Thank you so much my love for everything you have done for me and our daughter. We have had some truly unforgettable experiences throughout our journey.

I would like my lovely daughter Awsema to know that she is the light of my life. A consistent source of motivation and inspiration. Dearest daughter, I envisage my future through your dotting eyes. This thesis is a small present for you. May Allah bless you.

Lastly, I extent my personal thanks to all my colleagues and friends in Britain and Libya for their continuous support and encouragement.

Table of Contents

Abstract	I
Acknowledgements.....	II
List of Contents.....	III
List of Figures	VII
List of Tables	VIII
List of Abbreviations	X
List of Nomenclature	XI
Chapter 1: Introduction.....	1
1.1 General Background of the Research	3
1.2 Outlining the Research Problem	5
1.3 Motivation	6
1.4 Research Questions and Hypotheses.....	7
1.5 Research Aim and Objectives	8
1.6 Contributions	8
1.7 Research Structure	9
Chapter 2: Literature Review: Methods of Formal Modelling of User Interface Design.....	11
2.1 Introduction.....	11
2.2 Human-Computer Interaction	11
2.3 User Interface.....	12
2.4 Instruments for UI Development.....	13
2.4.1 Window Managers.....	13
2.4.2 Toolkits	13
2.4.2.1 Direct Graphical Specification Instruments.....	14
2.4.2.2 Model-based Tools.....	14
2.5 User Interface Complexity.....	14
2.5.1 Measures of UI Complexity.....	15
2.5.2 Summary	17
2.6 Usability of User Interface.....	18

2.7	Software Modelling.....	19
2.8	Development of a UI Based on Modelling.....	21
2.8.1	First- and Second-Generation MBUID Approaches.....	21
2.8.2	Third- and Fourth-Generation MBUID Approaches	23
2.8.3	Summary.....	25
2.9	Formal Specification Languages	26
2.10	Formal Models	27
2.10.1	Development of Interface Models	28
2.10.2	Summary.....	32
2.11	Differences between this Thesis and Previous Work	33
2.12	Chapter Summary.....	34
Chapter 3:	Techniques and Methodologies.....	36
3.1	Introduction.....	36
3.2	Process of FMMUID Development.....	36
3.3	Techniques of Analysis	37
3.3.1	First Phase.....	38
3.3.1.1	Choice of Case Studies	38
3.3.1.2	Choice of Existing Applications	39
3.3.1.3	Design Analysis	40
3.3.2	Second Phase: Usability Assessment Methods.....	41
3.3.2.1	Usability.....	41
3.3.2.2	Assessment of Usability.....	44
3.3.2.3	Usability Testing.....	50
3.3.2.4	Research Participants	50
3.3.2.5	Questionnaire	51
3.3.2.6	Procedure	54
3.3.2.7	Statistical Analysis of Quantitative Data	55
3.4	Chapter Summary.....	58
Chapter 4:	Formal Model of Mobile User Interface Design (FMMUID).....	59
4.1	Introduction.....	59
4.2	FMMUID Development.....	60
4.3	Summary	69

Chapter 5: FMMUID Using Case-Studies	70
5.1 Introduction.....	70
5.2 Case Study 1: iPlayCode Application	71
5.2.1 iPlayCode Application Design Screens	72
5.3 Case Study 2: SC Design.....	75
5.3.1 SC Application Design Screens.....	77
5.4 Summary	83
Chapter 6: Comparison and Model Validation	85
6.1 Introduction.....	85
6.2 Analysis of the Structure of Application Design	85
6.2.1 Case Study 1: iPlayCode Comparison Design.....	85
6.2.2 iPlayCode Screen Comparison	86
6.2.3 Case Study 2: SC Comparison Design.....	89
6.2.4 SC Screen Comparison	89
6.3 The Comparison of iPlayCode and SC with Other Application Elements	92
6.3.1 ANOVA Test of Single and Multiple-Screens for iPlayCode and SC	93
6.3.1.1 Outcomes of Case Study 1	93
6.3.1.2 Outcomes of Case Study 2.....	94
6.4 Questionnaire Analysis and Results	95
6.4.1 Sample Distribution.....	95
6.4.2 Reliability.....	95
6.4.3 Case Study 1: Results.....	96
6.4.3.1 Usefulness	98
6.4.3.2 Information Quality	98
6.4.3.3 Interface Quality	98
6.4.3.4 Overall Satisfaction.....	98
6.4.4 Case Study 2: Results.....	99
6.5 One-way ANOVA Test Results of both Studies.....	100
6.6 Summary	101
Chapter 7: Evaluation of Proposed Model.....	104
7.1 Introduction.....	104
7.2 Complexity.....	104

7.2.1	Navigation.....	105
7.2.2	Function	109
7.2.3	Colour	110
7.3	Assessment of Single and Multiple-Screens in the Two Case Studies	112
7.4	Assessment of Usability	113
7.5	Validation.....	114
7.6	Summary and Assessment of Research Questions	115
Chapter 8:	Conclusions.....	117
8.1	Result Overview.....	118
8.2	Research Contributions	120
8.3	Research Limitations	123
8.4	Further Research	124
8.5	Published Papers	124
References	125
Appendices	154
Appendix A:	Quiz Game Applications Description and Screen Shoots.....	154
Appendix B:	Social Media Applications Description and Screen shots	160
Appendix C:	Questionnaire of Usability	165
Appendix D:	Comparative Analysis of UI Design	168
Appendix E:	Comparative Analysis of UI Elements	170
Appendix F:	One-way ANOVA Test Results for iPlayCode, DK, Duolingo and C/C++..	176
Appendix G:	One-way ANOVA Test Results for SC, Google+, Facebook and Guntree...	183
Appendix H:	One-way ANOVA Test Analysis for Questionnaire	186

List of Figures

Figure 3.1: Hierarchical Structure for FMMUID.	37
Figure 5.1: Hierarchical Structure Screen Design for iPlayCode Application.	71
Figure 5.2: Screen Shots of Model Screens for iPlayCode UI.....	74
Figure 5.3: Hierarchical Structure Screen Design for SC Application.	76
Figure 5.4: Screen Shots of Model Screens for SC UI.....	79
Figure 5.5: Generating the Third Screen (S_3) in a Hierarchical Structure.	81
Figure 5.6: Generating the Fourteenth Screen (S_{14}) in a Hierarchical Structure.	82
Figure 5.7: Generating the Fifteenth Screen (S_{15}) in a Hierarchical Structure.	83
Figure 6.1: Comparison of UI Elements on Screen S_s (single screen) between Quiz Game Apps.....	86
Figure 6.2: Comparison of UI Elements on Screen S_i	87
Figure 6.3: Comparison of UI Elements on Screen S_g	87
Figure 6.4: Comparison of UI Elements on Screen S_{info}	88
Figure 6.5 Comparison of UI Elements on Screen S_s (single screen) between Social Media Apps.....	90
Figure 6.6: Comparison of UI Elements on Screen S_i	90
Figure 6.7: Comparison of UI Elements on Screen S_g	91
Figure 6.8: Comparison of UI Elements on Screen S_{info}	92
Figure A.1: Screen Shots of Model Screens for DK UI.....	155
Figure A.2: Screen Shots of Model Screens for Duolingo UI.....	157
Figure A.3: Screen Shots of Model Screens for C/C++ UI.....	159
FIGURE B.1: Screen Shots of Model Screens for Google+ UI.	161
Figure B.2: Screen Shots of Model Screens for Facebook UI.	162
Figure B.3: Screen Shots of Model Screens for Guntree UI.....	164
Figure D.1: Comparison of Hierarchical Structure Design of iPlayCode with Other Mobile Applications.....	168
Figure D.2: Comparison of Hierarchical Structure Design of SC with Other Mobile Applications.....	169

List of Tables

Table 3:1 The Mobile Applications Selected for the First Case Study (iPlayCode) Alongside their Descriptions.....	39
Table 3:2: The Mobile Applications Selected for the Second Case Study (SC) Alongside their Descriptions.....	39
Table 3:3: iOS Mobile Applications Selected for Comparative Analysis and their Features. ..	40
Table 3:4: Scores Obtained for Each Statement in a Questionnaire Based on the Likert Scale.	52
Table 3:5: The Weighted Mean Criterion in the Likert Scale.....	52
Table 3:6 Parametric and Nonparametric Testing.....	56
Table 6:1: Significance of Variance between Single Screens for iPlayCode and other Applications.....	93
Table 6:2: Significance of Variance between Single Screens for SC and other Applications..	94
Table 6:3: Sample Distribution of Usability-interface Data on the iOS Mobile Apps.	95
Table 6:4: The Reliability of iPlayCode and SC Compared to other Applications.....	96
Table 6:5: Questionnaire Scores for the iPlayCode, DK, Duolingo and C/C++ Applications. 97	
Table 6:6: Questionnaire Scores for the SC, Google+, Facebook and Gumtree Applications. 99	
Table 6:7: Significance of Variance between Case Studies and Existing Applications.	101
Table E:1: The Comparison of UI Elements on Single Screens for Quiz Game Apps.	170
Table E:2: Comparing Different Average Categories of Single Screens for Quiz Game Apps.	171
Table E:3: The Comparison of UI Elements on Single Screens for Social Media Apps.....	172
Table E:4: Comparing Different Average Categories of Single Screens for Social Media Apps.	173
Table E:5: Comparing Different Categories of Multiple Screens for Quiz Game Apps.....	174
Table E:6: Comparing Different Categories of Multiple Screens for Social Media Apps.....	175
Table F:1: ANOVA Test of S_s Screen for iPlayCode and other Apps.	176
Table F:2: ANOVA Test of S_i Screen for iPlayCode and other Apps.	176
Table F:3: ANOVA Test of S_g Screen for iPlayCode and other Apps.....	177
Table F:4: ANOVA Test of S_{info} Screen for iPlayCode and other Apps.....	177
Table F:5: T-test Results for S_i Screen between iPlayCode and DK Apps.	178
Table F:6: T-test Results for S_i Screen between iPlayCode and Duolingo Apps.	178
Table F:7: T-test Results for S_i Screen between iPlayCode and C/C++ Apps.	179

Table F:8: T-test Results for S_i Screen between DK and Duolingo Apps.....	179
Table F:9: T-test Results for S_i Screen between DK and C/C++ Apps.....	180
Table F:10: T-test Results for S_i Screen between Duolingo and C/C++ Apps.	180
Table F:11: F-Test Results between iPlayCode and DK Apps for S_i Screen.....	181
Table F:12: ANOVA Test of Multiple-screens for iPlayCode and other Apps.....	182
Table G:1: ANOVA TEST of S_s Screen for SC and other Apps.	183
Table G:2: ANOVA Test of S_i Screen for SC and other Apps.	183
TABLE G:3: ANOVA Test of S_g Screen for SC and other Apps.	184
Table G:4: ANOVA Test of S_{info} Screen for SC and other Apps.	184
Table G:5: ANOVA Test of Multiple-screens for SC and other Apps.	185
Table H:1: Usefulness Attribute Results.	186
Table H:2: Information Quality Attribute Results.	186
Table H:3: Interface Quality Attribute Results.....	187
Table H:4: Overall Satisfaction Attribute Results.	187
Table H:5: Usefulness Attribute Results.	188
Table H:6: Information Quality Attribute Results.....	188
Table H:7: Interface Quality Attribute Results.....	189
Table H:8: Overall Satisfaction Attribute Results.	189
Table H:9: Normal Data Distribution for iPlayCode App.	190
Table H:10: Normal Data Distribution for DK App.	190
Table H:11: Normal Data Distribution for Duolingo App.	191
Table H:12: Normal Data Distribution for C-C++ App.....	191
Table H:13: Normal Data Distribution for SC App.	192
Table H:14: Normal Data Distribution for Google+ App.....	192
Table H:15: Normal Data Distribution for Facebook App.	193
Table H:16: Normal Data Distribution for Gumtree App.	193

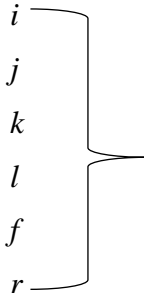
List of Abbreviations

AUI	ABSTRACT USER INTERFACE
CBSD	COMPONENT-BASED SOFTWARE DEVELOPMENT
CICM	COMPONENT INTERFACE COMPLEXITY METRIC
CIM	COMPUTATION INDEPENDENT MODEL
CSUQ	COMPUTER SYSTEM USABILITY QUESTIONNAIRE
DUI	DISTRIBUTED USER INTERFACE
FILL	FORMAL INTERACTION LOGIC LANGUAGE
FMMUID	FORMAL MODEL OF MOBILE USER INTERFACE DESIGN
GOMS	GOALS, OPERATORS, METHODS AND SELECTION
HCD	HUMAN-CENTRED DESIGN
HCI	HUMAN COMPUTER INTERACTION
IFML	INTERACTION FLOW MODELLING LANGUAGE
ISO	INTERNATIONAL STANDARDISATION ORGANISATION
ITS	INTERACTIVE TRANSACTION SYSTEMS ITS
LOC	LINES OF CODE
MBUID	MODEL-BASED UI DEVELOPMENT
MBUIDE	MODEL-BASED UI DEVELOPMENT ENVIRONMENT
MDA	MODEL-DRIVEN ARCHITECTURE
MDE	MODEL-DRIVEN ENGINEERING
PIM	PLATFORM INDEPENDENT MODEL
SC	SOCIAL COMMUNICATION
SUS	STUDIED SYSTEM
UE	USABILITY ENGINEERING
UEMs	USABILITY EVALUATION METHODS
UI	USER INTERFACE
UID	USER INTERFACE DESIGN
UIDLs	UI DESCRIPTION LANGUAGES
UIML	USER INTERFACE MARK-UP LANGUAGE
UIMS	USER INTERFACE MANAGEMENT SYSTEM
UML	UNIFIED MODELLING LANGUAGE
UsiXML	USER INTERFACE eXTENSIBLE MARKUP LANGUAGE
WYSIWYG	WHAT YOU SEE IS WHAT YOU GET

List of Nomenclature

S	Screen
I	Components
C	Colour
F	Function
\cup	Union
\supseteq	Superset

Subscript

u	Name or Number of Screen
S	Start
l	Level
q	Question
r	Results
fr	Final Result
g	General (calculate the result, search, etc.)
$info$	Information
p	Control Process Components
c	Contents/Search
v	Vision
i j k l f r	 Number of sub-screens
w	Number of Components, Colours or Functions for Each Screen
t	Number of Components that are Available on Each Screen
a	Number of Colours or Functions for Each Component
x	Number of Colours Including Functions
y	Number of Functions for Colours

Chapter 1: Introduction

Across various types of electronic devices, the interactivity, complexity, and ubiquity of mobile interfaces are constantly increasing, as these interfaces are used increasingly often in daily activities (Golovine, 2013). Both academics and industry specialists are becoming increasingly aware of how significant the User Interface (UI) element is in an interactive application, thereby prompting efforts to ensure compatibility between the user interface and different activities within settings that are accessible to as many users as possible (Santoro, 2005). Indeed, the creation of software and interfaces that satisfy the varying needs of users is considered paramount nowadays (Golovine, 2013). The ultimate goal is to develop interfaces that are flexible, efficient, productive, easy to use, and have minimal errors. However, this goal is far from being achieved, thereby leaving users dissatisfied with the poor usability of some applications. Therefore, numerous companies, designers and developers are prioritising the creation of interactive systems with a high degree of usability. Unfortunately, there are many obstacles opposing the creation of such systems that may cause designers to struggle in developing user interfaces that are capable of supporting various user activities (Golovine, 2013).

The complexity of the process of software development cannot be underestimated and expands with the introduction of new hardware types, interaction modes, and usage contexts. Furthermore, as software systems become ever larger and more complex, a new design problem emerges in addition to computation algorithms and data configurations, namely, the design and specification of an entire system structure (Garlan & Shaw, 1993).

Design-based UI components within the context of the computer science branch of software engineering may provide a potential solution to the issue of growing complexity. In employing such components, the purpose is to promote the novel approach of Formal Model of Mobile User Interface Design (FMMUID) to encourage the creation of practical UIs as well as to make design and development less time- and effort-consuming (see section 2.12).

The approaches to software development change in tandem with emerging software types and uses. However, some fundamental principles of software development have remained constant, as the goal of creating efficient and error-free software has stayed the same throughout the transformations that techniques and practical applications have undergone. Among these principles is the principle known as the formal model, which maintains that before software is implemented, it must be designed, developed, and tested with accurate and

reliable methods. Grounded in mathematics, formal methods are employed for developing software systems and offer a structure for methodical system specification, development, and analysis. Furthermore, these methods have a formal specification language with a formal syntax and semantics and are underpinned by a logical inference framework. A logical inference system determines a consequence connection normally specified by related regulations that specify a set of ideally created sentences in the specification language to a varied of appropriately created sentences (Wing, 1990). The formal method is well suited for complex UI applications of medium to large size. It enables designers not to concern themselves with low-level details but instead focus on the logical specification and analysis of interactive software applications based on the identification of pertinent abstractions (Golovine, 2013). This approach also highlights that software development should be centred on the users and their needs, and design methods should be employed accordingly (Bowen, 2008).

To ensure the reliability of particular UI designs prior to implementation, compliance with some major principles should be observed, irrespective of which general development method is adopted. The purpose of the thesis is to create and implement UIs on the basis of a robust software engineering approach, namely, the FMMUID, as well as to gain user feedback about the UI design, its functionality and the extent to which it satisfies user needs by subjecting the UI design to user testing. In addition to the FMMUID, this thesis employs the methods of Human-Computer Interaction (HCI) (Satyavathy & RachelBlessie, 2017) in the process of software development. FMMUID and HCI are increasingly being applied together in software projects, as they serve to complement one another.

The realisation that it is useless to develop sophisticated software if user interaction with its interfaces is inadequate prompted the emergence of HCI. It is no easy task to develop a UI (Myers, 1993); because it depends to a significant degree on the talent or skill of the designers, UI design can be perceived as a craft or even an art (Duce, 1995). The engineering issue of creating systems that are capable of meeting user needs is the main problem for HCI, and in the context of this issue, UI software development makes up the largest part of the engineering endeavours (Myers & Rosson, 1992). Reflecting what the system is capable of and its intended use, the functionality that is incorporated into software during the design process reveals itself during system use. Meanwhile, software usability can be defined as how effectively the software performs the various tasks or activities that are required by users. For

a software system to be considered efficient, its functionality and usability must be balanced (Institution, 1998; Karray et al., 2008).

A matter of key importance in HCI is the provision of uncomplicated UIs. The formulation of a system's quality attributes is dependent on usability assessment (Odeh & Adwan, 2009). To improve productivity and experience, minimise the error rate, and ensure system efficiency, users are often involved in the design or redesign of software system UIs within the field of Usability Engineering (UE) (Gulliksen et al., 2009; Gulliksen, 2007; Hix et al., 2004; Macleod, 1994). Redesigning the entire system development process so that it centres on UE knowledge, techniques, and tasks is the aim of the UE lifecycle. Furthermore, by iterating a series of UE techniques (e.g., prototyping, usability assessment, and user-interface mock-ups), the UE lifecycle is geared towards the achievement of the established usability objectives. Various approaches for user analysis, usability goal specification, and design assessment are provided by the UE (Metzker & Offergeld, 2001). However, above all, UE emphasises user feedback, user-system interaction, and the system's efficiency in allowing users to carry out their desired activities.

HCI researchers have introduced a range of usability questionnaires, as these are considered to be among the best methods for usability assessment. Hence, mobile UIs can be effectively appraised based on a questionnaire approach. In addition, although different usability aspects can be measured with usability questionnaires, those aspects are not necessarily the same in every questionnaire (Ryu, 2005).

1.1 General Background of the Research

Any method or set of methods that are rooted in a rational theoretical foundation is classified a formal method, which is so called because it involves the use of mathematical, logical notations. Prior to development, a description of the UI is provided based on these notations, which are thus referred to as requirements. To ensure that the design functions as it should, the requirements are subjected to different proofs, since they are defined in a mathematical language. Two key functions are undertaken regarding the requirements: ensuring that the requirements provide an accurate description of the software (validation) and ensuring the precision of the functionality and the reliability of the software (verification). As previously mentioned, a formal method is adopted in this thesis with the purpose of ensuring the functionality and proper usability of the proposed software. Furthermore, a hierarchical

structure and set theory constitute the foundation of the employed formal model. Set theory has been demonstrated to be highly valuable and effective in a multitude of contexts involving modelling based on discrete membership (Hood & Wilson, 2002). The initial semantics of notations are uncomplicated and amenable to interpretation. Therefore, the thesis adopts these straightforward notations for reasons of reader clarity and understanding. After formulation, validation and verification of specifications, the following step is conversion of the abstract description into an actual implementation, which will involve using the proposed model to develop case studies.

The reason for adopting the formal method is to ensure that the software functionality fulfils the requirements and to establish how this can be accomplished most effectively. In the context of the specification, the ‘how’ is approached from an abstract perspective to ensure that it provides high-level and not low-level solutions (i.e., code application).

The feature that is shared by all the various techniques, methods, and approaches that are included under the umbrella term HCI is that they are centred on the users, namely, the individuals for whom a certain application is intended and who employ the application in their daily lives. Based on the structured activities that are represented by HCI methods, designers can not only attain a comprehensive understanding of users, but also convey their ideas to the users and improve the design process with the opinions and views that are expressed by the users. In this thesis, high-fidelity prototypes of two mobile user interfaces have been created to enable users to become acquainted with the design and obtain various kinds of feedback from them.

To summarise, the two approaches to software development that are employed in this thesis are formal methods and HCI. These two approaches have been briefly described in this chapter, and a more comprehensive discussion about them and their application in this thesis is provided in the subsequent chapter. These two approaches have been selected because they enable the issue of effective design and implementation of usable software to be addressed pragmatically and they encompass the major aspects that are prioritised at present in the process of software development, namely, system functionality and usability. From one perspective, formal methods and HCI are similar in that they both aim to develop correct software that functions according to requirements. However, the two approaches differ in that formal methods are concerned primarily with system functionality, while HCI prioritises users and their interaction with the software as a way of meeting the requirements.

1.2 Outlining the Research Problem

Several problems are discussed in this thesis in the context of software engineering and the implementation of a model-based UI design methodology:

The first problem is that simultaneous visualisation of more than one hundred elements of the UI structure design of large software systems is highly challenging due to the complexity of such systems, particularly when designers have to deal with certain design details at the same time (Šnajberk, 2013). Consequently, the design structures are typically difficult to read. Developing a standard approach for expression of these designs is not easy either, because of the significant discrepancies that can exist between elements and their attributes.

Among the principles of design that bear the greatest significance are structure and balance. It is essential for the components of a design to be structured and balanced in an appropriate way, otherwise the entirety of the design dimensions will be in disarray and their meaning will be compromised. In addition, and more significantly, the users will have trouble gaining a good understanding of an interface.

The second issue is that formal methods are still quite limited using in the industry (Moussa et al., 2002), despite being rigorous and methodical. This may be explained in terms of the following:

1- Lack of familiarity with the complex formal notations

Formal notations may appear difficult to understand to software engineers, despite being underpinned by straightforward mathematical principles.

2- Imperfect lifecycle coverage

No model or notation is capable of supporting every software development process (i.e., specification, implementation, verification, and validation).

3- Restricted usage of formal methods for UI development

The UI takes up a significant amount of the time and effort that are dedicated to application development. UI formal specification is of great significance not only for identifying mistakes and problems during the preliminary development stages but also for demonstrating the target characteristics. However, formal methods are yet to become a standard approach to UI development, even though they have been the focus of extensive research.

1.3 Motivation

The development of current software systems is no easy task because, on the one hand, the real-time and stochastic requirements of these systems need to be considered, and on the other hand, these systems are complex, of large size and frequently critical in the majority of domains (Meedeniya, 2013). Therefore, robust modelling and analytical techniques are necessary for creating such software systems (Tang et al., 2010). To facilitate the development of these systems, it might be useful to first model the system design, which could then serve as a basis for the analysis and development of the software systems. To this end, to ensure the completeness and correctness of a system model and, implicitly, the consistency of the software system itself in terms of specifications, formal methods must be employed. The abstract mechanisms for software system development that are offered by model-based software development deal with these problems to some extent (Meedeniya, 2013). However, to enable a software model to be validated and verified during the design phase, the method of formal modelling is needed.

A formal modelling approach is advantageous primarily because it enables usability results to be derived at a preliminary stage of prototype implementation, thereby making the design more cost-effective and less time-consuming. Furthermore, the approach helps the designer better understand the manner in which user task performance is supported by the design (Kieras, 2009). In addition, quantitative comparisons of usability among different designs can be undertaken.

The chosen approach is advantageous for several reasons. Unlike conventional methods for interface design and development, this approach can contribute to enhanced usability by guiding designers' work. By comparison to the space of interfaces, the space of models for enhancements is not as large, since models present a higher degree of abstractness than UIs. Furthermore, designers do not need to create a new model to convert the model of an original interface into a new interface model, as simple alterations, such as element addition/elimination and modification of hierarchical structure levels, are sufficient. The redesign process is made more time-effective and less effort-consuming, as designers' decision-making is facilitated by this approach, which provides relevant design directions for achieving improved usability. Moreover, this thesis employs case studies to show that the proposed model is applicable and to demonstrate that it is syntactically and semantically correct.

One of the main drivers in undertaking this project is the lack of work done in understanding the effect of UI design on the overall quality of software and the development process. Very little has been done to examine the effect of complicated user interfaces. This needs to be redressed in order to improve user interfaces in the future. This is particularly true of mobile interfaces. A number of design methodologies have been developed that focus on the users of software, and published studies can be found in the areas of, for example, UE and HCI. However, it is important to undertake a project that focuses specifically on mobile interfaces.

1.4 Research Questions and Hypotheses

In keeping with the background theory and the problem statement, the central research question that this thesis aims to answer is the following:

CQ. How can interactive systems be developed by using a new formal model as a design tool and how can this tool ensure the low complexity and high usability of the UI design?

Elaborating on this key research question, two supplementary research questions, alongside related hypotheses, have been formulated to aid in the design and assessment of mobile UIs:

SRQ1. Is the design rationale aided by the new model approach?

SRQ2. Compared to the conventional approach, is the new model approach more effective in aiding the development of usable UIs?

There are two hypotheses that address the first supplementary question:

H1. By comparison to other UI applications, the new UI approach affords a structural design of more limited complexity.

H2. The new UI designs (i.e., the case studies) and other UI applications do not differ significantly in terms of single- and multiple-screen designs.

There is a single hypothesis that addresses the second supplementary question:

H3. The performances of the new UI designs and existing UI applications are not very different in terms of usefulness, information quality, interface quality and overall satisfaction dimensions.

To address the above questions, this work attempts UI design based on a new mathematical model and, following implementation, tests two mobile UI prototype applications. Both design and comparative analysis are conducted in support of this approach (see sections 6.2

and 6.3), while measurement of usability and application assessment are undertaken based on data that are derived from users via a questionnaire tool. Sections 6.4, 6.5 and 6.6 provide the results regarding the responses to the questions.

1.5 Research Aim and Objectives

The main aim of this research is to develop a new model, namely FMMUID, which is based on a hierarchical structure and set theory, for coping with the problem of complexity and to introduce a method that can be applied for simple and complex UI applications. The proposed model was built by integrating the UI design specifications into a formal model, which can be classified into four main factors: (a) screens, (b) components, (c) colours, and (d) functions (Chapter 4). Jacob remarked that *“The user interface is a critical element in a software system and one is handicapped in trying to design a good user interface without a clear and precise technique for specifying such interfaces.”* (Jacob, 1983, p. 259).

To achieve this aim, the following objectives have been outlined:

- 1- Develop a novel approach to UI design (Chapter 4);
- 2- Validate the application of the new approach on case studies (iPlayCode and SC; Chapter 5);
- 3- Analysing the design structure and UI elements (for case studies and existing applications) in order to measure the complexity and validate the proposed approach (Chapter 6);
- 4- Evaluating the status of the two aforementioned UI case studies (Chapter 5) from the viewpoint of the end-users (Chapter 7).

1.6 Contributions

The main contribution of this thesis is the proposal of a novel formal model for mobile UI design. In addition, the obtained results are significant for future directions of investigation into UI design.

The contributions can be summarised as follows:

- A new model that is rooted in hierarchical structure and set theory, which is intended to facilitate the creation of usable UIs (Chapter 4).
- Improvement in the quality of the software system and greater cost-effectiveness can be achieved via the proposed model.

- The proposed model demonstrates flexibility in terms of practical implementation, regardless of designers' or developers' level of knowledge about formal methods.
- Alternative designs can be supported by the proposed novel model (FMMUID) that has been legitimated by the empirical findings. (Chapter 5).
- A hierarchy diagram was employed to graphically demonstrate the proposed model and address different dimensions of design (Chapter 4 and Chapter 5).
- Two approaches were employed to measure the complexity and usability of the model.

The novelty of the model relies on the establishment of formal rules for modelling, composition, and integration, whilst ensuring that the model was syntactically and semantically correct. The model is flexible enough to be used for UIs of various levels of complexity. This model was implemented as a prototype and assessed based on two example case studies (Chapter 6 and Chapter 7).

1.7 Research Structure

In Chapter 2, the literature that is available on the topic is reviewed, with special emphasis on system usability and identification of the most appropriate sources in relation to the research paradigm. In addition, specific aspects that were addressed by earlier studies are discussed as well.

Chapter 3 is concerned with the research design and with presentation and justification of the approaches that are adopted to address the formulated research questions.

As a key part of this thesis, Chapter 4 describes the proposed model for UI design development, including syntax and semantics.

Chapter 5 describes two case studies for evaluating the proposed model and measuring the complexity and usability of the UI. These case studies highlight the outcome of the practical implementation of the suggested model and the model's advantages.

Chapter 6 provides an analysis and discussion of the research results and concluding remarks about the major research findings.

In Chapter 7, the employed approach is assessed from both a technical and a human point of view using a range of research techniques.

Finally, Chapter 8 summarises the proposed model and the achievements of the research. In addition, potential lines of inquiry for further research are proposed, the final conclusions are formulated, and the methodology for accomplishment of the overall research aim is delineated.

Chapter 2: Literature Review: Methods of Formal Modelling of User Interface Design

2.1 Introduction

The implementation of formal techniques in the design of UIs and interactive systems has been the focus of extensive study. There are two major aspects of this topic, namely, (a) integration of user interfaces, user concerns and UI-system interactivity into formal methods and (b) attempts of design process formalisation to enable UI designers and human-computer interaction practitioners to ensure the accuracy and functionality of different design components.

In this chapter, the integration of formal methods and user interface design (UID) is outlined. To this end, only those studies and examples that deal with the approaches of concern to this work are addressed rather than the entire literature on the topic.

The remainder of the chapter is structured as follows: the interactions between humans and computers are discussed in section 2.2; the user interface is the focus of section 2.3; the instruments of UI development are presented in section 2.4; the complexity of the UI and existing methods for dealing with that complexity are covered in section 2.5; usability and related concepts are addressed in section 2.6; the existing literature is systematically reviewed, and the research deficiencies and matters that still require attention, according to the findings of the literature review, are considered in sections 2.72.5 to 2.112.9; and the concluding remarks are provided in section 2.122.10.

2.2 Human-Computer Interaction

This part presents a succinct overview of HCI and discusses its implications for software. The manner in which humans employ computer technology and strategies for enhancing this usage are the main concerns of the field of HCI (De Oliveira, 2015). Several studies observed that convoluted processes of data entry, ambiguous error messages, rigid error management, and unclear cluttered screen sequences all cause users of advanced hardware machines significant irritation and dismay (Baecker, 2014; Bertino, 1985; Mital & Pennathur, 2004). Systems that behave in unintelligible and perplexing ways cause apprehension and frustration, especially among inexperienced users (Bertino, 1985).

Overcoming such issues has been a priority in a great proportion of recently conducted studies, which have drawn input not just from computer science but also from other disciplines; this has enhanced the complexity and richness of the HCI field. Contributions of particular significance have been made by the disciplines of software engineering and ergonomics (Abowd et al., 1992; Bastien & Scapin, 1993; Bevan, 2001; Coutaz & Calvary, 2012; Imaz & Benyon, 2007; Long, 1989; Vanderdonckt, 1994), both of which focus on examination of requirements, incremental and iterative design, and quality guarantee (Coutaz & Calvary, 2012). Indeed, software engineering is essential in HCI for aiding in the design and development of relevant, usable systems (Dix, 2016; Göransson et al., 2004), while ergonomics contributes by ensuring that the human-computer interfaces are ergonomic, relevant, and usable by issuing principles of design and/or assessment (Vanderdonckt, 1994). Successful HCI is the overall goal of all efforts to enhance human-computer interfaces.

It was only during the 1980s that HCI began to be accorded importance, when computer systems had already been in use in commercial and industrial applications for a long time. This may be because, in preceding times, the users acted as the computer system programmers and designers. From the end of the 1970s onwards, however, there has been a proliferation in the number of users who are not involved in system programming or design, owing to the launch of personal computing (Booth, 2014). Comprised of both personal software, such as text editors, spreadsheets, and computer games, and personal computer platforms, such as operating systems, programming languages, and hardware, personal computing simplified computer usage for all but also made it apparent to advanced computer users that computer systems had considerable limitations (Carroll, 2013). As a result of such transformations, interactive systems started to attract significant interest.

2.3 User interface

Communication between an interactive computer system and a user is facilitated by the UI of the system. All aspects of the system that the user can see are incorporated into the UI design (Jacob, 2003). The UI manages both the display output and the user input (Myers, 1995). The UI can also be understood as the portion of a computer system that a user can control, comprehend, or interact with via different senses (i.e., sight, hearing, touch, speech) (Galitz, 2007). Basically, the UI is the software system element that enables a user to interact (i.e., interface) with an application (Kennard, 2011).

It is quite challenging to develop a robust UI that demonstrates usability and responsiveness as well as an appealing design. No standard method for ensuring that a UI will be successful exists, even though there are principles of usability design (Nielsen, 1994a). Ironically, the complexity of UI development increases as the ease of use and ease of understanding increase because extra features are added (e.g., run-time validation or help messages) to make the interaction experience more enjoyable and agreeable for the user (Cerny et al., 2012; Myers, 1994). In addition, the lengthy process that is involved further exacerbates the difficulty of UI development. According to some studies, almost half of the code development time of an application is taken up by the UI (Cerny et al., 2012; Kennard & Steele, 2008; Myers & Rosson, 1992; Schlungbaum, 1996). For all these reasons, acceleration and simplification of UI development have become a priority for developers.

2.4 Instruments for UI Development

Ample efforts have been made by researchers to devise instruments that facilitate the development of UIs, accelerate the process and ensure its cost-effectiveness without compromising UI quality (Myers, 1995).

2.4.1 Window Managers

The output of the display screen can be created and the user's input can be recognised with the basic programming model that is supplied by Window Managers (Myers et al., 2000; Myers, 1995). Processes are separated by the Windowing System into a series of clear-cut screen areas, which are called windows. To develop interfaces at this level, all interface elements must be constructed from the very beginning. This could cause different irregularities throughout the interface, whilst also showing how slow-paced and cumbersome the process is (Rosenthal, 1988). Therefore, as it does not prevent irregularities and can slow the UI development process, such an approach is not a viable solution.

2.4.2 Toolkits

Widget libraries are supplied by toolkits (Myers et al. 2000; Myers, 1995) and implement a framework on the basis of which user interactions can be manipulated. They necessitate sole instantiation, as the UI elements are pre-established. This simplifies and speeds up the UI development process. On the downside, the UI is limited to the widget library owing to its dependence on the toolkit framework. There are two major types of UI development

instruments: direct graphical specification instruments and model-based instruments (Myers et al., 2000; Myers, 1992, 1995).

2.4.2.1 Direct Graphical Specification Instruments

This category of instruments is comprised of prototype instruments, data visualisation instruments, and editors for application-specific graphics, as well as interface builders such as Netbeans GUI builder (Oracle, 2009) and ‘what you see is what you get’ (WYSIWYG) (Kalnins et al., 2002). Such instruments enable partial or complete development of the UI by dragging objects with a pointing device to position them directly on the canvas. However, automation is largely impeded due to the lengthy and iterative development process. Additionally, no guidance for the development of effective and robust UIs is provided by these instruments.

2.4.2.2 Model-based Tools

UI production is achieved by model-based tools with the help of system models of high-level specification. Their main purpose is to alleviate a common problem that affects other tools, namely, the high cost of UI development, by automating the development process. Model-based tools, such as USIXML (Limbourg et al., 2004) and TERESA (Berti et al., 2004), are advantageous because they can use different numbers and models, achieve different degrees of automation, and generate various kinds of UIs. Thus, they are considered a more viable option than other types of tools for rapidly creating UIs of high quality (Meixner et al., 2011). Additional information about model-based tools is provided in section 2.10 and section 2.12.

2.5 User Interface Complexity

Complexity, which is synonymous with ‘intricacy’ and ‘convolution’ Anneberg and Singh (1993), has been described as an attribute of the connections that exist between the elements of an object (Zuse, 1993), and usually refers to the multiplicity of those connections or interactions. Furthermore, a high number of components, high dimensionality, and a wide range of possibilities have been identified as the characteristics that are demonstrated by all complex systems (Morowitz, 1995).

According to the IEEE Standard Glossary (std. 610.12), complexity reflects how challenging it is to understand and assess the design or implementation of a system or element (Radatz et al., 1990). A different study also defined complexity in terms of difficulty of understanding, noting that program comprehension takes up nearly half of the time of a software maintainer

(Alford, 1994). These definitions indicate a key aspect that is associated with complexity, namely, understandability. Another study argued that the cognitive complexity of a software is determined by the influence of the software features on the amount of resources that are required for the performance of a certain task (Henderson-Sellers, 1995). Similarly, Basili (1980) described software complexity as the amount of resources that are taken up by a system during interaction with a software program to carry out a specific task. Software complexity was considered by Brooks (1993) to be not an accidental characteristic but an essential one. The difference is that the type of issue and the skill level that is required for dealing with it determine the essential complexity, whereas the accidental complexity is the outcome of inappropriate endeavours to address the issue and is best understood as a complication. An issue is endowed with accidental complexity when an incorrect design is applied or when an unsuitable data structure is chosen. Complexity was explained by Coskun and Grabowski (2005) as a measure of difficulty, especially in terms of comprehension of the multiplicity of connections or interactions between at least two elements of an object. It is challenging to define and measure complexity because more than one factor influences understandability and this measure is subjective. Common definitions of complexity are frequently associated with series of interlinked parts, which are known as “systems”. Different definitions prioritise different aspects when referring to complexity, such as how the system behaves or its internal structure. Simon (1969) maintained that the system needs to be conceived as a tree-structured hierarchy to fully understand its complexity.

No unanimous agreement has been reached regarding the precise meaning of complexity, despite the numerous persuasive definitions that have been put forth (Edmonds, 1995). The multidimensionality of software is the reason why its complexity can be explained in more than one way. Therefore, software complexity is a matter that divides researchers and users.

2.5.1 Measures of UI Complexity

The complexity of a user interface, the evaluation of the degrees of complexity of a UI, and the implications of that complexity for the users of the interface have been thoroughly investigated in the literature. Over the years, many metrics have been proposed by researchers for the formal evaluation and prediction of software complexity. The following section outlines many of the metrics used for measuring the scale of a piece of software’s complexity.

The first software complexity metrics were designed and applied to software systems in the mid-seventies. The metrics that developers used included lines of code (LOC), Halstead's complexity metric, McCabe's cyclomatic complexity metric, and Kafura and Henry's fan-in/fan-out method (Chhillar & Bhasin, 2011; Kumari & Upadhyaya, 2011; McCabe, 1976; Yu & Zhou, 2010).

In addition, several metrics were proposed by Salman (2006), for the measurement of a system's complexity via a stringent focus on the structural complexity of the software. He considered a system's connectors, interfaces, composition trees and components as the prime indicators of structural complexity in a component-based system, even though these metrics are considered basic. Instead of considering the complexity of individual interfaces, the metrics focus instead only on the total number of the features described above.

Proposed by Gill and Grover (2004), the Component Interface Complexity Metric (or CICM) determines complexity via interface signatures, interface constraints, and an interface's packaging for different use contexts.

Significantly, however, CICM lacks an empirical method for evaluating each proposed metric.

Another paper by Youxin et al. (2009) proposed an architecture that used Component-based Software Development (CBSD), and then explained the development process according to the architecture. CBSD proposes that multi-layer architectures are more effective at solving problems in software and improving the quality of the software, while also making development easier.

The minimisation of software complexity whilst maintaining quality is vital. The analysis of software complexity via complexity metrics can result in the reduction of software design time and complexity, along with the amounts of testing and maintenance needed. According to CBSD, the size of a piece of software and its interfaces for each component are the two primary parameters that govern software complexity. In other words, the more methods that belong to a class, the harder it is for a developer to understand the software due to its complexity and interconnectedness.

However, interaction complexity is often considered more important. Therefore, the average number of interfaces per component should not exceed five when used in a CBSD, as greater numbers of interfaces result in increased levels of complexity and unreliability (Kumari &

Upadhyaya, 2011). This methodology, however, requires further research and empirical evidence.

Gerhardt-Powals et al. (1995) studied complexity in relation to submarine displays, while metrics associated with real-time UI complexity were explored by (Andriole & Adelman, 1995; Chignell, 1990). According to Kang and Seong (1998), operation, transition and screen are the three measures on the basis of which UI complexity can be assessed.

Ross and Burnett (2001) also investigated UI complexity in their study on human-machine interfaces in vehicle navigation systems, while UI complexity was addressed by (Martelli et al., 2003) in the context of medical informatics. Metrics for UI complexity assessment were proposed in both studies. Similarly, the metrics that were devised by Xing (2004) were underpinned by three complexity-related factors: numeric size, element diversity, and inter-element connections. In a different study, Cataldo et al. (2010) employed data from two large-scale systems that were developed by two different software organisations to analyse the extent to which the predisposition to failure of source code files was influenced by UI complexity. Moreover, Alemerien and Magel (2014) proposed that UI complexity could be assessed on the basis of its structure by using the GUIEvaluator tool. Another study by Ašeriškis et al. (2017) have used graph metrics (number of nodes and links) to measure the complexity.

2.5.2 Summary

Despite the lack of a general definition, it is important to be aware of software complexity, as it is indicative of a range of aspects, including system development, assessment, maintenance, rate of malfunction, reliability, and elements that are most likely to generate errors. Complex software poses challenges not only in terms of a greater rate of error, but also regarding development, assessment, debugging, and maintenance. Furthermore, as highlighted by most of the existing definitions, the degree of complexity of a system determines how understandable that system is.

UI complexity refers to the extent to which users perceive the information displays of the system to be easy to use and the user screen to be understandable. The usability of the output of the system depends on decision support and explanation complexity. The results that are generated by the system must be comprehensible by the users, and the advice that is provided must be logical and easy to convert into cognitive thinking and decision-making.

There are two aspects that are related to the understanding of the implications of UI complexity: the complexity of the display and the actual UI, and to the decision support that is offered. To determine the ideal approach for designing and deploying UIs, the implications of both these aspects of complexity must be clearly comprehended (Coskun & Grabowski, 2005).

This section has succinctly reviewed UI complexity, which is measured in terms of a range of aspects, including structure, layout, text quantity and graphics in a display, number of nodes and links, code, and colour. To the best of the author's knowledge, investigation of UI elements and design for measuring UI complexity has not been attempted by any other study, nor has a design solution for complex interfaces been formulated. The metric that is proposed by this thesis is based only on the component interface specification. The metric calculates an interface complexity through its a component's complexity, thereby providing an excellent indication of a component's reusability. Thus, this metric can be used for the selection of a simpler and more usable UI. Lastly, the metric is calculated manually. Therefore, these will be the focus of the following chapters of the thesis.

2.6 Usability of User Interface

A number of approaches have been used in the past few years to discover problems with mobile interfaces. For example, it has been proposed that a structured interview be used to evaluate user experience in using mobile phones (Park et al., 2013). An application called MastroCARONTE was developed to deliver local information direct to cars while traveling. Researchers distributed a set of questions to users of the software to ascertain user response (Gena & Torre, 2004).

The website of the library at Punjab University was the subject of a research project (Iqbal & Warraich, 2016)) to ascertain users' views of the website in five categories: affect; efficiency; helpfulness; control and learnability. Existing studies were evaluated and their findings in both theory and practice applied. A group of 300 users were given a carefully chosen set of questions and the resulting responses processed by SPSS. User responses were most positive in the categories of affect and efficiency, which indicates that users feel more benefit from these 'soft' categories than from the other, more practical, categories.

An important consideration in mobile interface design is how quickly and easily users can 'learn' the application, and analysis of this process is a valuable and frequently adopted

approach. This is often called a heuristic method and was adopted by Ji et al. (2006) who evaluated users ability to learn applications against a set of fixed criteria which could then be used by interface designers. It has been suggested that this approach can be generalised to be used in a variety of circumstances (Biel et al., 2010). The work of these researchers suggests a hands-on approach to interface evaluation.

The popular Google Android interface was studied from the point of view of ease of use in 2013. Under laboratory conditions, six testers undertook a ‘Cognitive Walkthrough’ of three Android apps – GOSMS Pro, Skype and WhatsApp. These were chosen by taking the three most commonly used Android apps from a review of five top apps in a popular computer magazine – in this case PC Magazine (Jadhav, Bhutkar, & Mehta, 2013).

Automated response measuring systems (Lift; Bobby) were used by Alexander and Baravalle (2012) to look at use profiles of three gov.uk websites and establish whether the criteria of ‘usability’ and ‘accessibility’ were interconnected. The ‘heuristic’ and ‘walkthrough’ methods were applied. This revealed that the sites, while following WCAG accessibility guidelines, scored poorly for usability.

2.7 Software Modelling

Design models are essential for successfully developing a software system. As a widely embraced engineering method, modelling underpins every procedure that needs to be implemented to deploy effective software (Booch et al., 2005) and its purpose is to mathematically represent the behaviour of a system or object (Giorgi et al., 2004). A model can be understood as a simplified version of a system that is created to achieve a particular objective and address existing issues without using the real system (Bézivin & Gerbé, 2001). In other words, a model is an abstract representation of an actual system that enables engineers to concentrate on the key features and not waste time on superfluous details (Brown, 2004). In addition, as observed by Kühne (2006), a model facilitates the formulation of estimates or the drawing of inferences. Other definitions have interpreted system modelling as a characterisation or specification of that system and its context to achieve a particular aim (Soley, 2000) or a series of statements about the system that is under consideration (Seidewitz, 2003).

Various different functions can be fulfilled by models, including (Booch et al., 2005): conveyance of the intended structural and behavioural features of a system; visualisation and

management of the system's architecture; promotion of a more comprehensive understanding of the system; visualisation of the end-product for users; formulation of particular theory-based projections that can then be assessed with data; and determination of how the different system elements are correlated.

Given that the human capacity for comprehending complexity is limited, modelling provides a highly useful tool for creating and understanding interactive systems as well as for breaking down components of great complexity into smaller units that are easier to deal with (Navarre et al., 2005).

The model-based approach is primarily geared towards uncovering relevant abstractions that are representative of the key features that need to be taken into account in the development and design of an interactive system (Marucci et al., 2003).

In the context of this approach, UI design can be understood as the process through which UI models are developed and perfected (Da Silva, 2000). Thus, the purpose of model-based design is to determine correlations among different models (Limbourg & Vanderdonckt, 2005).

UI development based on modelling is advantageous because decisions about design can be made by actually creating the desired task model, thereby enabling designers to explore interactive software applications from a semantic perspective instead of having to deal with the implementation process immediately, and because it affords a better understanding of the system for maintenance purposes as process reconstruction is facilitated by the systematic and iterative development approach (Sinnig, 2004).

As previously mentioned, a software model is an abstract version of an actual system that has already been developed or will be developed. Because it is a simplified representation of a real system, the model enables software designers and/or developers to better assess the system. In general, the modelling of the requirements of users regarding system functionality is undertaken in the engineering stage of software development and its formality and rigorousness vary. Visual and formal models, which are respectively based on diagrams (Rumbaugh et al., 2004) and formal techniques coupled with mathematical notions (Fitzgerald & Larsen, 2009; Kohlas et al., 2006; Schoeller et al., 2006), are the two classes of models that are used most often at the moment. Subsequently, an analysis model is generated by software engineers by refining the model either manually or electronically. The analysis model outlines the problem domain. This is followed by the creation of a design model that

proposes a solution for the problem that was modelled earlier. The generated solution can then be coded and tested on the basis of the design model (Pressman, 2005).

2.8 Development of a UI Based on Modelling

During the 1980s, model-based UI development (MBUID) was first introduced. Since then, there have been four generations of MBUID approaches. Spanning the period 1990-1996, the first-generation MBUID approaches were concerned with the automatic creation of UIs. The second-generation approaches were popular during 1995-2000 and facilitated specification, production and execution of UIs. The third-generation MBUID approaches were in use during 2000-2004 and attempted to achieve UI development for different interactive platforms. Last, the fourth-generation approaches, which have been in use since 2004, target the creation of UIs that demonstrate context sensitivity (Meixner et al., 2011).

2.8.1 First- and Second-Generation MBUID Approaches

The first-generation approaches were mainly geared towards facilitating UI creation and maintenance by enhancing UI development or generating better methods for achieving this process.

To aid the cost-effective development of good-quality UIs, a User Interface Management System (UIMS) that is known as COUSIN (Hayes et al., 1985) is devised to introduce a degree of abstraction in the manner in which the input/output events of the UI dialog are ordered. A ITS tool-based architecture with four layers (Wiecha et al., 1990) was proposed as an option for UI representation with multiple layers by breaking down the implementation, content, presentation and interaction of the UI into the action layer, dialog layer, style-rule layer, and style-program layer, respectively. The main advantage of this approach was that it enabled the presentation of the same UI with more than one style. Although the issue of improving methods for UI development has not yet been solved, the rapidity with which UI development approach have evolved caused early UIMSs to fail due to the problem of moving targets. This highlights the challenge of ensuring that tools can keep up with the speed at which technology develops (Myers et al., 2000).

The leverage of MBUID for UI creation was the goal of a different set of approaches. For example, GUIDE (Foley et al., 1991) and HUMANOID (Szekely et al., 1992) sought to automatically generate UIs to enable designers to consider various design possibilities prior to finalising the UI. Meanwhile, the methodology with the supporting environment offered by

TADEUS (Elwert & Schlunbaum, 1995) was intended to facilitate graphical UI creation based on a system model. A similar approach for UI creation from data models (entity relationship diagrams) was provided by GENIUS (Janssen et al., 1993) with its tool-supported method, in which UI dynamics were visually represented with the dialogue net model that was underpinned by petri nets. JANUS (Balzert et al., 1996) and FUSE (Lonczewski & Schreiber, 1996) were also systems that supported UI creation. JANUS had the additional function of enabling the production of code that established a connection between the UI and the data.

A straightforward rule-based approach was applied by most early MBUID approaches that were focused on automatic UI creation. TRIDENT (Vanderdonckt & Bodart, 1993), which provided tools for automatic creation of UIs for interactive business applications, as well as a generic architecture model that was relevant for those kinds of applications (Bodart et al., 1995), was one method that did not adopt this approach. To create UIs, TRIDENT took into account additional data, such as ergonomic rules, which were represented on the basis of a complex hierarchy. However, because they were often numerous, the application of such rules was cumbersome, despite offering a more advanced method for UI creation (Vanderdonckt & Bodart, 1996).

Enhancement of model-based UI representation was the goal of other systems. For instance, the design environment ADEPT (Markopoulos et al., 1992) did not simply generate a rapid tool for prototype production, but integrated the modelling theory. The presentation model was the main concern of MASTERMIND, which was a UI development environment that served as an accessory to HUMANOID and GUIDE (Szekely et al., 1996). MECANO employed the modelling language MIMIC and provided the MIM interface model (Puerta, 1996).

Early systems such as COUSIN, GENIUS, HUMANOID, and GUIDE were deficient in that they did not describe the UI in detail; instead, they represented it in various ways, such as with application code and ER diagrams in the cases of HUMANOID and GENIUS, respectively. However, second-generation systems, including ADEPT and MASTERMIND, did provide in-depth UI description. However, UIs only began to be represented at the highest abstraction level towards the end of the second generation, with objects such as the ConcurTaskTrees (CTT) (Paternò et al., 1997). Meanwhile, systems such as MOBI-D (Puerta & Eisenstein, 1998) considered novel methods for task model mapping to UI models of lower

level. Furthermore, the need to define UI specifications without dependence on technology to enable the creation of technology-specific UIs prompted the introduction of languages that do not depend on the technology, such as the User Interface Mark-up Language (UIML) (Abrams et al., 1999).

This generation of MBUID systems adopted a basic approach to multi-target UI development. Tool support for interactive system development was provided by AME by using object-based analysis models to create UIs, which were then customised according to the requirements of individual users (Märtn, 1996). Meanwhile, facilitation of the development of different UI features (e.g. display size, resolution, and colour depth) according to usage requirements was made possible by certain earlier systems such as ITS. However, most systems, including GENIUS and COUSIN, did not afford as much priority to adjustment as they did to UI consistency between various applications. User- and environment-based UI adjustment was supported by some later systems, such as AME, which employed standardised object categories; however, this support was focused less on adaptive behaviour than on manual development. Hence, the main limitation of first- and second-generation MBUID systems is that they employed the model-based approach only to create UIs and not for development of adaptive behaviour for supporting multi-context UIs.

2.8.2 Third- and Fourth-Generation MBUID Approaches

The third and fourth generations saw the introduction of domain-specific solutions, such as Teallach, which generated object databases by adopting the MBUID approach (Griffiths et al., 2001). A reference framework that employed multiple abstraction levels to assist MBUID and the introduction of novel UI Description Languages (UIDLs) constituted the main advances that were made in these generations.

Two principles underpinned the unified UI reference framework called CAMELEON (Calvary et al., 2003): a model-based approach and coverage of the design stage as well as the run-time stage of UIs with multiple targets (Fonseca, 2010). Within these generations of MBUID systems, CAMELEON constituted a ground-breaking innovation by using a model-based approach to offer abstraction assistance for UI development. A similar tool was MDE, which provided extra abstraction levels to facilitate multi-context UI development, unlike standard methods of UI development that generated solely a concrete level, such as buttons and text boxes.

CAMELEON can achieve UI representation at numerous abstraction levels. The highest abstraction level at which UI features are represented as tasks is the task model. The ConcurTaskTrees (Paterno, 2012) notation can be used to represent such a model, as it mediates connections between tasks and temporal operators. Representation of the domain model that is related to the discourse universe of an application can be accomplished with diagrams of the Unified Modelling Language (UML) category. In the case of MDA, this abstraction level is associated with the Computation Independent Model (CIM). A UI that is independent of all modalities (e.g., graphics, voice, gesture, etc.) is called an Abstract User Interface (AUI) model. UIDLs such as TERESA XML (Berti et al., 2004), UsiXML (Limbourg et al., 2004) and MARIA (Paterno et al., 2009) (fourth generation) can all enable representation of the AUI model. In MDA, this model is associated with the Platform-Independent Model (PIM). The abstraction level of the concrete user interface model depends on the modality, meaning that it can use graphical widgets (e.g., buttons, labels) in UI representation. TERESA XML, UIML (Abrams et al., 1999), and XIIML (Puerta & Eisenstein, 2002) are among the UIDLs that can be used to represent concrete user interfaces. Presentation technology, such as HTML, Windows Forms, WPF and Swing, are used to develop the final UI, which represents the real UI.

Multipath development (Limbourg et al., 2004), with an emphasis on mobile devices, and integration with available web services are the two major objectives of MBUID systems of the fourth generation. To a great extent, the approach that this study puts forward is orthogonal to these objectives. Using structural models as a foundation, the aim of this approach is to develop an effective strategy for interface development. The approach that was adopted by Macik and colleagues (2014) was to achieve streaming of platform-specific UIs based on integration with UI protocol and to use metrics for applications of automated element distribution in users' screens. This approach is disadvantageous because it does not pay significant attention to client interaction and the benefits of concern division are diminished from the clients' viewpoint because UI generation occurs at the side of the server. In a different study, a process of development for UIs with context sensitivity was proposed. The design component of this process was the central concern and tool support for model creation, amendment, and visualisation was considered as well. The approach had the drawback that it was not subjected a usability test to evaluate the usability of the design tool (Clerckx et al., 2005).

2.8.3 Summary

The abovementioned approaches present a range of issues:

1- MBUID is still not widely adopted in everyday industrial software development, even though it has been studied for over three decades.

2- Standardisation: The necessity of standard MBUID notations was first realised in 1999. Notation standardisation would make it easier to employ a common series of constructors for the characterisation of various UI models. Such constructors would enable UI models and their MBUIDs to be compared and reused. For example, more than one notation underpins current UI models, thereby making it harder to reuse them. However, the achievement of MBUID scalability for actual applications depends significantly on UI model reuse (Da Silva, 2000). Furthermore, although a wide range of UIDLs have been created, they have not been adopted extensively in the development of actual industrial software.

3- The difficulty involved in model development is the primary limitation of the model-based approach. Specification of a rich model is challenging because of its complexity, which stems from the characterisation of every interface feature (Puerta & Szekeley, 1994). Hence, it is critical that useful and relevant tool support is provided to the various interested parties, including programmers, UI designers, and interaction designers.

4- With respect to transformations, MBUIDs must be extensible. Standardised notations would aid the formulation and sharing of more effective transformations for various target platforms. Moreover, UIs that are developed through automatic generation are not very usable. Hence, it is inappropriate to employ a completely automatic transformation approach. This has been an issue since the introduction of MBUID (Calvary & Pinna, 2008; Myers, 1995). Although developers can benefit from transformations in the performance of basic tasks, they can make manual amendments to the developed UI (e.g., tweaks, beautification, etc.). However, restoration of other UI draft designs causes the loss of the manual amendments, which means that, to guarantee round-trip engineering, integration of the manual amendments into the models is essential. Apart from manual refinement of the developed UI, another viable strategy for making the UI more usable is to incorporate HCI patterns into the transformation processes. In addition, to reinforce the manual amendments by developers and designers, it would be useful to incorporate formalised standards and

guidelines into MBUID (Meskens et al., 2011). Moreover, ergonomic criteria should also be integrated, as their modelling is a persistent problem for HCI (Calvary & Pinna, 2008).

5- Smaller UIs or UI fragments that are produced via MBUID processes are unknown outside the research community. For the industry to embrace MBUID, UI models, languages and the related MBUIDE must be developed and adjusted accordingly. Large-scale case studies and applications in the real world could be of significant relevance to MBUID.

2.9 Formal Specification Languages

To justify a certain rationale regarding software models or specifications pre-implementation, formal methods are employed. In fact, software models and specifications are developed using formal language and notations, signifying that their clarity of meaning is ensured by well-defined syntax and semantics. Furthermore, the models' inherent logic makes it possible to use language manipulation for the purpose of performance of various tasks. Formal methods are applied to ensure that the developed software will demonstrate adequate and reliable behaviour under various conditions.

A variety of languages and notations are formal in nature, and, although they have the same objective, they can be employed in distinct manners. These include mathematically based languages such as Set Theory (Jech, 2013), Z (Iso, 2002), B (Abrial & Abrial, 2005), VDM (Andrews et al., 1996) and XML, which create independent and abstract views of any user interface application (Lepreux et al., 2006; Puerta & Eisenstein, 2002), while languages such as MARIA and CAP3 use concrete models of dialog flows (Paterno et al., 2009; Van den Bergh et al., 2011). In addition, for the description of executable user interface models FILL (Formal Interaction Logic Language) was presented by (Weyers, 2017). ICOs, meanwhile, focus on user interface behaviour (Navarre et al., 2009), and OMG standard IFML (Interaction Flow Modelling Language) is used in a support role for an application's front-end. IFML provides an expressive behavioural model (Brambilla & Fraternali, 2014), although it may be model-based, as in the cases of discrete-event systems (Cassandras & Lafortune, 1999), interactors (Duke & Harrison, 1993) and model-based testing (Utting & Legard, 2010); proof of correctness (Woodcock & Davies, 1996), refinement (Henson & Reeves, 2000; Wirth, 2002), etc., are also used. This list is not exhaustive, and new formal methods and notations continue to be introduced. However, they all have a common aim: describing software prior to its implementation to ease the development and maintenance processes.

Mathematical logic and set theory form the foundation of specification languages such as Z (Iso, 2002), OCL (Warmer & Kleppe, 1998) and VDM (Sharma & Singh, 2013) as well as for the mathematical formalism adopted in this thesis.

The formal representation of set theory semantics has been the focus of extensive research efforts in pursuit of validating and verifying the consistency of design models (Abrial et al., 2010; Bowen & Reeves, 2008a; Gajos & Weld, 2004; Goh & Case, 2016; Maalem & Zarour, 2016; Muji, 2015; Piroi, 2004; Riza et al., 2015; Takahara & Liu, 2006; Tr  tteberg, 2002). In relation to the definition of the formal representation of FMMUID (Chapter 4), this work uses simple set theory notations, as they are suitable for illustrating every FMMUID development.

2.10 Formal Models

Formal model has been defined in many different ways. B  zivin and Gerb   (2001) defined that a model simplifies a system built with an overall goal in mind. The model should answer questions about the system itself. Brown (2004) reported that models of abstract physical systems allow reasoning about the system without extraneous details. Another study (K  hne, 2006) defined that a model as an abstraction of a system which allows predictions or inferences about the system to be made. Seidewitz (2003) defined a model as a set of statements about a studied system (SUS). Meanwhile, Selic (2003) stated that engineering models aim to reduce risk through increasing understanding of a complex problem and its solutions before expensive implementations are undertaken.

A set of well-defined methods grounded in mathematics, formal models enable system design verification through the theoretical support they offer. To comprehensively define a formal modelling language, the syntax and semantics that make the language readable and expressive must be described as well. Due to their mathematical basis, formal methods are generally employed for the description and development of systems demonstrating efficiency, reliability and safety (Heymann & Degani, 2007).

Because they can improve clarity, overcome design errors and hence prevent system failures, formal methods are increasingly accepted as a key element of the design of software systems that demonstrate reliability (Bowles & Bordbar, 2007; Cimatti et al., 2011a, 2011b; Hillston & Kloul, 2006; Jensen et al., 2007; Mosbahi et al., 2011; Moschoyiannis et al., 2005; Ribeiro et al., 2005; Yang et al., 2005). These models are especially useful in mathematically validating system attributes such as performance, reachability, and correctness (Baier et al.,

2008; Bowles & Kloul, 2010; Gilmore et al., 2003; Grumberg & Long, 1994; Hillston & Kloul, 2006; Hinton et al., 2006; Katoen, 2008; Kwiatkowska et al., 2007). Furthermore, the tangibility of software designs is enhanced by formal models because use of the models can ensure that those designs are thoroughly validated and verified (Cabot et al., 2008; Jensen & Kristensen, 2009; Jensen et al., 2007; Rafe et al., 2009; Silva & Santos, 2004). Validation and verification respectively ensure that the appropriate system is specified by the design and that the specification is met by the final system.

2.10.1 Development of Interface Models

The issue can be approached somewhat differently by developing formal models of interfaces as entities that are independent of system specification. The purpose is to apply a formal treatment to the interface and to its interactive features by developing a model of them, thus securing the advantages of formal methods based on proof performance to identify favourable attributes and correctness.

Originally proposed by Dix and Runciman (1985) and then amended, the PIE model was one of the first manifestations of this approach. Interactive systems are characterised by the PIE model in terms of abstractions of programmes (P) that comprise a series of commands and input effects (E) with an interpretation function (I) between them. Interface properties such as observability, reachability, and undo can be analysed based on these three elements. The interface abstraction generated by the model is independent both of the functionality of the underpinning system and of a reliable comprehension of UI visual components and system users (other than the pre-established inputs). Thus, the interface and application logic are kept separate in the PIE model. The model enables the interface to be treated more formally, but it does not ensure that the interface will be compatible with the other system components, nor does it provide a means of creating a correlation between the model and a system specification.

To ascertain whether a UI is an amended version of a different UI based on comparison of the number of functionalities provided by the two UIs in question, a descriptive formal approach was proposed by Bowen and Reeves, (2008b). The amendment addressed by the authors was very similar to trace refinement. They maintained that they were concerned with verifying the accuracy of the requirements in relation to the system of higher refinement, but they largely disregarded the impact of the refinement levels on the requirements. Moreover, they failed to address distinct correlation levels and overlooked the appearance of the UI.

The formal description of the concept of a Distributed User Interface (DUI) was undertaken by López-Espin et al. (2011) based on a novel notation that the authors stated was useful in analysing the key DUI attributes (e.g., decomposability, portability, simultaneity and continuity). However, the study had a major limitation in that the DUI was not assessed through a usability test.

UI description was undertaken by (Thimbleby, 2004; Thimbleby et al., 2001) using a different approach based on matrix algebra and Markov models. In its focus on interface usability, the study ignored UI design and appearance and addressed solely the potential for interaction with regard to the likely underpinning state modifications and related possibilities or probabilities regarding the Markov models.

A less abstract approach has been used in other models to correlate more clearly the visual elements, models, and behaviours of the UI. For instance, the component-based model developed by Bumbulis et al. (1995) used a design process based on an iterative approach and took into account prototype models and feedback from the users. Both a prototype to be tested by users and a model for formal evaluation could be produced with the generated UI specification, thus integrating formality and the UCD approach. However, minimal level description, nearly the same as code, was provided by the proposed language (IL). Instead of visual designs, interface elements represented the foundation for the models, and they were generated based on direct translation from IL to the dynamic interface language Tcl/Tk (Xchange, 2005).

Model verification can also be undertaken on the basis of interface models. Aware of the fact that interface designers who lacked knowledge of formal interface models did not benefit much from such models, Loer and Harrison (2000, 2002 & 2006) developed a model verification framework (IFADIS) that was intended to overcome this difficulty for usability specialists. It was suggested that model verification enabled systems to be examined in terms of how reliable and usable they were. The authors recognised that their work remained limited in terms of facilitating model verification for usability engineers, indicating that the toolset was still troublesome for usability engineers despite its usefulness to the wider community of system engineers who lacked knowledge of model verification. Another limitation of the study is that it did not consider interface visual elements and that it dealt with user concerns not from the perspective of prototypes but from the perspective of interaction requirements and opportunities.

The development of UI abstractions for purposes of portability and plasticity is one of the latest trends in UI modelling. The development of software compatible with more than one platform (e.g., desktop machine, PDA, mobile phone, embedded device, etc.) and operating system is almost standard practice. This prompted the need to create UIs that are adaptable to various platforms (portability) and a need for this adaptation to be contextual (e.g., according to existing hardware and user location) (plasticity). UI modelling for portability and plasticity has been approached by various research groups based on languages resembling XML. For instance, the XIML project (Puerta & Eisenstein, 2002) sought to achieve description of interaction data and hence create an abstract UI model using an XML-based language.

The TERESA tool was created to supply a complete semi-automatic environment to enable several transformations that would benefit design creation and analysis at various levels of abstraction, including the task level, with the purpose of producing a concrete UI for a particular platform type (Mori et al., 2004; Mori et al., 2003). However, the work failed to address approaches for simplifying the creation of more than one version. The employed procedure involved the automatic creation of various concrete UIs on the basis of an XML-abstract UI model developed through an XML task model with ConcurTaskTrees (Paterno, 2012). The visual elements of the design and the formal model were closely correlated due to the shift from an abstract model to a concrete UI. At the same time, to enable distribution of different sub-surfaces among different devices, Peñalver and colleagues (2012) put forth a novel UI definition underpinned by the W3C XML Schema. To this end, a formal notation followed by an abstract user interface (AUI) model was used for DUI definition. Considering the restrictions outlined in the schema, the next step was the creation of a new XML instance (i.e., a concrete DUI) using an XML instance generator algorithm in certain UIDLs. The AUI model provided the target hierarchy related to all UIs and ensured consistency with the user task target by helping distribute UI components over various devices. On the downside, platform-specific native code (Arthur & Olsen Jr, 2011) is essential for the application of such solutions to software; otherwise, per-element mark-up tags are needed for UI division classification (Mori et al., 2003; Peñalver et al., 2012). As a result, the tools are cumbersome for use by developers and take a long time to implement. However, the UI design is derived from the model, not the other way around, because the process is underpinned by formal methods.

The USIXML language was used in a different type of XML approach (Limbourg et al., 2004). The work was intended to make it easier to convert between different versions of the

same UI and to develop concrete UIs from abstract models by supplying a mechanism for translation among various UI abstract models. To support the UI design, several tools were created, including GrafiXML and SketchiXML; the TERESA tool of the XIIML type could also be used. The value of these tools resided in the fact that they offered design environments comparable to standard tools, such as Visual Studio (Studio, 2017), that enabled designers to employ drag-and-drop methods to rapidly develop computerised prototypes. Because they were intended to support abstract UI models that could be adjusted to various conditions, these models had no connection to a model of the underpinning system. Description of the UI and its interaction was undertaken at a level that allowed subsequent correlation between the models and the system model according to platform or context. Although preservation of usability over different platforms was the overall goal of the work, it was not user interaction that underpinned its definition but the underlying task models.

It is advantageous to develop UI models without dependence on system specifications for several reasons. By formalising UIs, they can be subjected to analysis and verification of target attributes, just as in cases in which formal methods are employed for application logic. Furthermore, in relation to UI development for more than one platform or context, UI models are helpful as a foundation for the transformation approach.

Similarly, a refinement process is available for UIs designed as part of a formal system specification via the use of Interactors (Duke & Harrison, 1993; Faconti & Paternò, 1990). This allows UIs to be modified based on the language describing the Interactors. This language may be Z (Bramwell et al., 1995), VDM (Doherty & Harrison, 1997) or other formalisms. However, the drawbacks of this work are that usability claims must be informally validated. This approach is also hard to communicate to designers who are unaccustomed to formal methods. In contrast, this thesis examines the complexity of a user interface via notational semantics, making it uncomplicated and easy to understand. This thesis, therefore, adopts this method of straightforward notation for ease of understanding by designers.

Calvary et al. (2004) proposed an adaptive model for plastic injectors. The adaptation of the injectors with the Comet approach. The adaptation of the injectors is based on an interactor's resource descriptions. Resources, in this context, refer to factors such as screen space. This paper also proposes the encapsulation of all Camelon reference framework models in the same component and adaptation mechanisms. However, the researcher believes that an

encapsulation in the same software component (the comet) will affect all model specifications and the adaptation mechanism's surcharge component. In addition, adaptation aspects and self-adaptation in base components may affect the system's usability. Calvary et al. have yet to develop a tool that supports this approach. In contrast to this, two case studies were developed in this research to validate the proposed model.

2.10.2 Summary

The above examples illustrate that the development of independent methods can enhance the formality of the UI design process, thus conferring the advantages of formal methods, such as ensuring that the various dimensions of the UI design are correct and providing a more comprehensive system specification that is inclusive of UI concerns as well as a closer correlation between UI requirements and the underlying system at a preliminary stage. Furthermore, as has been highlighted, the existing approaches present some limitations with regard to the formulation of a cohesive approach to software development. However, in most cases, those limitations are not so much indicative of research failure than of the goal of achieving integration. To put it differently, the studies that were reviewed are only deficient insofar as the current work is concerned, as they do not provide an appropriate solution for the issue at hand.

The approaches can only partly solve the issues of the integration of UI design with a formal software development approach because they address only one dimension of the design process. Furthermore, it is not easy to integrate them into a single, more cohesive approach because each approach either suggests a different formalism or employs available formal methods differently. Furthermore, in some works, the proposed models were not used with complex case studies, nor was usability testing conducted with the involvement of the target users.

New approaches to UI modelling must be devised to overcome some of the issues outlined above and to develop formalisms that specialists without knowledge of formal methods can employ without difficulty. It is advantageous to integrate UI design into the formal process (based on inclusion of the UI in system specifications, for example) because it concentrates on the design and thus ensures that all design components work towards the same aim. This research will apply a novel approach using simple and complex mobile user interface prototypes (which are given as examples and case studies) to test the flexibility and effectiveness of the proposed model (see Chapter 5).

2.11 Differences between this Thesis and Previous Work

In the following section, the aspects that distinguish the approach proposed in this thesis from the approaches used in earlier studies are explained on the basis of the foregoing review of the work that has been conducted on formal methods and UI design.

As presented in section 2.9, a number of general UIDLs that share a similar structure and a set theory foundation have been employed. To reduce the complexity of the approach for designers and developers, this thesis employs the set theory directly and uses simple symbols and straightforward expressions. Furthermore, the thesis aims to develop language of greater flexibility and concreteness and more usable tools. However, along with improvement in industrial techniques, better integration between languages and tools for system engineering and development must be achieved.

Apart from abstract presentation or visualisation concepts, abstraction of the visual aspects of UI and the application of a formal approach to modelling or UI description have been observed in most of the available studies. This may appear to be a sound approach, since abstraction is usually among the priorities of specification. On the other hand, UI designers and HCI specialists, being equally preoccupied with design appearance and functionality, tend to place great emphasis on visualisation. They are also inclined towards specific methods and techniques because they consider them appropriate and adequate for communication with users during the process of design. The functionality of available tools for UI “drawing” should form the basis of tools for UI structure modelling, thus enabling a natural conversion between models of abstract interaction objects and concrete interaction objects. HCI designers consider that approaching the design from the perspective of user requirements and abilities and including visual prototypes within the design process is an appropriate strategy. Thus, such methods must be recognised as efficient and should be retained in this thesis. The intention is to make it easier for UI designers to create uncomplicated UI that can be reused by employing formal methods in the UI design process irrespective of the approach adopted to address the issue of integration of UI design and a formal software development approach.

The thesis proposes the integration of formal methods into UI design as an approach to mobile UI design. It is hoped that this can be achieved in a manner that can be replicated without increasing the burden on the software development process, which already possesses great complexity. The main technique of usability assessment involves including target users

in the testing process to gain insight into the way in which UIs are employed by users and to identify the issues they encounter in doing so. To this end, the discrepancies between the proposed method and conventional methods are assessed through measures of complexity (structure design and number of elements) and through usability tests based on comparative analysis of two case study interface designs and other existing applications (see Chapter 6).

Hence, this thesis is primarily concerned with improving UI design. After a comprehensive literature review, a formal model has been developed, which is supported by a hierarchical structure according to UI elements and specifically screens. A description of the contents of the interface and how they correlate with one another is provided. This thesis differs from previous works in that it is the first to undertake a thorough assessment, adopt a formal method in the investigation of case studies of greater and lesser complexity, and conduct a comparative analysis of the case studies and existing applications.

2.12 Chapter Summary

The present chapter reviewed the literature on formal methods and UI design and highlighted differences between literature and this thesis in terms of goals and methodology. Most of the examined studies took one of two approaches, namely, integration of user interface concerns and requirements into a formal specification to achieve formalisation of the UI design or introduction of formality into various components of the UCD process. Each of these approaches has specific advantages because the user requirements of usability and functionality and formal UI models are the main determinants of the UI visual design.

The approach that will be adopted in this thesis to apply formal methods and HCI has been clearly outlined, and the description of every stage of the process will be based on the language of set theory. The formal process based on set theory and on a UI design process has been discussed and explained with the aid of several examples (see Chapter 4 and Chapter 5 for additional details). This direction of research has been adopted due to the desire to formulate comprehensive solutions of UI design that are capable of offering the assurance of correctness and robustness that formal methods can offer while at the same time taking into account aspects related to usability.

Although this thesis is grounded in theory, its contribution to real-life contexts of software development is not merely theoretical in nature but also practical. Putting forth integration techniques that are reliable and effective constitutes a significant input because both formal

methods and HCI are accorded high priority within the software development industry. To exemplify the manner in which the proposed model and methods can be employed, practical illustrations will be provided throughout this thesis. Furthermore, the new model approach will be outlined and discussed in detail in the fourth chapter, together with the case studies (see chapter 5) that are referred to throughout the thesis and that incorporate the proposed method. Each of these aspects will be comprehensively addressed in the chapters that follow.

Chapter 3: Techniques and Methodologies

3.1 Introduction

In this chapter, the research methodology and procedures employed in the creation, analysis and assessment of the proposed FMMUID are presented. More specifically, the creation of the model based on a hierarchical structure and set theory is elaborated upon in section 3.2, whereas the analysis techniques (i.e., manual analysis and a count-based technique to measure complexity) and the usability study that is employed to assess four key dimensions of the two case studies and existing application interfaces (i.e., usefulness, information quality, interface quality, and overall satisfaction) are presented in section 3.3. This section also describes the questionnaire and the tools that were employed to analyse the statistical data. An overview of the chapter is provided in section 3.4.

3.2 Process of FMMUID Development

The mobile UI design model in this thesis is based on the formal or mathematical principles at an abstract level. For the purposes of this thesis, a mathematical model is understood as a mathematical representation of how actual devices and objects behave (Cimatti et al., 2011a). Hierarchical structure and set theory are the two pillars of the mathematical model. As a dimension of mathematical logic, which is the foundation of computer science, set theory is essential for most mathematics and represents a crucial reasoning language and tool. It facilitates formalisation and rationalisation of computation and its objects. Syntax, semantics and logic are the key characteristics of set theory (Winskel, 2010). To describe the new hierarchical-structure-based model, this thesis employed set theory notations.

Both mobile UI designers and users navigating the UI screens could benefit from mobile applications with mobile UIs based on hierarchical structures (Chen & White, 2013; Sahami Shirazi et al., 2013). Munzner (2000) put forth a method whereby a page (screen) is placed on an abstract level of the link hierarchy established by its primary parent; provided that their maximum weight on the links to the page is the same, several pages may be candidates for the primary parent of the page and the one allocated this position is the one appearing first in the link hierarchy.

A navigable mobile UI application based on a hierarchical structure is proposed in this thesis (Figure 3.1). The hierarchical structure is advantageous because it enables users to see where

they are situated on the mobile UI in relation to other screens, it affords a greater number of frameworks, and it permits retention of the initial UI design in the model. A variety of elements make up the system of high complexity that is a mobile UI, and the manner in which all these elements are interactively correlated can be represented through the assembly of the structure of hierarchical components.

Employing a hierarchical structure that facilitates users' navigation through the hierarchy based on the selection of various options is a popular strategy. The hierarchical navigation can be implemented with a basic screen and related sub-screen. Each element of the series of container sub-screens (S_i) that constitute the basic screen (S_s) itself consists of a series of other sub-screens ($S_1 \dots S_n$). The related screen (S_{info}) can be navigated by choosing one basic screen or sub-screen (S_g) (Figure 3.1). A certain number of elements representing components, colours, and functions (I , C , and F) are possessed by every screen or sub-screen. These elements differ in terms of number and type from one screen to the next. A discussion of all the elements (S_s , S_i , S_g , S_{info} , I , C , and F) is provided in chapter four. The proposal for mobile UI design based on FMMUID has been inspired by the method of screen-component-based hierarchical structure.

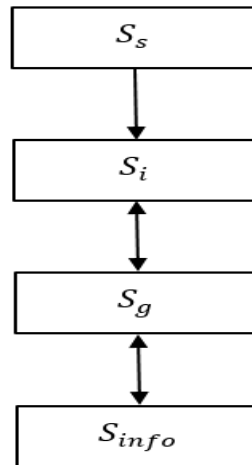


Figure 3.1: Hierarchical structure for FMMUID.

3.3 Techniques of Analysis

This section is divided into two phases. The first phase is geared towards demonstration and validation of the novel approach and uses a range of methods to address the first research question (1.4). In the second phase, a comparative analysis is undertaken to determine how

the usability of the new UI applications in the case studies compares to that of existing applications, thus addressing the second research question (1.4).

3.3.1 First Phase

The new approach is put to the test through the two case studies: iPlayCode (study 1) and SC (study 2), along with mobile applications with functionalities identical to those of the existing applications. iPlayCode represents a quiz game that is intended to provide basic programming skills to users who are completely unfamiliar with programming languages. The social media application SC is accessible to any group of users and facilitates searches for friends, services, communities, and other activities. Chapter 5 discusses these two case studies in detail.

iOS and Android are the two most popular platforms in user communities. Currently iOS and Android apps are in fierce competition, although iOS apps have ranked higher in quality than Android apps (Martínez-Pérez et al., 2013). The approach to security used by iOS also appears to be more attack-resistant (Mohamed & Patel, 2015). In 2013, Android had a 79% threat level, while iOS, in contrast, had a 0% threat level (Symantec, 2014). Hence, the iOS platform is the platform of choice for both the case study applications and the chosen reference applications in this research.

3.3.1.1 Choice of Case studies

The quiz game iPlayCode and the social media SC applications are created by the researcher of this thesis at the University of Huddersfield. An Apple iMac with Xcode 5.0.2 software and Photoshop 6 were used to create iPlayCode and SC applications, and carry out the testing through the iOS 7.0.3 (11B508) simulator.

There are several arguments justifying the selection of the iPlayCode and SC mobile applications for the purposes of the thesis. To determine how flexible and effective the new model is, as well as how adaptable it is to various design types, it was necessary to select two different types of UI designs with different levels of complexity. The quiz game iPlayCode is highly popular particularly among university students, and compared to social media applications, it has fewer screens and is not very complex. The SC application is also popular among numerous users, but it has many screens and is of significantly greater complexity.

3.3.1.2 Choice of Existing Applications

According to functionality and user services, two major types of mobile applications are currently available. In keeping with the design analysis, six mobile applications have been selected from the app store, three for each of the two case studies, for comparison. Table 3.1 presents the three mobile applications chosen for case study 1; all are educational applications intended to impart knowledge and skills to users (see Appendix A).

Table 3:1 The mobile applications selected for the first case study (iPlayCode) alongside their descriptions.

No.	Name of Mobile Application	Purpose of the application
1	iPlayCode (Ihnissi & Lu, 2015)	Designed to teach beginners basic programming skills.
2	DK Quiz (Kindersley, 2012)	Designed for people to practice and develop their General knowledge skills.
3	Duolingo (Duolingo, 2012)	Designed for people who are interested in learning new languages.
4	C/C++ Quiz (LTD, 2015)	Designed for programmers to learn and test their C/C++ programming skills

Table 3.2 presents the three mobile applications chosen for case study 2, which provide services associated with dating, making friends, and finding places. Appendix B provides an overview of all the existing (social media) applications.

Table 3:2: The mobile applications selected for the second case study (SC) alongside their descriptions.

No.	Name of Mobile Application	Purpose of the application
1	SC	Social networking with classified ads services
2	Google+ (Google, 2011)	Interest-based social networking
3	Facebook (Facebook, 2009)	Online social media and social networking service
4	Gumtree (Gumtree.com, 2008)	Free local classified ads

Three criteria have been applied to select the existing apps, namely, download figures, rating, and popularity, with some scoring below 4.0 out of 5.0 and others scoring more than 4.0 out of 5.0.

According to Table 3.3, apart from “C/C++ Quiz”, which has a small number of users, all mobile applications that were selected enjoy a high degree of popularity. Differences in rating, number of users, and design help to determine how flexible the new model is.

Table 3:3: iOS mobile applications selected for comparative analysis and their features.

No.	Product name	Free	Rating 5.0	Number of users downloading	Quiz game	Social media	Platform	Published date	Collection date
1	DK Quiz	Yes	4.6	4305	Yes	No	iOS	Updated 9/7/2015	17/3/2016
2	Duolingo	Yes	4.7	114431	Yes	No	iOS	Updated 16/3/2016	17/3/2016
3	C/C++ Quiz	Yes	3	10	Yes	No	iOS	Updated 25/2/2015	17/3/2016
4	Google+ - interests, communities, discovery	Yes	4.1	71257	No	Yes	iOS	Updated 2/3/2016	17/3/2016
5	Facebook	Yes	2.4	2884179	No	Yes	iOS	Updated 3/3/2016	17/3/2016
6	Gumtree	Yes	4.6	15166	No	Yes	iOS	Updated 1/3/2016	17/3/2016

3.3.1.3 Design Analysis

The methods employed to analyse the UI design are presented in the following part. Validation of the new model based on a comparison between the case studies and the existing applications in terms of the design of the hierarchical structure (see Appendix D) was conducted on the basis of manual analysis and a count-based method (Altaboli & Lin, 2011). The manual analysis involved the case studies and downloading the existing applications on an iPhone. Their hierarchical structures were drawn on paper and the hierarchical structures for the case studies were compared with those of existing applications in terms of complexity (how complex the case studies were compared to the existing apps). Fundamentally, four primary provenances of UI complexity have been quantified: the number of screens; the number of elements on each screen; the number of functions on each screen; and the number of colours on each a screen. The interface complexity and comprehensibility may thus be heightened by increasing the amount of these elements. In addition, whether the model screens exist in current applications was investigated.

The count-based method of assessment can be implemented without difficulty and involves counting the elements on the screen (e.g. the number of components, functions, or colours). According to their characteristics, there are four categories of mobile app components, namely, control components (e.g. buttons, segmented, text fields, check boxes, radio buttons, switches, date pickers and pickers), content components (e.g. screen, table, action sheet, and alert), vision components (e.g. text view, collection view, image view, default cell styles, video view, and activity view controller), and navigation components (e.g. navigation bar, menu, search bar, and tab bar) (Ihnissi & Lu, 2014).

Model demonstration involved a comparison between the two case studies and existing applications in terms of the number of elements on a single screen (S_s , S_b , S_g , and S_{info}) and multiple-screens (all screens). This approach was straightforward to implement in the process of design and its purpose was to measure the complexity (more discussion in chapter six).

The approach was applied to the mobile UIs to examine it in more depth and the obtained results were subjected to the ANOVA test, t-test, and f-test (see section 3.3.2.7) to determine whether the case studies differed from the existing applications.

3.3.2 Second Phase: Usability Assessment Methods

User feedback and assessment constituted an essential part of the approach adopted in this research because determination of the usability of the two case studies from the users' viewpoint was the overall aim of the thesis. The reason for allocating so much weight to the users' viewpoint is that the experience of the end-users is a key factor determining whether a mobile UI application is successful or fails.

3.3.2.1 Usability

The software industry embraced “usability” during the 1990s, although the concept was introduced in the early part of the previous decade (Bygstad et al., 2008; Lewis, 2006a; Nielsen & Molich, 1990). It is relevant for many different disciplines because it is multi-faceted in nature (Ferré et al., 2001). However, usability is primarily understood in the form in which it is adopted in Usability Engineering (UE) in relation to the UI, as representing how easy a software system is to use and learn (González et al., 2008; Juristo et al., 2007a). The field of UE is geared towards endowing the UI design with usability through the use of structured approaches in the lifecycle of system development (Scholtz, 2004). To put it differently, through its goal of enhancing a software system's UI, UE can be understood as a process that makes use of a series of approaches at different stages of development to meet the usability objectives of that system (Jakob, 1993; Lecerof & Paternò, 1998). Ferré et al. (2001) emphasised that apart from UI appearance, usability is also concerned with system-user interaction, given that this concept is a key dimension of the field of HCI (Juristo et al., 2007a), which seeks to determine the efficiency and ease with which a software or product enables users to undertake their intended activities (Han et al., 2001). Despite being still a relatively new concept, usability has come to be considered among the essential implications

that are associated with how users, systems, tasks and the environment interact with one another (Lewis, 2006b).

3.3.2.1.1 Usability Definitions

In 1971, Miller was the first to propose usability as an indicator of “ease of use” (Shackel, 2009). Originally deriving from the term “user-friendly” (Folmer & Bosch, 2004), usability has been defined in many different ways, as is the case with numerous other terms in software engineering (Shackel & Richardson, 1991). Indeed, usability has been defined according to the particular perspectives that have been adopted in different studies (Bevan & Kirakowski, 1991b; Dubey & Gulati, 2012). The term is closely correlated with a series of dimensions, including the speed with which an action can be undertaken, performance, learnability, and user satisfaction. All of these dimensions are associated to some degree with usability, which can thus be understood as an attribute of quality (Iso & Std, 2001; Juristo, 2009). Usability was comprehensively described by Jakob (1993) as being one of the defining attributes of a system, and as indicating the extent to which users accept the system and consider it suitable for meeting their needs and requirements. Usability has been defined in myriad ways (Abran et al., 2003; Ferré et al., 2001; Juristo et al., 2007a; Juristo et al., 2007b), but some definitions are more relevant than others (Casaló et al., 2010), particularly those put forth by the International Standardisation Organisation (ISO) (Abran et al., 2003). For example, a widely used definition is the one in part 11 of ISO 9241-11 (1998), where usability is characterised in terms of three user-related dimensions, namely, effectiveness, efficiency and satisfaction, and the degree to which users consider a software or product as demonstrating these dimensions in particular settings (Abran et al., 2003; Beckert & Grebing, 2012; Stone et al., 2005). Meanwhile, usability has been defined from both product- and user-based perspectives in the ISO standard for software qualities (ISO 1991b, cited in (Bevan & Kirakowski, 1991a)) as a series of software properties that are related to the effort that usage entails and users’ evaluations of usage. Conversely, usability was explained by Brinck et al. (2002) as the ability of users to carry out more than one task. A different study described usability as the ability to use a product to perform a given task rapidly and without difficulty (Dumas & Redish, 1999). In addition, Lewis (2006b) considered the interactions among users, products, tasks and environments to be the cornerstone of usability. The most straightforward definition of usability was cited by Juristo and colleagues (2007), who described the concept as “quality in use” (Abran et al., 2003; ISO, 1999).

3.3.2.1.2 Prototyping as a Design Solution

In the context of UI design, a procedure of great importance is prototyping (Buchenau & Suri, 2000). Usability specialists are charged with monitoring users during task performance as a way of testing prototypes that resemble the actual system (Walker et al., 2002). Thus, a prototype can be understood as a basic model of a final interactive system, which enables clarification of the scopes of various solutions and of the requirements of users. A prototype design is a version of the true software system, but on a much smaller scale, and is intended to help stakeholders assess the acceptability of the system (Rogers et al., 2011b; Szekely, 1995). Furthermore, prototyping also enables developers to improve the design of the UI by gaining a better understanding of users and the manner in which they interact with a system and undertake general tasks (Buchenau & Suri, 2000; Nielsen, 1994b; Sharp et al., 2007). Moreover, prototyping entails replicating one or more versions of the system UI design that reflect only the key dimensions of the true system. As observed by Rubin and Chisnell (2008), it is not necessary to recreate all the functions of a system when prototyping it but only those functions that are required to achieve the specific goals of testing. Therefore, the actual system is merely simulated by the original UI prototype through the creation of representations that are based on a choice of features or users' requirements (Carr & Verner, 1997), which can be subjected to comparative analysis.

Prototyping is helpful throughout the process of system development: in the initial stages, it enables concepts to be brought before stakeholders and assessed to determine users' requirements; in the middle stages, it facilitates verification of system specifications; in the final stages, prototyping can be used to solve issues related to usability or design. Furthermore, apart from facilitating selection of alternative designs, prototyping can help address research queries as well (Rogers et al., 2011b). Researchers are divided on the issue of low- versus high-fidelity prototyping, with some, such as Preece et al. (1994), advocating the use of both for HCD; however, the majority of researchers argue that the same outcomes can usually be obtained with low-fidelity prototyping as with high-fidelity prototyping (Camburn et al., 2017; Jennifer et al., 2002). Hence, this thesis adopts a high-fidelity prototyping method for the development of UI prototypes.

3.3.2.2 Assessment of Usability

Researchers began to pay attention to the matter of usability assessment when the field of HCI first took shape (Hartson et al., 2001). As previously highlighted, the acceptability of a software application hinges significantly on the quality factor of usability (Abran et al., 2003; Madan & Dubey, 2012). Assessment can be more easily undertaken if usability is interpreted from the perspective of quality of use, thereby facilitating the amendment of any aspects that may hinder users from performing their tasks effectively (Macleod, 1994). Therefore, to determine the efficiency with which users can employ the system, it is essential to measure this quality factor by considering target users undertaking tasks with the system. Moreover, given that users' perspectives are afforded greater importance than designers' abilities, usability can be considered synonymous with assessment (Quesenbery, 2004). Identification of the system's weak and strong points and formulation of solutions for enhancing system usability are the main goals of usability assessment (Hamborg et al., 2004).

To ensure product or system usability (Rauf et al., 2010; Trivedi & Khanum, 2012), usability assessment focuses on the inspection of the UI design in terms of its efficiency, effectiveness, user satisfaction, error tolerance, and learnability. In addition, usability assessment is geared towards dealing with any issues or weak points that are flagged by users whilst undertaking tasks and that may impact system usability. Basically, the purpose of usability assessment is to determine how suitable a system is in enabling users to achieve their objectives (Buie & Murray, 2012; Stone et al., 2005). Verification of system functionality and users' responsiveness to the UI are important goals of usability assessment as well (Dix et al., 2004). In addition, the importance of usability assessment in generating feedback about software development and detecting issues and causative factors that can then be rectified has been emphasised by Rosson and Carroll (2002). In short, usability assessment represents the entire methodical process of determining how usable a software system is (Hoegh et al., 2006).

3.3.2.2.1 Assessment Techniques

A variety of approaches have emerged due to the formulation of techniques for usability assessment that help ensure that a system or application is usable (Blandford et al., 2008; Hartson et al., 2001). However, these techniques are not classified in the same way by all researchers, which gives rise to ambiguities and results in a lack of clarity about which techniques are most suitable for a particular product and about the weaknesses and

shortcomings of the techniques (Zins et al., 2004). To overcome this difficulty and better comprehend the various usability evaluation methods (UEMs) based on a proper comparative analysis, several researchers have attempted to develop a series of standardised usability measures. For instance, Riihiahho (2000) distinguished user testing, including context-based enquiry, usability testing, cognitive walkthroughs and pluralistic walkthroughs, along with usability investigation, including heuristic evolution, cognitive walkthroughs and GOMS (goals, operators, methods and selection rules), as the two major dimensions of usability assessment. Meanwhile, Harms and Schweibenz (2000) classified usability assessment into heuristic evaluation and usability testing.

3.3.2.2.1.1 Analytical Techniques

Only HCI or usability specialist assessors can carry out analytical techniques to assessing how usable a system or application is to identify potential usability issues and provide feedback as to how the issues can be addressed and system/application usability can be improved (Abran et al., 2003). This approach is helpful for making design amendments, but it is not a substitute for usability testing with target users. Furthermore, as discussed in greater detail in the next part, greater emphasis is put on analytical evaluation because that the results can be obtained more quickly and it is not usually as expensive as user-based techniques (Dillon, 2001). On the downside, analytical techniques, which are also known as inspection techniques, rely greatly on the assessor's capabilities; therefore, they have a low degree of objectivity (Abran et al., 2003; Jakob & Mack, 1994).

3.3.2.2.1.1.1 Heuristic Assessment

Heuristic assessment is a commonly used informal technique for evaluating usability (Nielsen, 1993a), which is applied by usability specialists to determine whether the interactive components of a system or application comply with the established usability standards (Abran et al., 2003; Holzinger, 2005; Nielsen, 1994b; och Dag et al., 2001). The aim is to abbreviate the improvement iterations and increase the development iterations (Dillon, 2001; Ferré et al., 2001). This technique simplifies the assessors' task, as it provides a straightforward list of design guidelines for interface evaluation. Thus, the assessors simply must examine the various interactive components of the interface by going over it repeatedly and comparing those components to the list of usability design guidelines. The assessors can disseminate their conclusions upon completion of the evaluation. In a study that was

conducted on market-driven packaged software development, in which achievement of user satisfaction is critical, Dag and colleagues (2001) sought to identify usability issues by employing two techniques, namely, a questionnaire for collecting information about users' views on the software and a heuristic assessment. The heuristic assessment was the standard assessment that was proposed by Nielsen (1994b) and was comprised of ten guidelines that target the key dimensions of usability, including "user control and freedom" and "flexibility and efficiency in use".

3.3.2.2.1.1.2 Cognitive Walkthrough

Different kinds of walkthrough methods are used in the field of HCI (Lewis & Wharton, 1997). Cognitive walkthrough is a task-focused method that requires the UI specialist or usability assessor to identify the precise order of task performance and anticipate how a user will behave in relation to a certain task. Concerned with cognitive aspects such as learnability, based on investigation of users' mental processes (Rieman et al., 1993), cognitive walkthrough relies on a technique of system learning that is popular with users, namely, exploration, to assess how easy a design is to learn and use (Nielsen, 1994b; Polson et al., 1992). This type of walkthrough necessitates a more comprehensive analysis of the order in which a user performs a series of actions, such as communication-based problem-solving at every stage and verification of whether it is possible to anticipate the simulated user's objectives and memory content to determine the precise action (Dillon, 2001; Holzinger, 2005). Aside from accurately predicting a user's most likely responses, the usability specialist must justify why a user is likely to have difficulties with particular interface features.

3.3.2.2.1.2 Model-based Technique

Although the model-based approach is not a widely employed type of assessment, there are a number of related techniques that can produce precise estimates of particular elements of user-interface interaction, including the time required to successfully carry out a task and also the learnability of a task sequence. To determine the precise order of user behaviours, the specialist carries out a comprehensive task analysis and implements an analytical model to obtain the usability index. The GOMS techniques, which was proposed by Dillon (2001), is the most popular model-based approach (Card et al., 1983). This technique involves dividing user behaviour into a series of basic elements by employing a framework that is underpinned

by cognitive psychology. Therefore, this technique allows the assessor to examine any interface design and determine the amount of time that is required by a user for task completion.

3.3.2.2.1.3 Empirical or User-based Techniques

Empirical assessment of usability involves a number of users performing specific tasks and interacting with particular interface designs with the purpose of investigating a system or application (Bastien, 2010). Upon completion of the tasks, either quantitative approaches (e.g., questionnaire) or qualitative approaches (e.g., interviews, verbal articulation of opinions, field observation, and focus groups) are employed to collect data from the users. It is believed that this method of employing target users to undertake different tasks is the method of assessment of system or application usability that has the highest reliability and validity (Dillon, 2001). Assessors use software prototype design to determine usability according to users' opinions (Catarci et al., 2004), which makes it easier to identify not only possible issues with usability, but also the features of an interactive system that are most preferred by users (Abran et al., 2003; Freiberg & Baumeister, 2008). The importance of user testing as a technique of usability assessment has been frequently highlighted, with some researchers even going as far as calling it "irreplaceable" Nielsen (1994b), because it supplies invaluable information about the manner in which the users approach task performance as well as about the difficulties they face when interacting with a particular interface. In addition, user-based assessment is the most appropriate method for testing usability in this work, as system implementation has already occurred (Costabile, 2001). By affording such great significance to human factors, the user-based technique helps determine how effective and efficient a system is and how satisfied users are with it, as well as whether it contains any issues or flaws and what kind of amendments are necessary (Dillon, 2001). In addition, there is agreement that the user-based technique can achieve usability assessment of the highest degree of reliability and validity (Abran et al., 2003). For all these reasons, it is the preferred method for assessing how usable a system is.

3.3.2.2.1.4 Comparative Analysis

The comparative analysis assessment technique can be applied at any stage of the lifecycle of product development. It can facilitate comparison of a range of potential designs in terms of their suitability for a specific system, in which case it is known as competitive usability

testing, or it can aid comparison between a novel interface and existing versions or a similar system from rival companies. When comparison is undertaken between different designs to assess their usability, learnability, strengths and weaknesses, the method is applied in an informal way and is exploratory in nature. In contrast, when the analysis involves a controlled experiment with various groups of users, the method is carried out in a more formal manner (Shneiderman, 2010).

3.3.2.2.1.4.1 Query Methods

Taking the form of interviews or questionnaire, query methods are intended to directly question the users regarding their experience of employing a certain system. Mostly subjective data are collected from the users with these methods, but objective data can also be obtained, owing to the ability to capture the users' physical reactions to the system. Questionnaires are usually used when the aim is to gain an understanding of what the users think about a system or product as well as why they prefer a certain system or product (Carvalho, 2001).

3.3.2.2.1.4.1.1 Interviews

The interview is a popular tool for acquiring information regarding the system requirements of users, stakeholders and domain specialists (Maguire, 2001). There are two types of interview: structured and semi-structured. The structured interview is usually preferred when the interviewees' different responses can be anticipated, but it is necessary to know how strong each view is (Macaulay, 1996). The semi-structured interview involves asking interviewees a set of fixed questions but also offering them the opportunity to elaborate on their answers further. Therefore, the semi-structured interview is most appropriate when interviewees' various responses cannot be anticipated, but there is a good understanding of general matters.

3.3.2.2.1.4.1.2 Survey

Asking users to provide the necessary information is the most straightforward way of exploring many of the facets of usability. The questionnaire is a widely used tool for collecting information about matters that are challenging to assess in an objective fashion, such as matters related to users' subjective satisfaction and the concerns that they might have (Jakob, 1993; Karat, 1993; Shneiderman & Plaisant, 2005). Demographic information is also

commonly collected with the help of the questionnaire (Rogers et al., 2011b). The procedure of the questionnaire involves gaining insight into users' needs and requirements, work practices, and opinions about novel systems or concepts by administering a series of questions to a sample of target users that they need to answer in writing. The data that are collected in this way are quantitative in nature and the procedure is considered particularly useful, as it enables the rapid questionnaire of a large number of users (Preece et al., 1994). Among the different user-based methods, the questionnaire method is one of the most important. However, it is essential for the questionnaire to be formulated appropriately and the included questions to be relevant, to ensure that the obtained data are of high quality (Ferré et al., 2001). As observed by Kirakowski (2000), a usability questionnaire is advantageous because it is a source of feedback from the users' perspective, which will be representative of the larger user population, provided that the questionnaire demonstrates reliability and it is carried out properly. The questionnaire method has been applied in numerous studies that focus on usability assessment. For example, Lewis (1993) adopted psychometric techniques to improve and assess standard questionnaires for gauging subjective usability in relation to an IBM application. In general, the employed measures of subjective usability were answers to questionnaire items, which are based on the Likert scale, that captured users' opinions about how easy the application was to use and learn as well as how appealing the interface was (Alty, 1992). och Dag et al. (2001) chose both quantitative and qualitative methods in the form of two popular techniques of usability assessment, the questionnaire being one of them, to assess the usability of a system that was produced by a leading software development company. In a different study, the Computer System Usability Questionnaire (CSUQ) was employed to gain information about what users thought of a prototype (Schnall et al., 2012). Consisting of 19 items, the IBM-developed CSUQ was geared towards evaluating system usefulness, information quality, interface quality, and overall satisfaction to determine how satisfied users were with the system usability. A value of 0.95 was obtained for the coefficient alpha of the entire CSUQ, which was indicative of the method's reliability, while system usefulness had a coefficient alpha of 0.93, information quality had a coefficient alpha of 0.91, and interface quality had a coefficient of 0.89 (Lewis, 1995). This questionnaire has been adapted for gathering information for the purposes of this thesis and is discussed in greater depth in section 3.3.2.5.

3.3.2.3 Usability Testing

Stemming from the classical experimental approach, usability testing is believed to be among the most effective assessment methods of product design (Rubin & Chisnell, 2008), as it sheds light on the interaction between target users and the system UI and enables identification of the difficulties that are encountered in that interaction (Lewis, 2006b). As observed by Evans (2002), there is a direct correlation between the objectives and measurable goals of usability testing. Furthermore, usability testing is usually carried out in a location, such as a laboratory, where the practitioners can control not only the tasks that the participants have to perform with the system under consideration, but also the environmental and social factors that may have an effect on participants' conduct and behaviour during the test (Rogers et al., 2011b). Usability testing is considered the most effective way of ascertaining system usability (Ferre et al., 2017; Rogers et al., 2011b; Spencer, 2004) and is employed in human-centred design (HCD) at various stages of the development and design processes to assess the system design (Nielsen, 1993b; Preece, 1993; Rubin, 1994). It is particularly efficient when it is conducted in the context of the system development process (Jeffrey & Chisnell, 1994).

Usability testing represents a process whereby a product or system is assessed with the participation of target users to determine how usable it is (Rubin & Chisnell, 2008). It is usually conducted in a laboratory or other controlled space, with a sample of target users being asked to interact with a system or product and perform certain tasks within carefully established settings and the results being documented for subsequent analysis (Corporation, 2000; Ferré et al., 2001). According to the Microsoft Corporation, usability testing is the gold standard for measuring the extent to which a system design fulfils users' requirements to improve the task performance (Wichansky, 2000).

3.3.2.4 Research Participants

The research participants represent a portion of the general population under examination and are recruited to address the research objectives and to formulate conclusions that can be extrapolated to the entire population (Pedhazur & Schmelkin, 2013).

With regard to age, experience, background and level of education, a random approach was adopted in choosing participants from among University of Huddersfield students who volunteered for the research. The goal of the research was to examine the usability of the

mobile UIs for the two case studies and the existing applications from the users' viewpoint. As previously mentioned, the thesis investigated UI usability from the end-users' viewpoint because their feedback contributes significantly to determining whether UIs succeed or fail.

A sample consisting of five participants has been argued by numerous researchers to be sufficient to detect 80% of usability issues (Turner et al., 2006). A number of 16 ± 4 participants was established by Alroobaea and Mayhew (2014) as sufficient for identifying not only both significant and less significant issues but also design- and navigation-related issues as well as issues pertaining to a system's functionality and purpose, especially within the context of comparative research. Nevertheless, a larger number of participants was targeted by sending invitation to all researchers in Hot Desk Area in Computing and Engineering School to test the applications. A total of 496 participants responded and assessed the eight mobile applications. This resulted in obtaining 62 participants for every application. Tables 3.1 and 3.2 list the chosen applications.

3.3.2.5 Questionnaire

Because most businesses are concerned with measuring customer satisfaction, a key element of quality management is the satisfaction questionnaire. Indeed, this issue represents the core of the new ISO 9000 (2000) norms (Russell, 2000). To determine how humans interact with device interfaces, usability tests include an empirical evaluation that measures attributes such as usefulness, information quality, interface quality and overall satisfaction from the perspective of user experience.

Determining how satisfied users were with the features of usefulness and usability was the reason for conducting the questionnaire. The answers to the Computer System Usability Questionnaire (CSUQ) developed by IBM constituted the source for the various measures (Lewis, 1995). This study adopts the CSUQ questionnaire based approach, which avoids the need to carry out physical tests under controlled conditions. The CSUQ method can be used in a wide variety of circumstances, with different sets of users and different physical environments. In addition, usefully, in 9 out of 10 cases the CSUQ approach yields the same results regardless of the number of questionnaires returned (Lewis, 1992; 1995; 2002). Analysis of different sets of questions yields four different sets of results, which reveal overall user satisfaction with the software, users' assessment of the interface, the information contained in the system and its functionality. In this way, a wider view can be obtained of users' opinions and used to reduce difficulties with the system.

The respondents were given a choice of answers based on a five-point Likert scale in which 1 and 5 denoted strong disagreement and strong agreement, respectively (Likert, 1932). The Likert scale is considered a highly valid and reliable tool for the measurement of social and political perceptions because it allows respondents to be directly involved in the process, which is the reason it enjoys such great popularity (Taylor & Heath, 1996). Research in the field of HCI often makes use of the Likert scale tool (Love, 2005). As explained by Taylor and Heath (1996), the Likert scale not only ensures the direct participation of the target group from whom data are collected, but it is also highly reliable and valid. For these reasons, the Likert scale is one of the tools most frequently employed to measure people's attitudes and perceptions with regard to a wide range of issues of a social, political or other nature. Furthermore, the questionnaire integrates four metrics that can be examined to enable particular arguments regarding UI usefulness and usability to be extrapolated.

As mentioned above, a Likert scale was used to determine the participants' final views regarding the statements associated with each research dimension. This required calculation of the weighted mean of the answers to the statements for each dimension to indicate its significance. The weighted means that were obtained are presented in in Table 3.4.

Table 3:4: Scores obtained for each statement in a questionnaire based on the Likert scale.

Response	Weight
Strongly Disagree	1
Disagree	2
Neutral	3
Agree	4
Strongly Agree	5

To allocate the answers to each statement to a particular category, the procedure outlined in Table 3.4 was applied. Based on the weighted mean value, measurement and analysis of all dimensions were conducted in keeping with the procedure shown in Table 3.5, which indicates the criterion of Likert-scale range, also known as the statistical range.

Table 3:5: The weighted mean criterion in the Likert scale.

Response	Weight Mean
Strongly Disagree	From 1.00 to less than 1.80
Disagree	From 1.80 to less than 2.60
Neutral	From 2.60 to less than 3.40
Agree	From 3.40 to less than 4.20

Strongly Agree	From 4.20 to less than 5.00
----------------	-----------------------------

The 19 questions that make up the CSUQ are divided into four sections. Questions 1-7 constitute the first section, which aims to address how useful a system is. The answers to the questions in this section can be used to gauge whether the users considered the expected services to be absent or present. Questions 10-15 make up the second section, which focuses on the quality and relevance of information regarding the interfaces. Questions 9, 16, 17, and 18 are included in the third section, which addresses the interface quality and the extent to which users are satisfied with the presentation of the assessed interactive system. Questions 8 and 19 are included in the fourth section, which is used to generate an overview of user satisfaction by considering the entirety of attributes.

Some additional questions are included in the CSUQ to obtain personal information about the participants, including gender, age, experience, and educational level (see Appendix C).

Closed-ended questions are included in the online questionnaire. Closed-ended or fixed-response questions are intended to measure how strongly respondents feel with regard to certain statements (Jordan, 2002).

This thesis is used a questionnaire to evaluate the usability. Questionnaires are widely used in a broad range of research disciplines (Lazar et al., 2010). They are an excellent way to gather information for analysis (Zaharias & Poylymenakou, 2009). As discussed by Dix et al. (1998) the process of questioning users about their use of an application, and gathering their responses can take place at first hand in an interview setting, as well as more remotely in a written document. Analysis of responses, as Spencer (2004) points out, yields information that is invaluable in the development process. Nowadays of course, questionnaires can be distributed by email or other forms of social media, which is almost instantaneous and can have negligible costs. These developments are investigated and discussed in the work of Root and Draper (1983) and Zaharias and Poylymenakou (2009).

3.3.2.5.1 Data Collection Approach

The data necessary for the purposes of this thesis were collected using quantitative method. In keeping with the suggestion of Rogers et al. (2011a), a quantitative method approach was adopted to derive quantitative data from a questionnaire (Creswell & Clark, 2011) to attain optimal outcomes for the usability assessment.

Research that generates results that can be subjected to statistical analysis and summary is classified as quantitative research, and quantitative data analysis yields results in a numerical form. Questionnaire and experiments are the methods most frequently used to collect data in quantitative research. Although positivist research makes the greatest use of quantitative data, critical and interpretive research may use such data as well (Oates, 2005). In this thesis, data on each of the eight UI applications were derived from the same questionnaire. In addition, sixty-two users at Huddersfield University were involved with this study, evaluating the user interfaces of eight applications. This took place in July 2016.

Usability assessment involves the collection of data from users after they have tested specific applications to learn what they think about them (Teoh et al., 2009). UI app usability can be determined in various ways. In this work, UI app usability was measured from the perspective of the users.

UI usability evaluation is most frequently undertaken on the basis of the inspection method and the user testing method. The inspection method has been applied in earlier studies in the form of heuristic evaluation and cognitive walkthrough (Jeng, 2005; Nielsen, 1995; Thompson et al., 2003); in this method, little importance is assigned to input from end-users (Banati et al., 2006). On the other hand, the user testing method relies primarily on the use of a questionnaire for data collection (Hsieh & Huang, 2008; Thompson et al., 2003), and it is recommended that the questionnaire be provided during the time when the system is engaged by the users (Banati et al., 2006; Hsieh & Huang, 2008; Thompson et al., 2003). The user testing method yields direct information about the manner in which users interact with the interfaces; therefore, it is deemed an efficient method for usability measurement (Nielsen, 1994b).

The test questionnaire is intended to gather information about users' perspectives and earlier experiences as well as demographic details related to the process of usability assessment. Because this questionnaire approach is rapid and inexpensive and permits collection of quantifiable data, it was chosen for the purpose of the thesis (Lazar et al., 2010).

3.3.2.6 Procedure

The creation and dissemination of the questionnaire in this work were achieved with the help of the Google online questionnaire software. Online questionnaires are advantageous because

they are inexpensive and easy to produce, distribute, and recover in completed form. Furthermore, the online questionnaire developed in this thesis was designed to be compatible not only with PCs and laptops but also with smartphones (Rogers et al., 2011a).

Prior to initiating the research process, the participants were informed of the study aims and objectives as well as of the experimental goal of assessing the usability of the designed mobile UI apps (see Appendix C). If the participants indicated that they were familiar with the concept of usability, they were asked to explain the concept to ensure that they really did understand it; if they were not familiar with it, they were given a succinct explanation of the concept and what it entailed. No time limitations were imposed on the participants with regard to completion of testing the eight mobile UIs. An Apple iMac with Xcode 5.0.2 software was used to conduct the testing through the iOS 7.0.3 (11B508) simulator. Subsequently, the participants were asked to compare the two case studies with each of the existing applications.

Once the participants had completed the testing, they were given a questionnaire and asked to indicate how usable the mobile UIs were with respect to usefulness, information quality, interface quality, and overall satisfaction as well as to provide recommendations for UI improvement. The Microsoft Excel programme was used to process and analyse the answers provided by the participants.

3.3.2.7 Statistical Analysis of Quantitative Data

The quantitative data were analysed using various methods of statistical analysis. However, a key consideration was that the statistical tests employed had to permit data analysis during the research planning stage to permit determination of the validity of the formulated hypotheses (Wood et al., 2000). The initial procedure was determination of Cronbach's alpha coefficient to measure the reliability of the questionnaire. A result or outcome of a process or event can be described as reliable if the result can be repeated consistently (Moliterni, 2008). The value of this coefficient usually falls between 0 and 1. The reliability measurement is based on a single test that yields a singular reliability estimate (Gliem & Gliem, 2003). In general, the following standards are adopted: ≥ 0.9 – Excellent; ≥ 0.8 – Good; ≥ 0.7 – Acceptable; ≥ 0.6 – Questionable; ≥ 0.5 – Poor; and ≤ 0.5 – Unacceptable (George & Mallery, 2003). Subsection 6.4.2 presents the results obtained from the assessment of the questionnaire reliability based on Cronbach's alpha coefficient.

3.3.2.7.1 Normality

In a graphic representation of data, as described by, for example (Hair et al., 2003), the normality of a curve or distribution is a measure of how closely it matches that of an average or normal dataset. Two distinct properties of the curve are important in determining this. ‘Skewness’ is a measure of asymmetry in the distribution. ‘Kurtosis’ is a measure of the distribution of spikes or peaks in the data curve. Pallant and Manual (2010) provide a full description of these properties. According to Fidel (2000), Grayetter and Wallnau (2014) and Trochim (2006) a curve can be considered normal if the skewness and Kurtosis values are within the range ± 2 . For results in this study see section C in Appendix H.

3.3.2.7.2 Parametric and Non Parametric of Data Analysis

Tests for analysing obtained data can be broadly divided into two types: parametric and non-parametric. Parametric tests are more stringent in their definitions of constants and variables and are therefore generally regarded as more accurate. However, non-parametric tests are often used if either the quality of data is insufficiently rigorous to permit parametric testing, or the variable which is the actual subject of research has not been measured at a sufficiently detailed level (Rockinson-Szapkiw, 2013). The following shows the different uses of these types of tests.

Table 3:6 Parametric and Nonparametric testing (from Pallant (2013)).

	Parametric	Nonparametric
Assumed Distribution	Normal	Any
Assumed Variance	Homogeneous	Any
Level of Measurement	Ratio and Interval	Any; Ordinal and Nominal
Central Tendency Measure	Mean	Median, Mode
Statistical Procedures		
	Independent samples t test	Mann – Whitney test
	Paired Sample t test	Wilcoxon
	One way, between group ANOVA	Kruskal- Wallis
	One way, repeated measures ANOVA	Friedman Test
	Factorial ANOVA	None
	MANOVA	None
	Pearson	Spearman, Kendall Tau, Chi Square
	Bivariate Regression	None

Non-parametric tests do not necessarily use the data obtained in a study, and do not require data points to follow a particular or ‘normal’ distribution. Rather they are based on analysis of results gains a subjective ranking of criteria. They are therefore generally regarded as less accurate.

3.3.2.7.3 Normality in Parametric Testing

Parametric testing is generally more rigorous and accurate than non-parametric testing but requires the determination of a ‘normal’ data distribution for the test, against which actual test data can be compared. If this is done correctly then variations can be accurately measured, for example in ANOVA testing or t-testing. According to Marshall and Boggs (2016), if the test data departs significantly from the ‘normal’ distribution used in analysis this does not affect the accuracy of the result. (Marshall & Boggis, 2016).

For purposes of representing and understanding the data, the general attributes were subjected to descriptive statistical analysis. Calculation of the average (mean) and percentages was undertaken for all independent attributes. Furthermore, to generate a scoring for each research dimension indicative of that dimension’s status, descriptive statistics was applied to the questionnaire items based on the Likert scale (Rogers et al., 2011b). ANOVA (ANalysis Of Variance) is a frequently used method for the evaluation of statistics in many fields, and was adopted in this study for evaluation of both CSUQ and ordinal data (Acock, 2010). Moreover, as more than two UI apps were tested, hypothesis assessment was performed using the one-way ANOVA test to compare the mean of the case studies and the mean of the existing applications (Acock, 2008). In other words, this method of analysis was intended to determine whether the means of the answers and number of elements differed significantly. Since the ANOVA test confirmed the existence of a significant difference, a paired t-test was conducted afterwards to compare the means of the two UIs and determine what caused the significant difference between the two UI apps in the independent screens. However, the determinant of this significant difference could not be revealed through the paired t-test (Lazar et al., 2010; Sauro & Lewis, 2011). Therefore, an f-test was then performed to shed light on this issue. This test sought to determine whether the variances of UI₁ and UI₂ were identical in different parts. The results revealed that, for both the t-test and f-test, the p-value was lower than 0.05, meaning that the null hypothesis could be rejected. If the p-value had been higher than 0.05, the null hypothesis would have been accepted (Ali, 2013; Rosnow & Rosenthal, 1996). The ANOVA test was applied in both phases, but the t-test and f-test were

applied only in the first phase, with a significant difference being discovered in the S_i screen (see subsection 6.3.1).

3.4 Chapter Summary

The methodology and approaches employed to achieve the established research aim and objectives were outlined in this chapter. The chapter presented the research methodology followed by justification of the choice of model creation method and discussion and legitimization of the methods of analysis and selection. The approach adopted for the purpose of data collection was then presented, and the procedures and methods of data analysis were outlined.

Chapter 4: Development of An novel Formal Model of Mobile User Interface Design (FMMUID)

4.1 Introduction

The new approach, known as FMMUID, is introduced in the current chapter as a solution for achieving the integration of the UI design with the formal method. Several UI models were discussed in second chapter, along with their weak and strong points. This chapter is concerned with the discrepancies between the discussed models. To this end, a formal UI model is developed to serve as a foundation for UI designs. Formal descriptions of the produced designs are put forward. In this way, the advantages of a UI model, such as the ability to demonstrate the UI attributes, can be secured.

When creating such models, it is essential to ensure that they are as simple as possible and that developers do not need to concern themselves with challenging terminology or the model development process. Therefore, in this thesis a formal model is developed, in that it fulfils every requirement associated with formalism regarding syntax, semantics and logic, thus, can be employed in a meticulous process, without being so complex that it is difficult to understand and employ.

As indicated in section 3.2, hierarchical structure and set theory language were the two building components of the FMMUID (model abstraction). This framework clearly reveals the main components of the UI and shows how they are interrelated. The purpose of the UI components is to ensure that the UI can respond effectively to users' requirements by establishing a good rapport with their demands. A variety of interface components make up the mobile interface, which is a system of great complexity. The manner in which the components of the interface interact with one another can be observed from the way in which the hierarchical component structure is assembled

The contribution of an approach and methodology that can be applied repeatedly to develop the user interface design on the basis of interface elements is the purpose of the present endeavours focusing on FMMUID. Thus, a designer or developer could determine the variables underpinning user interface design by employing the methodology supplied in our model. The variables that affect design include user specifications, various computing and environmental context data, and application limitations.

The remainder of the current chapter is organised in the following way. The basic construct of FMMUID is presented in section 4.2, focusing on FMMUID definition, equations, and constituents. A brief overview of the material presented in the chapter is provided in section 4.3.

4.2 FMMUID Development

The FMMUID in this thesis has been created manually by the author. An overview of the construction of the FMMUID is provided in this section and its main elements are defined. The FMMUID is advantageous because it can act as a foundation for the creation of new applications, uses uncomplicated terms that enables software engineers and UI designers to successfully achieve UI development, and facilitates comprehension of the UI conditions and solutions for an adequate design.

Interface designers can employ the FMMUID to create a hierarchical structure of the elements of the design as a whole. The FMMUID can achieve the analysis and construction of the elements connection model in the complex system on the basis of an approach of system structure modelling. Set theory (mathematical logic) constitutes the theoretical cornerstone of the FMMUID. This theory is applicable in various contexts. A set is made up of any assemblage of items (Hood & Wilson, 2002). The FMMUID facilitates analysis or characterisation of the manner in which elements are connected to each other in a complex system.

The complicated relationships among user interface elements impacting users as well as the application settings and user requirements impacting the selection of a user interface during design have been taken into consideration by our model. To successfully address these aspects, the FMMUID makes it possible to choose one suitable user interface from all potential user interfaces.

The FMMUID is represented as a family of sets of screens that may have other sub-families of sets of screens and components. Each of the screens is defined as a family of sets that is a combination of some functions that are considered to be its elements. The main elements of these families of sets are the interface components, colours and functions represented by I , C and F . These elements are also represented as a sub family set of other sets.

The *FMMUID* mainly consists of seven family sets:

$$FMMUID = \{S_s, S_i, S_g, S_{info}, I, C, F\}$$

The FMMUID is built up of the familiar elements of a graphical user interface – different screens using different colours and layouts, objects on screen and the functions relating to or triggered by them. The equation at 4.1 sets out the naming conventions, the interactions and the thinking behind components of the interface, and describes them exactly.

Definition 1

The FMMUID can be mathematically represented by equation (4.1). S_s represents the family of sets of the start screen, which is the family set of all other sub-family sets of screens. S_i (main screen) is a sub family set of screen of S_s including the sub family set of screens, S_1, S_2, \dots, S_n . Notice that each sub family set of screens can contain other sets of sub family sets of screens based on the requirements of the application. S_g is a sub-family set of screen of content (calculate the result, search, etc.), and S_{info} is a sub family set of screens that represent information (such as results or information). In addition, each screen or sub-screen has three sub-families (I, C, F), where I represents the interface components of the family of sets (p, c, v), F represents the functions, and C represents the colours.

Definition 2

The union symbol "U" represents the combination of the family of sets. The union in this model is used to combine the sub family sets in to the family of sets of the main screen (S_i) family set. Therefore, the family of sets of the whole model should have all of the options and possible functions (as sub family set elements) of the model.

Definition 3

Interface components I : represent the family of sets of all of the interface components (buttons, menus, text fields, images, video views, tables, etc.) of the screens. Each of the sub family sets may have different interface components from the other family sets that link that particular family set with the following sub family sets.

Definition 4

Colours C : signify the family of sets for the colour scheme, which is a combination of different colours on different screens. The main elements of this family set are the primary colours, which are red, green and blue. Any combination of these colours will provide a

different colour as required by the designer. The family of set C represents all possible combinations of the colours using these three primary colours.

Definition 5

Functions F : represent the family of sets of all of the possible functions (count, calculate, navigate etc.) available on the different screens. The navigation, calculation or combinations of colour are generated based on the predefined function. The function set “F” is a generic set of combination of all these functions. Because of these functions there are various interface and components at each screens. Therefore, it should be subset of all the sets.

Definition 6

Each of the screens should have the components of these families of sets (I, C, F) . However, depending on the screen, the elements of the family may vary. The elements of these families of sets will be explained further via equations (4.5) -(4.9). Therefore, the elements for each family of sets of a screen can be represented as:

$$S_u = \{I_u, C_u, F_u\}, \text{ where } u \in \mathbb{N}$$

The subscript u denotes the name or the number of the screen. As previously mentioned, this analysis consists of three functions. The family of sets for a screen will have three elements, and these elements represent the family of sets as well. For example: screen S_g , which represents a family of sets, will have the sub family sets of I_g, C_g and F_g as its element. S_g is the superset of S_{info} and the elements of S_{info} are I_{info}, C_{info} and F_{info} . Therefore, the family of set C_g should include the elements of the sub family set C_{info} , and hence S_g will have C_{info} as an element or sub family as well. Similar trends will occur for other elements and other screen families of sets as well.

Definition 7

Equation (4.1) demonstrates the relationships between the family of sets and the elements for each sub family set. This definition illustrates the elements of the sub family sets for equation (4.1).

$$\text{family of set of } S_{info} = \{I_{info}, C_{info}, F_{info}\}$$

$$\text{family of set of } S_g = \{I_g, C_g, F_g\}$$

where $S_{info} \in S_g$

family of set of $S_i = \{I_i, C_i, F_i\}$

where $S_g \in S_i$

family of set of $S_s = \{I_s, C_s, F_s\}$

where $S_i \in S_s$

$$FMMUID = \left(\left(\left(S_s \supseteq \bigcup_{i=1}^n S_i \supseteq (S_g \supseteq S_{info}) \right) \supseteq I \right) \supseteq C \right) \supseteq F \quad (4.1)$$

Lemma 1

From equation (4.1), it can be seen that S_{info} is part of S_g (i.e., S_{info} is contained within S_g). It is worth mentioning that the results in S_g are displayed by S_{info} in some cases. S_g exists within the family of sets of screens S_i for all values of i where $i \geq 1$, and the family of sets of screens S_i is contained within the family of sets of S_s . Where n represents the maximum number of i value. The parameter $I \geq 1$ is contained within S_{info}, S_g, S_i and S_s , and the parameter $C \geq 1$ is contained within I, S_{info}, S_g, S_i and S_s . In addition, the family of sets for function F should be included in parameters I, C and all of the screens, which is represented in equation (4.1). There are some special cases that partially (not heavily) depend on achieving all of the six conditions stated below, as will be explained in more detail later.

The proposed model is valid when all of the following six conditions are met:

$$\text{Condition 1: } S_g \supseteq S_{info} \quad (A)$$

$$\text{Condition 2: } \bigcup_{i=1}^n S_i \supseteq (S_g \supseteq S_{info}) \quad (B)$$

$$\text{Condition 3: } S_s \supseteq \left(\bigcup_{i=1}^n S_i \supseteq (S_g \supseteq S_{info}) \right) \quad (C)$$

$$\text{Condition 4: } \left(S_s \supseteq \left(\bigcup_{i=1}^n S_i \supseteq (S_g \supseteq S_{info}) \right) \right) \supseteq I \quad (D)$$

$$\text{Condition 5: } \left(\left(S_s \supseteq \left(\bigcup_{i=1}^n S_i \supseteq (S_g \supseteq S_{info}) \right) \right) \supseteq I \right) \supseteq C \quad (E)$$

$$\text{Condition 6: } \left(\left(\left(S_s \supseteq \left(\bigcup_{i=1}^n S_i \supseteq (S_g \supseteq S_{info}) \right) \right) \supseteq I \right) \supseteq C \right) \supseteq F \quad (F)$$

It is worth mentioning that other screens can be inserted between S_s and S_i , depending on the requirements of the application. This provides more dynamic and flexible applicability to the model. The same can be said for S_i and $(S_g \supseteq S_{info})$.

Proof

Suppose that the first condition (A) for the proposed model was met i.e. S_g is a family set of S_{info} . Therefore, the components or elements of S_{info} will be a part of the S_g as mentioned above in Lemma 1 and depicted in Figure 3.1. Similarly, in condition (B), S_g is a sub-family of S_i , where $i = \{1, \dots, n\}$. This represents that there can be more than one screen, and each screen may have more options that will lead to S_g . Therefore, all S_g elements have to be elements of S_i . Figures 3.1, D.1 and D.2 in Appendix D depicts the evidence of the relation between S_i and S_g components. Subsequently, condition (C) depicts that S_s is the union of all S_i , representing S_i being a sub family set of S_s . Hence, all of the components of the S_i screen and other sub screens will be part of S_s . Moreover, condition (D) shows that I (family set of interface components) is a sub family set of all of the screens. Each screen should have interface components, which link a screen with other screens. Therefore, the family set should have the interface components of the sub family set as well to establish a linkage between them. From the definition of I (Definition 3), it can be seen that each screen has a minimum of one component that links two screens. Condition (E) represents that each screen and the interface components consist of various colours. Henceforth, C is the sub family set of all of the components and screen family sets. Likewise, condition (F) demonstrates that F (family set of functions) is the sub family set of all colours and components, and the screen family sets comprise different functions.

Definition 8

S_i is a sub screen of S_s including the combination of screens of S_1, S_2, \dots, S_n . S_{kj} represents a screen for $k = \{1, 2, \dots, g\}$, $j = \{1, 2, \dots, m\}$ and S_{kjl} is a sub screen generated from S_{kj} for $l = \{1, 2, \dots, a\}$.

$$S_i = \bigcup_{k=1}^g \bigcup_{j=1}^m S_{kj} \supseteq \bigcup_{l=1}^a S_{kjl} \quad (4.2)$$

Lemma 2

Equation (4.2) represents a special case for generating S_i from S_{kj} and S_{kjl} ; the formula expressed above states that S_{kj} , for any ordered size k, j , is the family set of S_{kjl} for any ordered size l , where S_{kjl} is the sub-screen contained within S_{kj} for any maximal ordered size k and j ($k, j \geq 1$). However, it is not necessary that each S_{kj} contains S_{kjl} .

Proof

The expression above (equation 4.2) is used to determine the possible combination of screens S_1, S_2, \dots, S_n . Each S_i is obtained from S_{kj} and S_{kjl} . Each S_{kj} has an ordered size- g and m and has the probability of generating a different number of S_{kjl} depending on the value of a and may not produce any sub-screens. This expression gives an idea of how many possible sub-screens can be obtained from any given screen.

$$S_i = \bigcup_{k=1}^g \bigcup_{j=1}^m S_{kj} \quad (4.3)$$

Special case of lemma 2

Other screens can be generated via a certain part of the previous formula in another way (see equation 4.2). However, parameter S_{kj} in equation (4.3) differs from that in the original equation (4.2). The differences involve the design parameters, which include the interface components (I), colour representation (C) and functions (F).

Definition 9

S_i is a sub screen of S_s , where $i = 1, 2, \dots, n$, and generates three different screens S_{kj} (for $k, j \geq 1$), S_{kf} (for $f \geq 1$), and S_{kr} (for $r \geq 1$). However, there is only one sub-screen (S_{kjl}) that can be generated from S_{kj} for $l = \{1, 2, \dots, a\}$.

$$S_i = \left(\bigcup_{k=1}^g \bigcup_{j=1}^m S_{kj} \supseteq \bigcup_{l=1}^a S_{kjl} \right) \cup \bigcup_{k=1}^g \bigcup_{f=1}^n S_{kf} \cup \bigcup_{k=1}^g \bigcup_{r=1}^q S_{kr} \quad (4.4)$$

Lemma 3

Another way to generate S_i can be expressed by equation (4.4). The formula states that S_{kj} for any ordered size ($k, j \geq 1$) is a family set of S_{kjl} for any ordered size l (i.e., S_{kjl} is contained within S_{kj}), S_{kf} is a sub screen of S_i for ordered size k, f , and S_{kr} is another sub

screen generated from the main screen S_i for ordered size k, r . Both S_{kf} and S_{kr} are united with S_{kj} to obtain S_i .

Proof

The expression above is used to show a possible combination of family sets for screens S_i . Each set of S_i generates screens (S_{kj}, S_{kf} , and S_{kr}) and sub-screens S_{kjl} that can be generated from S_{kj} . Formula (4.4) states that each S_i is produced by a sub family set of screens S_{kj} that is in a union case with S_{kf} and S_{kr} for the ordered size (k, f , and $r \geq 1$). The generated S_{kj} has probabilities of generating different numbers of sub-screens S_{kjl} , depending on the value of l . However, it is not compulsory for each screen to have sub-screens.

$$I = \bigcup_{i=1}^n (p_i \cup c_i \cup v_i) \quad (4.5)$$

Theorem 1

The user interface component I can be mathematically expressed in formula (4.5), which contains three main features: a control process (p_i), contents (c_i), and vision (v_i). These features have a set of n inputs.

p_i represents a family set of control process components that constitute the application such as buttons and menus.

$$p = \{p_1, p_2, \dots, p_n\}$$

c_i represents a family set of content properties that include textboxes and tables.

$$c = \{c_1, c_2, \dots, c_n\}$$

v_i represents a family set of vision properties such as text view, image view, and video view.

$$v = \{v_1, v_2, \dots, v_n\}$$

Definition 10

$i = \{1, 2, \dots, n\}$ represents the number of screens.

$w = \{1, 2, \dots, m\}$ signifies the number of components, colours or functions for each screen.

$t = \{1, 2, \dots, l\}$ denotes the number of components that are available on each screen.

$a = \{1, 2, \dots, q\}$ represents the number of colours or functions for each component.

$x = \{1, 2, \dots, r\}$ signifies the number of colours including functions.

$y = \{1, 2, \dots, d\}$ denotes the number of functions for colours.

$$I = \bigcup_{w=1}^m S_u^{I_w} \quad (4.6)$$

where subscript u is used to represent the s, i, g and $info$ screens. In the following, the S_i screen is used as an example, but the same can be applied for the other screens, i.e., S_s, S_g and S_{info} .

$$I_i = \bigcup_{i=1}^n \bigcup_{w=1}^m S_i^{I_w} \quad (4.7)$$

Equation (4.7) symbolises the family set for the components available in the whole model. In a screen, a variety of components $S_i^{I_w}$ can be presented to define different functions and link to other screens. For example, $S_1^{I_1}$ represents the first component of screen 1, and $S_1^{I_2}$ depicts the second component of that screen. Therefore, the family set for component I is the combination of all of the components available on each screen.

$$I_i = \{S_i^{I_w}, \dots, S_n^{I_m}\}$$

$$I_1 = \left\{ \{S_1^{I_1}, S_1^{I_2}, \dots\}, \{S_2^{I_1}, S_2^{I_2}, \dots\}, \dots, \{S_n^{I_m}\} \right\}$$

Can also be written as

$$I = \{I_1, I_2, \dots, I_n\}$$

$$C_i = \bigcup_{i=1}^n \bigcup_{w=1}^m S_i^{C_w} \supseteq \bigcup_{t=1}^l \bigcup_{a=1}^q I_t^{C_a} \quad (4.8)$$

For every screen, there might be a combination of colours to define the components (e.g., buttons and menus). Each component (I) of the screen might have a different colour combination as well. Because the screen should represent all of the components, the family set for the colour scheme for a certain screen should include all of the colour combinations for the components.

In the above expression (equation 4.8), $I_t^{C_a}$ denotes the family set of colours for the components.

$S_i^{C_w}$ denotes the sub family set of colours of the screen, which is a combination of all of the colours available in that screen.

Therefore, for all values of i, w, t and a , the family set of colours for the screen and components can be represented as

$$C_i = \left\{ \{S_i^{C_w}, \dots, S_n^{C_m}\}, \{I_t^{C_a}, \dots, I_l^{C_q}\} \right\}$$

$$C_1 = \left\{ \{S_1^{C_1}, S_1^{C_2}, \dots, S_1^{C_m}\}, \{I_1^{C_1}, I_1^{C_2}, \dots, I_1^{C_q}\} \right\}$$

The family set of the colour scheme (C) should include all of these sub family colour sets (C_i) for each screen and its components, which can be represented as

$$C \supseteq C_i$$

The above equation can also be written as

$$C = \{C_1, C_2, \dots, C_n\}$$

$$F_i = (\cup_{i=1}^n \cup_{w=1}^m S_i^{F_w} \supseteq \cup_{t=1}^l \cup_{a=1}^q I_t^{F_a}) \supseteq \cup_{x=1}^r \cup_{y=1}^d C_x^{F_y} \quad (4.9)$$

Equation (4.9) can also be represented using a similar system to that used for the colour scheme. Each function can be represented via different symbols $I_t^{F_a}$ that might be represented with different colours. In the screen, the interface components have different functions to navigate through the system. $I_t^{F_a}$ capture all of those functions for the interface components. The colours might change when a different function is selected within the same screen. Changing the colour is a part of the function as well. Hence, the whole colour scheme is part of the function $C_x^{F_y}$, which is represented in the equation above. In addition, there might be an opportunity to have various other functions (graphical representation) on the screen that might not be captured by the interface and the colour function that are in the family set of $S_i^{F_w}$. The above equation denotes that the family set function for a screen $S_i^{F_w}$ comprises of all of the components and colours along with the other functions available in that screen. The family function set F will represent all of the sub family set functions for each screen.

$S_i^{F_w}$ represents the family set of functions that are available for each screen for $i, w \geq 1$.

$$F_i = \left\{ \{S_i^{F_w}, \dots, S_n^{F_m}\}, \{I_t^{F_a}, \dots, I_l^{F_q}\}, \{C_x^{F_y}, \dots, C_r^{F_d}\} \right\}$$

$$F_1 = \left\{ \{S_1^{F_1}, S_1^{F_2}, \dots, S_1^{F_m}\}, \{I_1^{F_1}, I_1^{F_2}, \dots, I_1^{F_q}\}, \{C_1^{F_1}, C_1^{F_2}, \dots, C_1^{F_d}\} \right\}$$

Therefore, the final F should be represented as follows, which is the family set for all of the sub family function sets,

$$F \supseteq F_i$$

which can also be represented as

$$F = \{F_1, F_2, \dots, F_n\}$$

Therefore, each screen should have some attributes of the family sets I, C and F , which are represented as $S_i^{I_w}$, $S_i^{C_w}$ and $S_i^{F_w}$.

4.3 Summary

An introduction to the formal model FMMUID was provided in the current chapter. Underpinning the initial step in the integration of the UI design process with the formal model, the FMMUID facilitates, simplifies and structures characterisation of the formal design, including its prototypes (see the following chapter). Furthermore, by providing a description of every potential design, the FMMUID affords a static perspective on UI designs, while the navigational possibilities for UI can be derived from the manner in which the components are interrelated. In adopting a design approach to model development, designers must inspect the UI from multiple angles, thus ensuring that any unidentified issues or deficiencies with the UI design are detected. In addition, the model facilitates identification of the optimal features of UIs during the preliminary design process phase. The model allows evaluation at the prototyping phase of features such as UI reactivity and consistency that are not usually visible until the later phases of design. This makes it easier to make adjustments in the event that any issues are discovered.

Following an introduction to the formal interpretation of designs based on the FMMUID, the next chapter will proceed to the creation of case studies to illustrate the application of the suggested model.

Chapter 5: Using the proposed Approach on Case-Studies

5.1 Introduction

Examination of case studies (examples) is usually the ideal way to determine how applicable a formal model is (Meedeniya, 2013). In this thesis, it has already been demonstrated that the proposed model is accurate in terms of semantics. In the following part, the model is assessed in terms of how applicable and usable it is in practice.

Two distinct mobile applications are employed as case studies to determine the applicability of the proposed model. At the same time, this approach enables assessment of how useful the FMMUID is in practice. The examples used address every aspect of the anticipated analysis of the proposed model.

The case studies addressed in this chapter are related to the FMMUID-based UI design. This is in keeping with the research goal of creating a UI for a mobile application by implementing the FMMUID to determine the feasibility of incorporating this kind of formal model into the process of UI design.

The designers or developers can manually generate the user interfaces. An abstract representation of a user interface is the main input for the UI generation process. It describes what the interface should present to the user and outputs a concrete user interface.

The quiz game iPlayCode and the social media SC are two case study examples of how application prototypes are used to assess the validity of the FMMUID. The case studies are intended to evaluate the applicability of the proposed model by employing two UI applications with different levels of design complexity. Furthermore, additional validation of the FMMUID has been achieved by comparing the hierarchical structure designs and numbers of elements in the case studies and chosen applications.

Any typical UI screen has a feature of blending different structures. However, this thesis implements a hierarchical approach to develop a UI, as illustrated in Figures 5.1 and 5.3, which also depict an overview of the levels within the hierarchical structure. In this type of approach, a particular screen within the hierarchy can be arrived at solely through a single higher-ranking screen. The structure depth, the number of hierarchical levels, and breadth, a count of the options available, are two important elements that must be taken into account when developing such a hierarchical structure.

5.2 Case Study 1: iPlayCode Application

The aim of this application is to teach beginners basic programming skills. The target learner is someone who has no experience with programming. The application guides users through a game that teaches them how to write correct syntax in programming languages based on Objective C, C#, C++, and Java for Android, Java and Python. iPlayCode uses both gaming elements such as timed challenges and rewards, and a supportive system with syntactic judgements to build a fun learning environment.

Figure 5.1 illustrates the hierarchical structure for iPlayCode, which is represented as a combination of families of sets of different screens. Each screen has been characterised as different sets of family components, colour schemes and functions. The figure below illustrates the structure of iPlayCode, via the hierarchical structure, where each screen is shown with links connecting it to others.

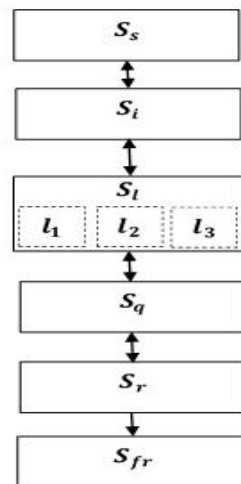


Figure 5.1: Hierarchical structure screen design for iPlayCode application.

where symbol S represents the screen and the meanings of the subscripts are shown below.

Subscript:

s	Start
i	Number of sub-screens of screen S_i
l	Levels
q	Question
r	Result
fr	Final result

5.2.1 iPlayCode Application Design Screens

S_s : S_s denotes the start screen and the elements of this screen are the options for registration (text box), selecting the preferred language (button), a function to count the number of times that it has been used before (text view), a graphical representation (image view) and a button to link to the next screen, which is the main screen (S_i). Once the user name and language have been selected, the user can move to the next screen by selecting the button (play).

S_i : S_i denotes the main screen, which shows six programming languages (buttons), when a link to the previous screen (button), when the user selects a programming language by pressing on a button, it will take the user to the levels screen (S_l).

S_l : The next screen S_l can be termed a levels screen where the user has the options to choose the level of the game (easy, medium, or hard). In this analysis, three navigation levels are used, referring to l_1 (level 1- easy), l_2 (level 2- medium) and l_3 (level 3- hard).

All of the levels (l_1 , l_2 and l_3) consist of a button to move back and change the game level and a text instruction to move forward in the game. The maximum number of options available for level 1, level 2 and level 3 are 3, 6, and 4 respectively. However, the number of options may vary depending on the option chosen from the level screen. When the user selects an option from a level, it will take the user to the question screen (S_q).

S_q : The question screen (S_q) consists of various random questions. To make the game more challenging and keep track of the score, there is a timer and score board showing the number of questions answered along with the score. There are two buttons to select, either the correct or the wrong answer and a button to move back to the previous screen. When the user selects either the “Right” or “Wrong” button from the question screen, the button text changes to “Help” and “Next” and the timer changes to the question score. After completing all of the

questions for the selected options and level, the user can check their total score, which will lead them to the result screen (S_r)

S_r : The result screen (S_r) consists of four buttons: a return button to go to the main screen and choose another programming language (S_i), one to return to the questions to play again (S_q), one to choose another level ($l_1 - l_3$), and one to go to the final result screen (S_{fr}). There is also a display to show the total score for the option and level.

S_{fr} : The final results screen (S_{fr}) shows the total score for all levels for the user. The return button on the screen will take the user to the main screen (S_i). The score is shown in image and text view.

Screen name	Screenshots of iPlayCode
S_s	
S_i	
$S_r = S_g$	
$S_{fr} = S_{info}$	

Figure 5.2: Screen shots of model screens for iPlayCode UI.

Definition 1

Equation (5.1) represents the FMMUID for iPlayCode ($FMMUID_{iPlayCode}$), which is also shown as a combination of sub family sets. Therefore, the components for those sub family sets are:

family of set of $S_{fr} = \{I_{fr}, C_{fr}, F_{fr}\}$

family of set of $S_r = \{I_r, C_r, F_r\}$

where $S_{fr} \in S_r$

family of set of $S_q = \{I_q, C_q, F_q\}$

where $S_r \in S_q$

family of set of $S_l = \{I_l, C_l, F_l\}$

where $S_q \in S_l$

family of set of $S_i = \{I_i, C_i, F_i\}$

where $S_l \in S_i$

family of set of $S_s = \{I_s, C_s, F_s\}$

where $S_i \in S_s$

Equation (5.1) represented using the proposed model in the iPlayCode application can be expressed as follows based on the developed FMMUID:

$$FMMUID_{iPlayCode} = \left(\left(\left(S_s \supseteq \bigcup_{i=1}^n S_i \supseteq S_l \supseteq S_q \supseteq (S_r \supseteq S_{fr}) \right) \supseteq I \right) \supseteq C \right) \supseteq F \quad (5.1)$$

where S_s is the start screen, S_i is the main screen, S_l is the level screens, S_q is the question screen, S_r is the result screen for each level, and S_{fr} represents the final result screen for all levels.

For example, in the above equation, the $FMMUID_{iPlayCode}$ represents the FMMUID for iPlayCode that is shown as a family set of other sub family sets of screens. The sub family set screens in this equation are S_s, S_i, S_l, S_q, S_r , and S_{fr} . Each of the screens has different function elements.

5.3 Case Study 2: SC Design

With the modern development of society and technology, people are connected with each other through various social media platforms. Most of one's day to day needs can be found on the internet. Hence, this application is a one stop link to find different types of information. This application is aimed at all types of users to find friends, services,

communities and other data. This application allows the user to add advertisements and modify them whenever needed.

The hierarchical structure for the SC application is shown in Figure 5.3. This application is modelled using an approach similar to that used for iPlayCode. This application also consists of different screens that can be represented as sets of families of elements.

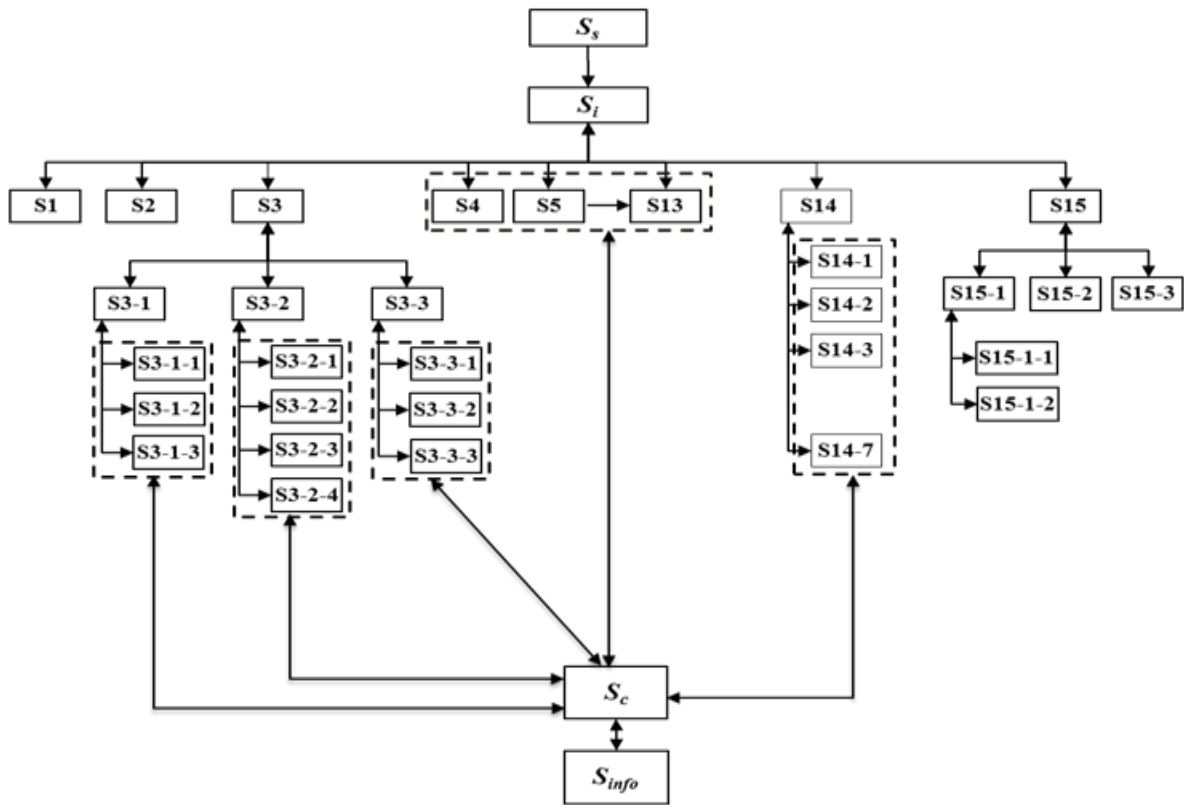


Figure 5.3: Hierarchical structure screen design for SC application.

The letters below were used in describing the screens in SC.

S Screen

Subscripts:

s Start

c Search

$info$ Information

i Number of sub-screen of screen S

$i - n$ Number of sub-screen of screen S_i

$i - n - j$ Number of sub-screen of screen S_{i-n}

5.3.1 SC Application Design Screens

S_s : S_s denotes the start screen, which consists of an image view, text view and a button to lead to the next screen (S_i).

S_i : S_i denotes the main screen, which shows fifteen navigation screens: Top list S_1 , “New releases” S_2 , “Discover” S_3 (menus), “Quick link” $S_4 - S_{13}$, “Browse” S_{14} (button) and “Advertising management” S_{15} (buttons). In addition, there are two places to upload images (image view).

S_1 to S_{15} :

S_1 : Is the first sub screen that consists of a table, image view, text view and button.

S_2 : Is the “New release” screen, which comprises a table, an image view, a text view and a button.

S_3 : Is the “Discover” screen that includes three menus, which are “Service” (S_{31}), “Communities” (S_{32}) and “Communications” (S_{33}). In addition, there is also an image view, a text view and two buttons.

The user can select various options from each of the menus, depending on their requirements. For example, under “Service”, the user can find information about “Airports” (S_{311}), “Trains” (S_{312}) and “Bus stations” (S_{313}). Similarly, for “Communities”, there are options to find “Friends” (S_{321}), “Selling” (S_{322}), “House Renting” (S_{323}), and “Pets” (S_{324}), and for “Communications”, there are options to find “Parties” (S_{331}), “Entertainment” (S_{332}) and “Gym” (S_{333}). Each of these screens consists of a number of menus, image views, text views and buttons.

$S_4 - S_{13}$: Are the “Quick link” screens showing images of the options that are available in the previous screens, providing direct access to the search screen (S_c). More options can be added to this screen to make the application more user-friendly.

S_{14} : Is the “Browse” screen, which offers the option to sell products and includes the available products in table format, image view, text view and a button. The sub-screens for this screen are “Cars” S_{141} , “Clothes” S_{142} , “Bikes” S_{143} , “Motorcycles” S_{144} , “Mobiles” S_{145} , “Laptops” S_{146} and “TVs” S_{147} .

When the user selects any option, it will lead to a search screen (S_c), where the user needs to input the location, and after that, the available results will be shown on the information screen (S_{info}).

S_1 and S_2 do not generate any sub-screens and are not connected to the search screen S_c , while $S_3 - S_{14}$ are directly connected to the search screen S_c without any sub-screens (see figure 3).

S_{15} : Is the “Advert management” screen including three menus and a button. The first menu is “Post Ad” (S_{151}), which links to two other sub screens: “Add photo” S_{1511} to upload an image for an advertisement and “Post Ad” S_{1512} to post the advertisement. The “My advertisement” screen S_{152} consists of all of the user advertisement posts so far and has an option to delete any previous advertisements. This screen consists of an image view, a text view and a button. “Saved search” S_{153} stores all previous search results and consists of a button and a table.

S_c : Is the search screen, which is connected to most of the previous screens ($S_3, S_4 - S_{13}$ and S_{14}). This screen consists of image view, a text box, a table and two buttons.

S_{info} : Is the information screen, providing detailed information about the products, events, and services. This screen consists of an image view, a table and a button.

Screen name	Screenshots of SC
S_s	
S_i	
$S_c = S_g$	
S_{info}	

Figure 5.4: Screen shots of model screens for SC UI.

Definition 2

Equation (5.2) depicts a model for social commutation (SC) represented via a family of sets having other sub family sets. The following section will illustrate the components of these sub family sets.

family of set of $S_{info} = \{I_{info}, C_{info}, F_{info}\}$

family of set of $S_c = \{I_c, C_c, F_c\}$

where $S_{info} \in S_c$

family of set of $S_i = \{I_i, C_i, F_i\}$

where $S_c \in S_i$

family of set of $S_s = \{I_s, C_s, F_s\}$

where $S_i \in S_s$

Equation (5.2) depicts the model for SC, which is also represented as the combination union of various sub family sets of screens. There are few discrepancies between equation (5.1) and equation (5.2). The main difference between the two equations is the representation of the family set of screens. However, in the SC application, the formula (5.2) representation of the model is slightly different than that of iPlayCode.

$$FMMUID_{SC} = \left(\left(\left(S_s \supseteq \bigcup_{i=1}^n S_i \supseteq (S_c \supseteq S_{info}) \right) \supseteq I \right) \supseteq C \right) \supseteq F \quad (5.2)$$

where S_c is the search screen, and S_{info} is the information screen, which has the elements as described in equation (4.1). The model can be better explained by a set of examples as follows:

Example 1: Generating third screen

Referring to Figure 5.5, the example below generates the third screen based on lemma 2.

Assume $i = 3$, $k = 3$, $j = 1, 2, 3$ and $l = 1, 2, 3, 4$

$$S_3 = \bigcup_{j=1}^m S_{3j} \supseteq \bigcup_{l=1}^a S_{3jl}$$

- Generating sub-screens from screen (S_{31})

- 1) generating S_{311} , let us suppose that $j = 1, l = 1$, then $S_3 = S_{31} \supseteq S_{311}$
- 2) generating S_{312} , let us suppose that $j = 1, l = 2$, then $S_3 = S_{31} \supseteq S_{312}$
- 3) generating S_{313} , let us suppose that $j = 1, l = 3$, then $S_3 = S_{31} \supseteq S_{313}$

- Generating sub screens from screen (S_{32})

- 1) generating S_{321} , let us suppose that $j = 2, l = 1$, then $S_3 = S_{32} \supseteq S_{321}$
- 2) generating S_{322} , let us suppose that $j = 2, l = 2$, then $S_3 = S_{32} \supseteq S_{322}$
- 3) generating S_{323} , let us suppose that $j = 2, l = 3$, then $S_3 = S_{32} \supseteq S_{323}$
- 4) generating S_{324} , let us suppose that $j = 2, l = 4$, then $S_3 = S_{32} \supseteq S_{324}$

- Generating sub screens from screen (S_{33})

- 1) generating S_{331} , let us suppose that $j = 3, l = 1$, then $S_3 = S_{33} \supseteq S_{331}$
- 2) generating S_{332} , let us suppose that $j = 3, l = 2$, then $S_3 = S_{33} \supseteq S_{332}$
- 3) generating S_{333} , let us suppose that $j = 3, l = 3$, then $S_3 = S_{33} \supseteq S_{333}$

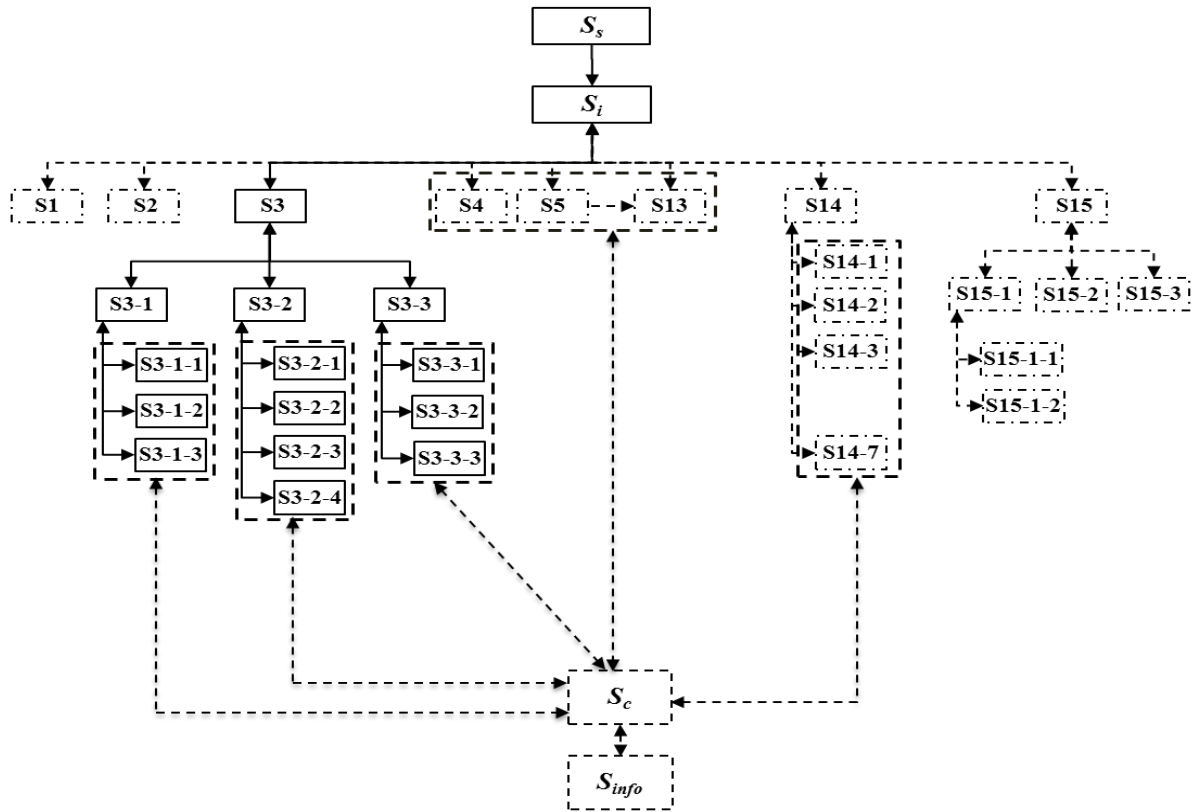


Figure 5.5: Generating the third screen (S_3) in a hierarchical structure.

Example 2: Generating fourteenth (14) screen

This example is a typical circumstance of the special case of lemma 2 in that fourteen screens are generated as described in Figure 5.6.

Let us assume $i = 14, k = 14, j = 1, 2 \dots, 7$

$$S_{14} = \cup_{j=1}^7 S_{14j} = S_{141} \cup S_{142} \cup, \dots, S_{147}$$

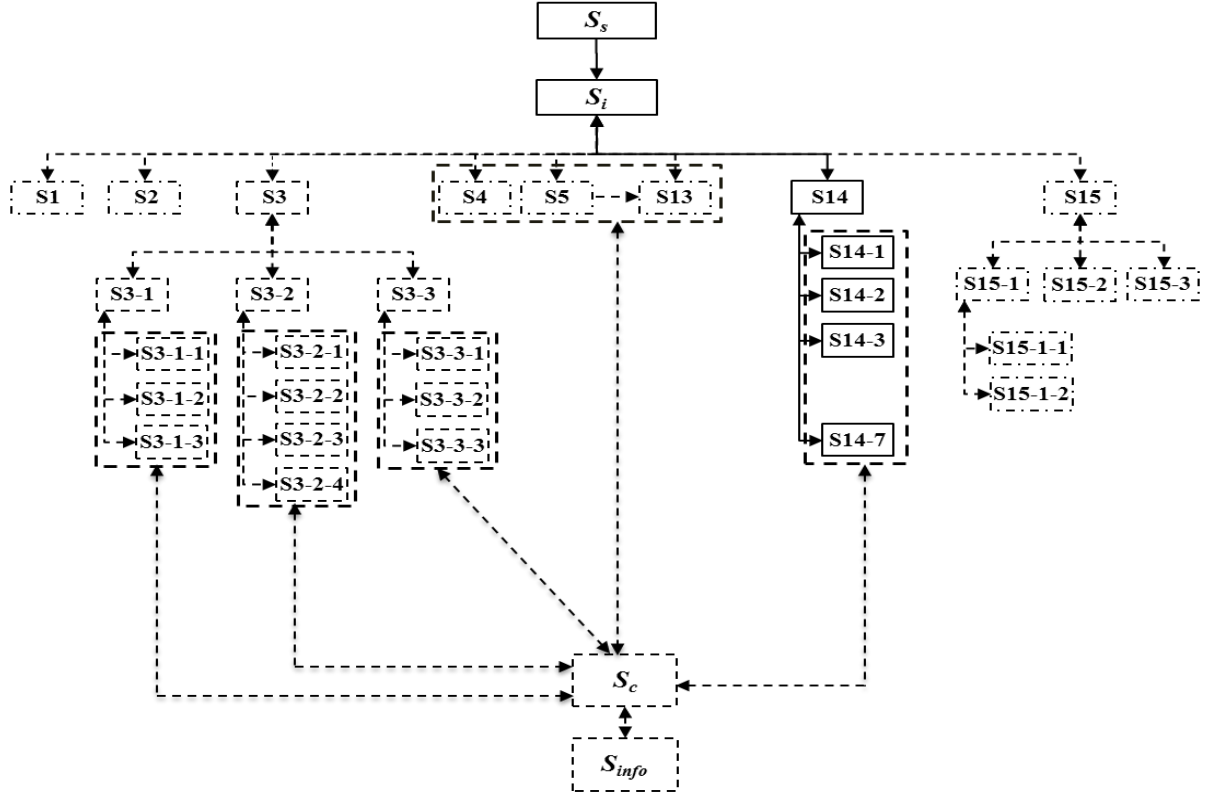


Figure 5.6: Generating the fourteenth screen (S_{14}) in a hierarchical structure.

Example 3: Generating fifteenth (15) screen

This example describes the case when Lemma 3 is applicable.

Let assume $i = 15, k = 15, j = 1, l = 1, 2, f = 2$ and $r = 3$

- Generating S_{1511} , let us suppose that: $j = 1, l = 1$, then $S_{15} = S_{151} \supseteq S_{1511}$
- Generating S_{1512} , let us suppose that: $j = 1, l = 2$, then $S_{15} = S_{151} \supseteq S_{1512}$
- Generating S_{152} can be obtained by setting value of $f = 2$
- Generating S_{153} can be obtained by setting value of $r = 3$

$$S_{15} = (S_{151} \supseteq \bigcup_{l=1}^2 S_{151l}) \cup S_{152} \cup S_{153}$$

It is worth mentioning that screen S_{15} is not linked to screen S_c .

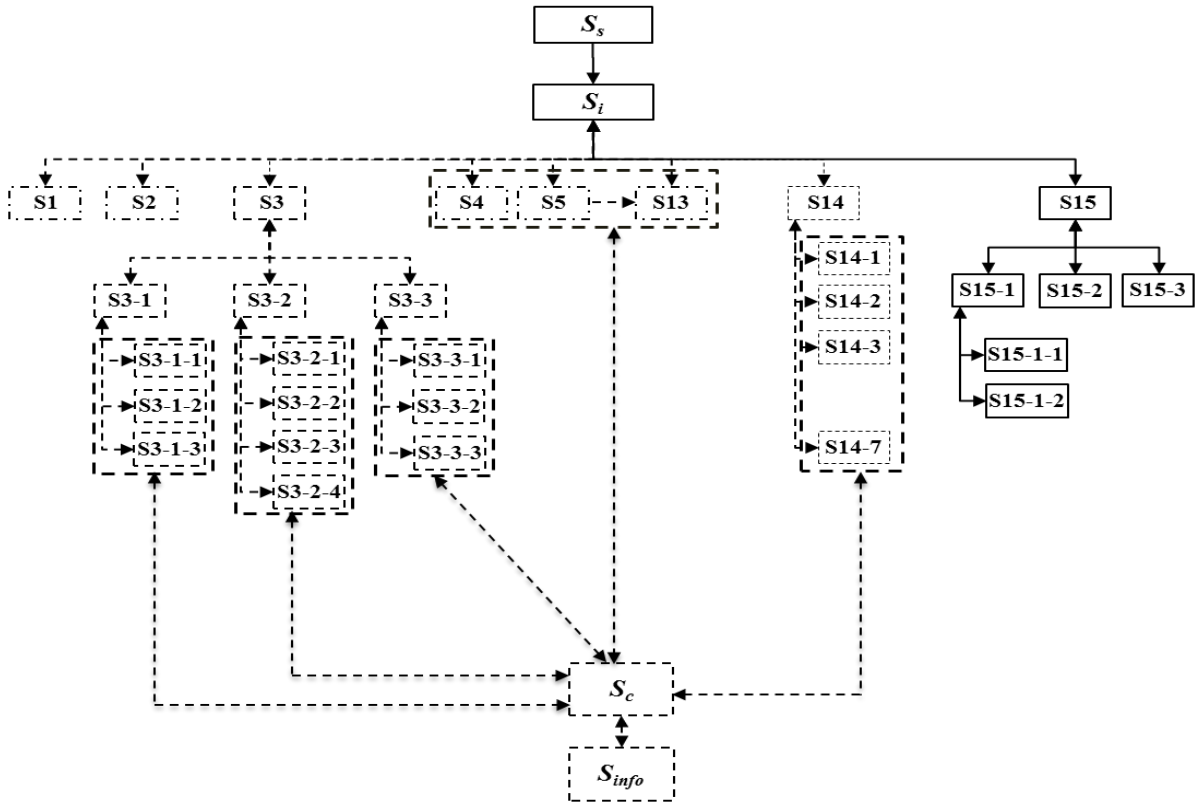


Figure 5.7: Generating the fifteenth screen (S_{15}) in a hierarchical structure.

5.4 Summary

The utility of software tools for designers and developers typically lies in their ability to facilitate analysis and implementation of original designs even in the absence of formal model expertise. This chapter has demonstrated that the proposed model can be easily and effectively implemented. Moreover, two case studies related to different levels of UI functionality were used to assess the applicability the FMMUID.

Specifically, by designing the two UIs using the model, following have been shown:

- 1- The model is flexible, as it allowed the design interfaces with different levels of complexity.
- 2- It is cost effective, as it allowed the designer to detect problems and solve them before the implementation of the final design.
- 3- The design development can be achieved with this model without the need for in-depth understanding of formal methods.
- 4- The complexity of the outcome determines the productivity of the proposed approach.

Four factors have been identified as particularly influential to the complexity of a UI

design navigation (the number of screens determines the depth and breadth), number of components, number of functions, and screen colours (see section 7.2). By enhancing these factors, the interface may be made more complex and less understandable.

In addition to applicability, the efficiency of the formal model in validating the design by detecting the hierarchical structure of limited complexity between potential diagrams has also been assessed with the help of hierarchy diagrams (see Appendix D). These procedures provided support for the accuracy of the proposed model.

Chapter 6: Analysis and Validation of the Proposed Approach to Measure the Complexity and Usability

6.1 Introduction

The main purpose of this chapter is to assess and prove the proposed model. This chapter analyses the proposed model based on two developed mobile UI application studies. The first study corresponds to iPlayCode. The hierarchical structured design of iPlayCode is created and compared with the UI designs of three other applications namely, DK Quiz, Duolingo and C/C++ Quiz in 6.2.1. Similarly, in the second study of a developed mobile application, the structured design was created for SC. The SC structured designed was compared with those of other social media apps, namely, Google+, Facebook and Gumtree as described in 6.2.3. The comparison of iPlayCode and SC with other apps is presented in section 6.3. In section 6.4, an additional questionnaire is used to evaluate the users' satisfaction in terms of usefulness, information quality, interface quality, and overall satisfaction. In section 6.5, the ANOVA test technique is used to analyse the responses of users. Finally, a summary is presented in section 6.6.

6.2 Analysis of the Structure of Application Design

The new model consists of screens designed for both the iPlayCode and SC apps. The main structured design of each app is based on common screens S_s , S_i , S_g and S_{info} . The various elements that were included in these screens are systematically tabulated and summarised in the upcoming sections. Eventually, the goal of this design analysis is to convey the straightforwardness of the proposed model.

6.2.1 Case Study 1: iPlayCode Comparison Design

The hierarchical structure of the iPlayCode design consists of six main screen names, as well as a screen subscript that represents different elements.

Figure D.1 in Appendix D, describes the comparison of the hierarchical structure design of iPlayCode with those of the other three apps. It demonstrates that the designs of all the apps consist of the main screens, namely, S_s , S_i , S_g and S_{info} . The DK and Duolingo apps consist of complicated structures, in contrast to the iPlayCode and C/C++, which have simple structural designs. iPlayCode has six levels, from S_s to S_{ff} , of depth, and one category of breadth. The

DK app consists of seven levels, from S_s to S_{info} , of depth and five of breadth (flat), from S_I to S_5 . The Duolingo app consists of nine levels, from S_s to S_{info} , of depth and four levels (S_I to S_4) of breadth. The C/C++ app consists of five levels of depth, from S_s to S_{info} , and from $S_{setting}$ to $S_{information}$ in breadth.

6.2.2 iPlayCode Screen Comparison

This section compares the main four screens of the new model (S_s , S_i , S_g and S_{info}) between the iPlayCode application and the DK, Duolingo, and C/C++ applications.

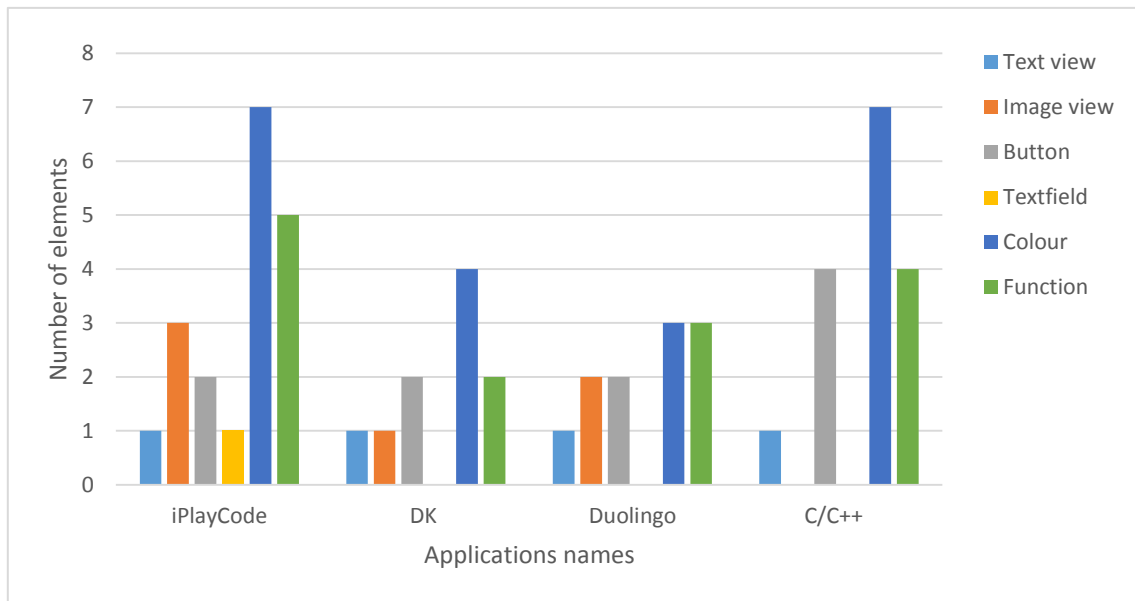


Figure 6.1: Comparison of UI elements on screen S_s (single screen) between quiz game apps.

Figure 6.1 shows the comparison of the elements on screen S_s between iPlayCode and DK, Duolingo and C/C++. The number of image views in iPlayCode is higher than that in DK by two and that in Duolingo by one; C/C++ did not have any image views. All four apps have 1 similar text view in common. In terms of buttons, C/C++ has the most buttons (two), compared with the others. iPlayCode has only 1 text field, in contrast to the other apps. In terms of colours, the most colours (7) were used in both iPlayCode and C/C++; 4 colours were used in DK and the fewest colours (3) were used in Duolingo. Additionally, the most function features (5) were used in iPlayCode; the fewest (2) were used in DK; 3 were used in Duolingo and 4 in C/C++.

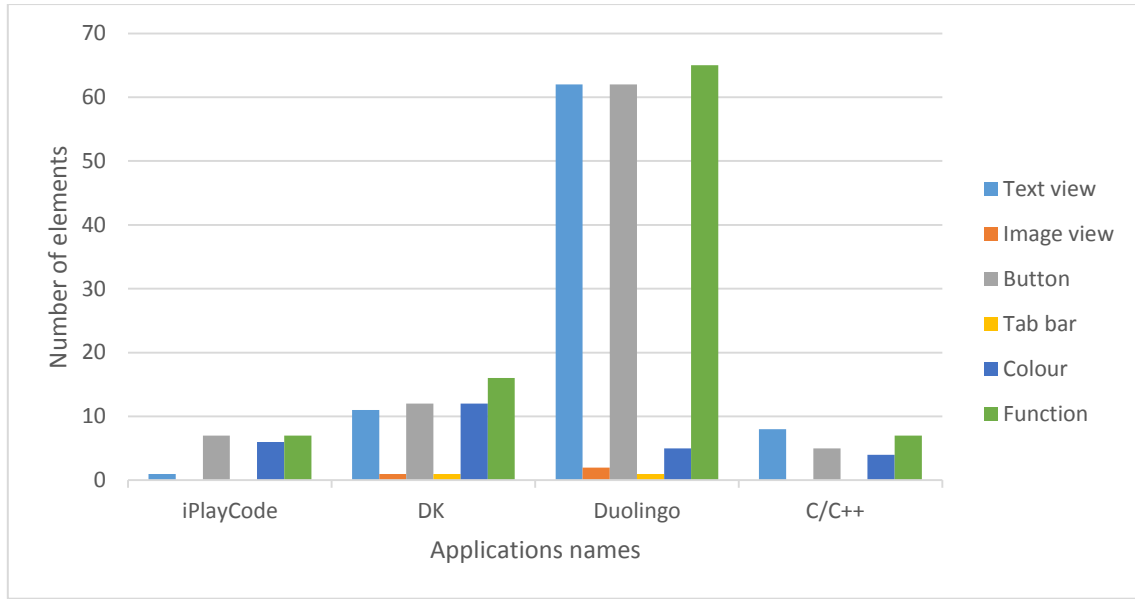


Figure 6.2: Comparison of UI elements on screen S_i .

In terms of screen S_i , Figure 6.2 illustrates that Duolingo recorded the most text views (62), whereas the fewest were recorded in iPlayCode (1). Duolingo had 1 more image view than DK, whereas iPlayCode and C/C++ did not have any image views. The most buttons were used in Duolingo (62), whereas the fewest buttons (5) were used in C/C++; seven were used in iPlayCode and 12 in DK. DK and Duolingo have the most tab bars (1), whereas iPlayCode and C/C++ did not have any tab bars. DK used the most colours (12), whereas the fewest colours (4) were used in C/C++. The most function features were used in Duolingo (65), whereas the fewest features were used in C/C++ and iPlayCode (7), followed by 16 in DK.

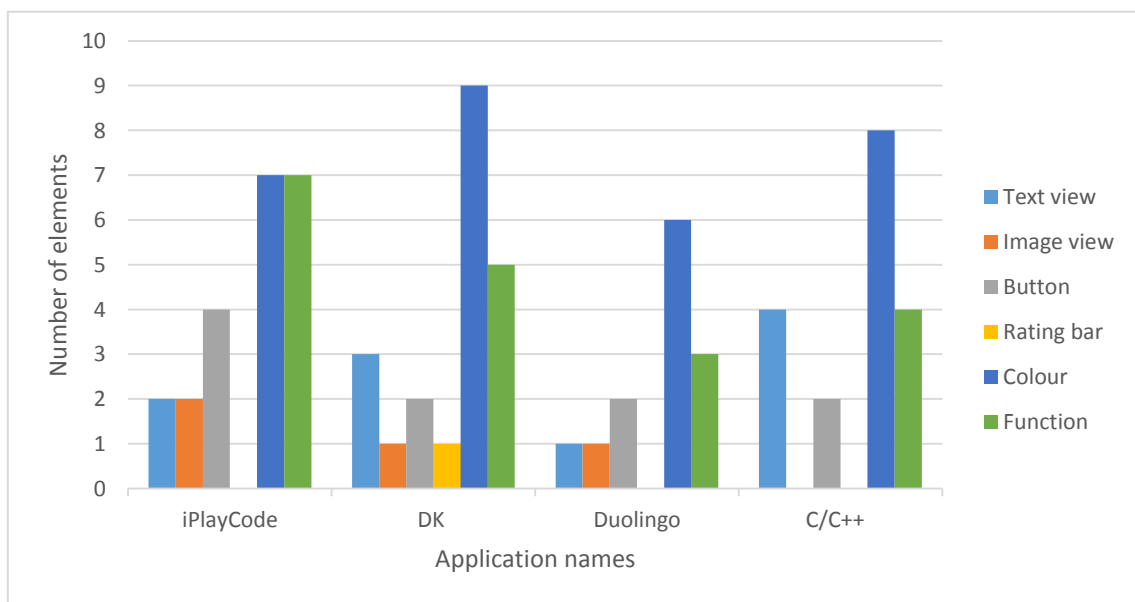


Figure 6.3: Comparison of UI elements on screen S_g .

Regarding the S_g screen, according to Figure 6.3, the most text views (4) are recorded in C/C++ and the fewest (1) in Duolingo, followed by 2 in iPlayCode and 3 in DK. The most image views (2) are used in iPlayCode, and in both DK and Duolingo, 1 view is used; C/C++ does not have any image views. The most buttons (4) are recorded in iPlayCode and the fewest (2) in each of DK, Duolingo and C/C++. DK used the most colours (9) and Duolingo used the fewest (6), followed by iPlayCode and C/C++ (7). The most function features (7) were used in iPlayCode and the fewest (3) were used in Duolingo; 4 were used in C/C++ and 5 in DK.

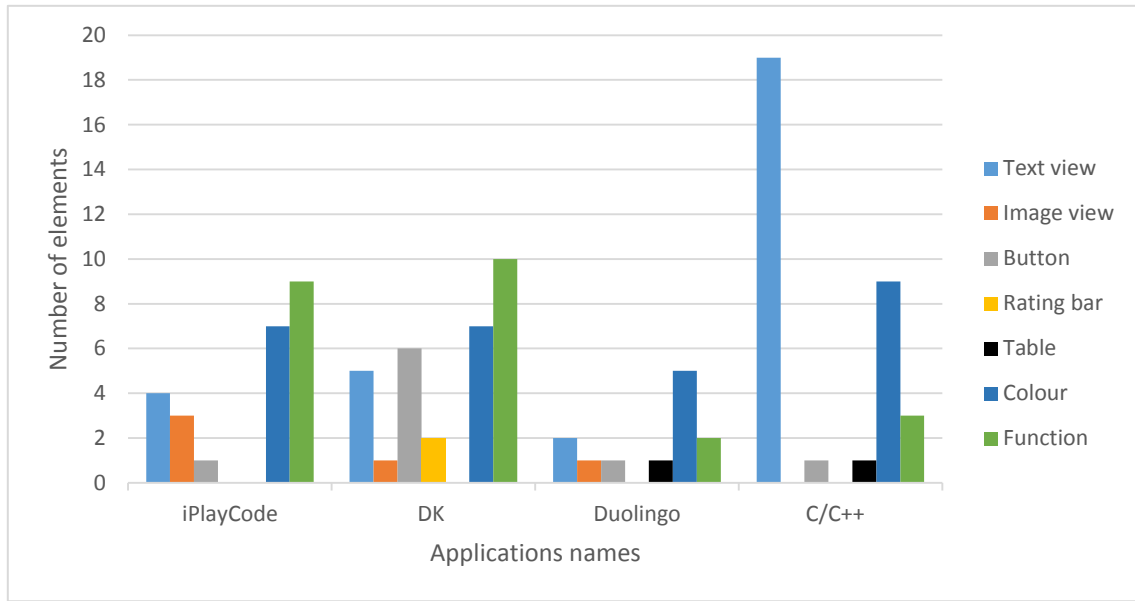


Figure 6.4: Comparison of UI elements on screen S_{info} .

Finally, Figure 6.4 revealed that the most text views (19) are used in screen S_{info} in the C/C++ app, while Duolingo has the fewest text views (2), followed by iPlayCode (4) and DK (5). The most image views (3) are recorded in iPlayCode; both DK and Duolingo have 1, and C/C++ has the lowest image view. The most buttons (6) are recorded by DK, while the rest of the apps each have one. Duolingo and C/C++ have 1 table in common compared to the remaining apps, and DK has 2 rating bars. However, the number of colour choices is different from one app to another: C/C++ uses the most colours (9), followed by iPlayCode (7), DK (7) and Duolingo (5). Lastly, the number of features depends on the functionality design of each app such that DK has the highest features (10), with 9 in iPlayCode, 3 in C/C++ and 2 in Duolingo, which has the lowest. S_s , S_i , S_g , S_{info} screenshots for each application are shown in Appendix A.

6.2.3 Case Study 2: SC Comparison Design

Social media applications are designed to facilitate interaction and communication not only between friends and acquaintances but also between customers and businesses. The SC hierarchical structure design is compared with those of other mobile apps in Figure D.2 in Appendix D. In the following part, SC is compared with the popular mobile apps Google+, Facebook, and Gumtree. For each application, the structured design is comprised of four major model screens (S_s , S_i , S_g and S_{info}), together with their subscript labels ($S_1 \dots, S_n$).

The hierarchical structure design is what primarily sets SC apart from the other three applications, especially Gumtree; it displays a higher degree of compartmentalisation with regard to design complexity. For example, by contrast to Gumtree, which has a depth structure that consists of only ten levels ($S_s - S_{I2I322I}$) and a breadth structure of only five levels ($S_I - S_5$), SC possesses depth and breadth structures of ten and fifteen levels, respectively ($S_s - S_{I4-7}$ and $S_I - S_{I5}$), which afford it a structured design that is easy to navigate. Meanwhile, Google has a six-level depth structure ($S_s - S_{5I2I}$) and five-level breadth structure ($S_I - S_5$), and Facebook has a thirteen-level depth structure ($S_s - S_M$) and five-level breadth structure ($S_I - S_5$), which means that Facebook has the most complex structure among the four applications.

6.2.4 SC Screen Comparison

To ensure that elements were used adequately in the creation of SC, the elements in other social media mobile apps were closely examined. To identify design differences between the mobile applications under investigation, a comparison has been conducted between the SC screens S_s , S_i , S_g and S_{info} , and the screens of the other three mobile applications.

In terms of screen S_s for the SC application, Figure 6.5 shows that Gumtree was associated with the highest number of image views (2) and the highest number of text views (7), while SC, Google+ and Facebook all had fewer image views (1) and fewer text views (2). By contrast, Facebook had the highest number of buttons (4), followed by Gumtree (3), and SC and Google+ with 1 each. Furthermore, Gumtree also differed from the other applications by possessing 2 collection views, 1 search bar and 1 tab bar. Google+ used the most colours (6), followed by Gumtree (5), SC (3), and Facebook (2). Gumtree used the highest number of function features (12), followed by Facebook (6) and SC and Google+ with 1 each.

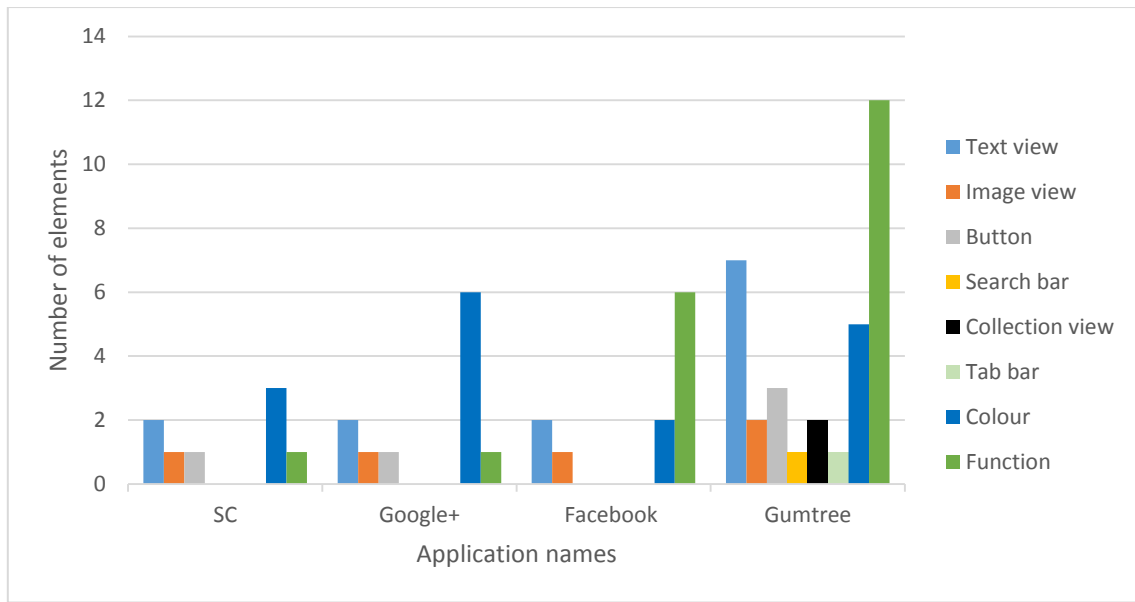


Figure 6.5 Comparison of UI elements on screen S_s (single screen) between social media apps.

Regarding screen S_i , Figure 6.6 revealed that the highest number of image views was achieved by SC and Google+ with 3 each, followed by Guntree (2), and Facebook (1).

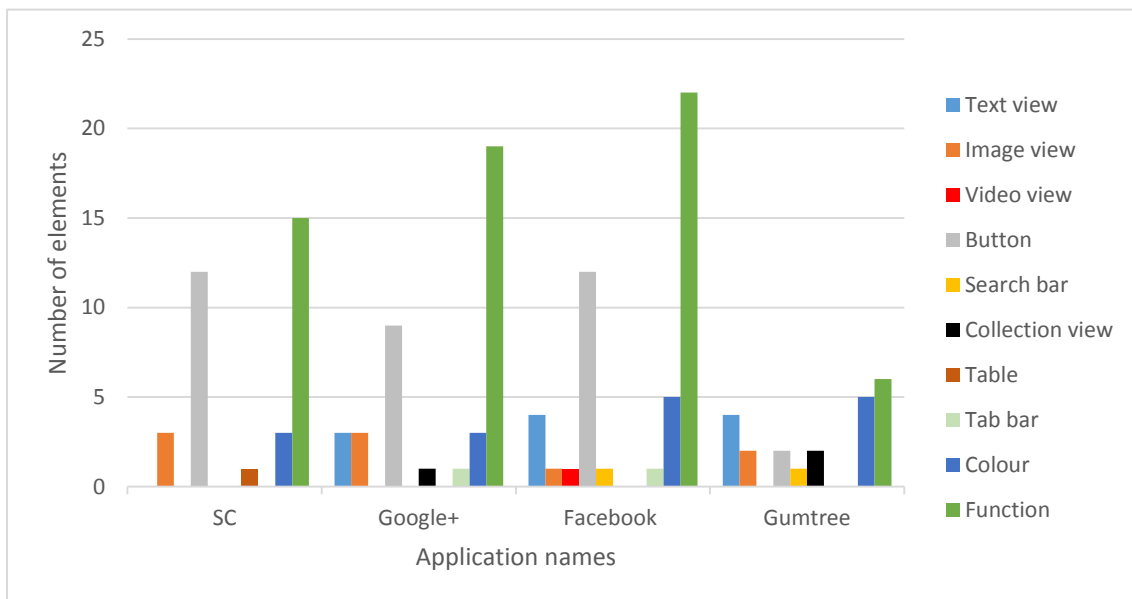


Figure 6.6: Comparison of UI elements on screen S_i .

The highest number of text views was recorded by Facebook and Guntree with 4 each, followed by Google+ (3), while SC did not record a single text view. By contrast, SC and Facebook had the highest number of buttons with 12 each, followed by Google+ (9), and Guntree (2). Google+ and Facebook each presented 1 tab bar, while SC had 1 table, unlike

the other three. Different from the others as well, Gumtree had 2 collection views and Facebook had 1 video view. Moreover, 1 search bar was presented by Facebook and Gumtree, but none by SC and Google+. Facebook and Gumtree also used a higher number of colours (5) compared to SC and Google+ (3). Facebook had the greatest number of used function features (22), followed by Google+ with 19, SC with 15, and Gumtree with 6.

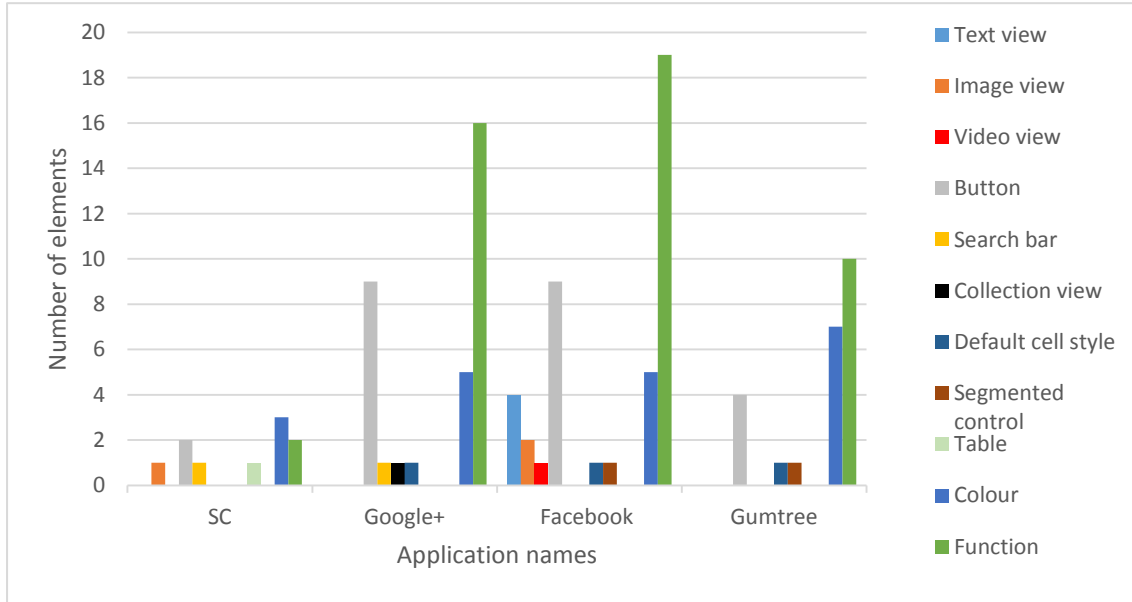


Figure 6.7: Comparison of UI elements on screen S_g .

Figure 6.7 shows the comparison of the elements of screen S_g between the SC, Google+, Facebook and Gumtree applications. Facebook had the highest number of text views (4). Regarding image views, Facebook had one more image view compared to SC, while Google+ and Gumtree had none. One default cell style was exhibited by each of the applications except SC, which had none. Similarly, 1 search bar was exhibited by SC and Google, but none by Facebook and Gumtree. Google+ differed from the other applications by possessing 1 collection view, while Facebook and Gumtree differed from the other two by possessing 1 segmented control, and Facebook differed from the rest due to its 1 video view. Moreover, Gumtree had the highest number of used colours (7), followed by Google+ and Facebook with 5 each and SC with 3. Facebook had the highest number of function features (19), followed by Google+ (16), Gumtree (10), and SC (2).

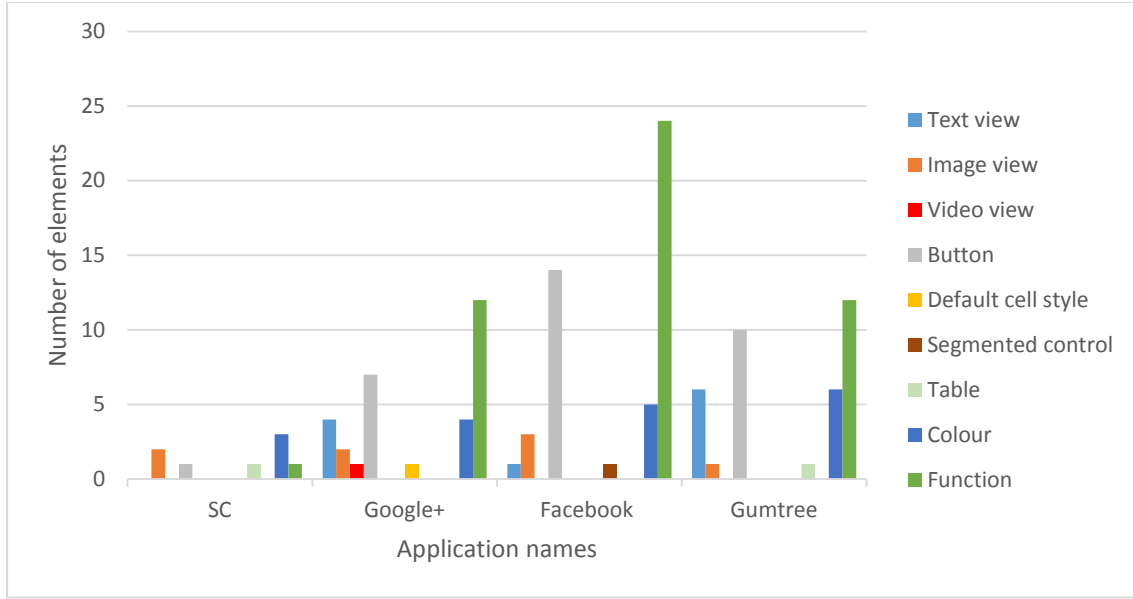


Figure 6.8: Comparison of UI elements on screen S_{info} .

Last but not least, regarding the S_{info} screen, Figure 6.8 illustrated that Gumtree had the highest number of text views (6), followed by Google+ (1), Facebook (1), and SC (0). On the other hand, Gumtree had the fewest image views (1), whereas Facebook had the most image views, with one additional view compared to SC and Google+. SC and Gumtree differed from the other two applications by possessing 1 table, whereas Facebook differed from the others by possessing 1 video view, 1 default cell style, and 1 segmented control. Facebook had the most buttons (14), followed by Gumtree (10), Google+ (7), and SC (1). Furthermore, the maximal and minimal numbers of colours were used by Gumtree (6) and SC (3), respectively. Facebook used the highest number of function features (24), followed by Google+ and Gumtree with 12 each and SC with 1. Appendix A provides further details about the screenshots S_s , S_i , S_g , and S_{info} associated with each of the applications.

6.3 The Comparison of iPlayCode and SC with Other Application Elements

The comparative analysis of iPlayCode and SC with other similar mobile apps is presented in this section. As was previously mentioned (see section 3.3.1.3), the elements are divided into six categories: control, content, vision, navigation, colour and function elements. The aim of this comparative analysis is to get an idea of the simplicity of the proposed model and the designed mobile applications, namely, iPlayCode and SC.

6.3.1 ANOVA Test of Single and Multiple-screens for iPlayCode and SC

The testing technique involves numbering elements on the screen. Various types of data (e.g., number of images and buttons) have been gathered from eight application designs. Among these designs, the single-screen (S_s , S_i , S_g , and S_{info}) and multiple-screen designs (all screens) were subjected to comparative analysis based on the one-way ANOVA test with the purpose of identifying any potential differences (see Appendix F). The outcomes of the counts-based assessment for single and multiple-screens are provided in Appendix E.

Table 6:1: Significance of variance between single screens for iPlayCode and other applications.

Model screens		iPlayCode, DK, Duolingo, C/C++		
		df	F	Sig
S_s	Between Groups	3	0.471	0.706
	With Groups	20		
	Total	23		
S_i	Between Groups	3	3.152	0.048
	With Groups	20		
	Total	23		
S_g	Between Groups	3	0.242	0.866
	With Groups	20		
	Total	23		
S_{info}	Between Groups	3	0.748	0.536
	With Groups	20		
	Total	23		

6.3.1.1 Outcomes of Case Study 1

The differences identified between the interface designs were subjected to significance testing. The four single screens S_s , S_i , S_g , and S_{info} were included as the independent variables of distinct interface designs. The p-values of S_s (0.706), S_g (0.866), and S_{info} (0.536) were all greater than 0.05 and the differences were insignificant in these screens. On the other hand, the p-value of S_i (0.048) is lower than 0.05, confirming that the differences were significant in S_i among iPlayCode, DK, Duolingo and C/C++ (see Table F.2 in Appendix F).

To determine how significantly S_i differed among the abovementioned applications, a paired t-test was conducted, the findings of which are provided in (see Tables F.5 to F.10 in Appendix F). S_i was observed to be significantly different between iPlayCode and DK (0.024), but not between iPlayCode and Duolingo ($p=0.089$); iPlayCode and C/C++ ($p=0.728$); DK and Duolingo ($p=0.120$); DK and C/C++ ($p=0.103$); or Duolingo and C/C++

($p=0.094$). It could thus be concluded that the four applications did not differ significantly in terms of the single screens.

An f-test was also conducted to determine how significantly iPlayCode and DK differed in terms of screen S_i . The two applications were established to be insignificantly different regarding screen S_i (0.10) because the p-value obtained was greater than 0.05 (see Table F.11 in Appendix F).

The differences between the interface designs for iPlayCode, DK, Duolingo and C/C++ with regard to multiple-screens were assessed through another ANOVA test. Differences were confirmed to be insignificant, as the obtained p-value of 0.423 was greater than 0.05. (see Table F.12 in Appendix F).

6.3.1.2 Outcomes of Case Study 2

The ANOVA test was applied to determine if SC differed significantly from Google+, Facebook, and Guntree with respect to the four single screens S_s , S_i , S_g , and S_{info} . The results are provided in Table 6.2.

Table 6.2: Significance of variance between single screens for SC and other applications.

Model screens		SC, Google+, Facebook, Guntree		
		df	F	Sig
S_s	Between Groups	3	1.408	0.270
	With Groups	20		
	Total	23		
S_i	Between Groups	3	0.450	0.720
	With Groups	20		
	Total	23		
S_g	Between Groups	3	0.626	0.607
	With Groups	20		
	Total	23		
S_{info}	Between Groups	3	0.901	0.458
	With Groups	20		
	Total	23		

According to Table 6.2, ANOVA test results revealed that no significant differences were discovered among the UI applications for S_s , S_i , S_g , and S_{info} (single screens), as the p-values were 0.270, 0.720, 0.607 and 0.458 > 0.05 respectively. (see Tables G.1 to G.4 in Appendix G for more details).

Likewise, the ANOVA test result shows that the UI applications did not differ significantly for multiple-screens either, as the p-value was $0.699 > 0.05$. (see Table G.5 in Appendix G).

6.4 Questionnaire Analysis and Results

6.4.1 Sample Distribution

Table 6.3 shows that the participants who provided the user experience data related to the usability testing were between 18 and 49 years of age; the age group of 18 to 34 are 35.5%; in the age group of 35 to 49 are 64.5%; 69.40% of them were males, and 30.60% were females. The educational level of all the participants was undergraduate level or higher; 9.70% were undergraduates and 90.30% were post-graduates.

Table 6.3: Sample distribution of usability-interface data on the iOS mobile apps.

Item	Group	Number of participants	Percentage
Gender	Male	43	69.4%
	Female	19	30.6%
Age Group	18-34	22	35.5%
	35-49	40	64.5%
Educational Degree	Under Graduate	6	9.7%
	Post Graduate	56	90.3%
Experience in using smart phone	1-5	5	8.1%
	6-11	46	74.2%
	More than 12	11	17.7%

The majority of the participants had a moderate amount of experience in terms of years of using smartphones. Approximately 74.2% of the users had between 6 and 11 years of experience in using smartphones; 17.7% of the participants had more than 12 years of experience in using smartphones, and they were considered to belong to the expert category; 8.1% of the participants had approximately 1 to 5 years of experience in using smartphones.

6.4.2 Reliability

A questionnaire is considered to possess reliability if it demonstrates consistent development over time (Bowles & Bordbar, 2007; Hillston & Kloul, 2006). The questionnaire used in this thesis was assessed to determine its reliability with respect to each of its four dimensions (i.e., usefulness, information quality, interface quality, and overall satisfaction) by applying Cronbach's alpha as a reliability measurement tool. Table 6.4 provides an overview of the

results obtained from the participants' answers following the completion of tasks using the recommended high-fidelity prototype and the existing applications.

Table 6:4: The reliability of iPlayCode and SC compared to other applications.

		iPlayCode	SC
Attribute	No of items	Alpha	
Usefulness	7	0.98	0.94
Information Quality	6	0.79	0.98
Interface Quality	4	0.92	0.71
Overall Satisfaction	2	0.92	0.93
Overall	19	0.90	0.89

The value of Cronbach's alpha was greater than 0.70 for both iPlayCode (0.79-0.98) and SC (0.71-0.98), which is satisfactory according to (Cimatti et al., 2011a). These results confirm that the scale is sufficiently reliable to be employed as a measure of usability.

6.4.3 Case Study 1: Results

To better understand the outcomes of the questionnaire, the mean of the user responses regarding each mobile app were calculated. Together with the mean of each response, the statistical range of each response was used to determine how the answers should be categorised (Table 6.5).

Table 6:5: Questionnaire scores for the iPlayCode, DK, Duolingo and C/C++ applications.

Item		iPlayCode	DK	Duolingo	C/C++
Usefulness		Mean			
1	Overall, I am satisfied with how easy it is to use this interface:	4.323	4.468	4.468	4.435
2	It was simple to use this interface:	4.452	4.452	4.452	4.452
3	I can effectively complete my work using this interface:	4.177	4.032	4.065	4.032
4	I am able to complete my work quickly using this interface:	4.145	4.113	4.000	4.016
5	I am able to efficiently complete my work using this interface:	4.242	4.000	4.000	4.048
6	I feel comfortable using this interface:	4.355	4.226	4.032	4.065
7	It was easy to learn to use this interface:	4.484	4.468	4.452	4.419
Total		4.311	4.251	4.210	4.210
Information quality					
1	Whenever I make a mistake using the interface, I recover easily:	4.435	4.387	4.387	4.145
2	The information provided with this interface is clear	4.387	4.403	4.452	4.355
3	It is easy to find the information I needed	4.258	4.113	4.177	4.032
4	The information provided for the interface is easy to understand	4.323	4.258	4.435	4.403
5	The information is effective in helping me complete the tasks and scenarios	4.371	4.081	4.177	4.371
6	The organization of information on the interface screens is clear:	4.484	4.435	4.435	4.468
Total		4.376	4.280	4.344	4.296
Interface quality					
1	The interface gives error messages that clearly tell me how to fix problems	4.403	4.371	4.355	4.032
2	The interface is pleasant	4.468	4.435	4.339	4.129
3	I like using the interface of this mobile application	4.371	4.452	4.435	4.323
4	This interface has all the functions and capabilities I expect it to have	4.113	4.048	4.097	4.000
Total		4.339	4.327	4.306	4.121
Overall satisfaction					
1	I believe I became productive quickly using this interface:	4.323	4.048	4.032	4.113
2	Overall, I am satisfied with this interface.	4.452	4.452	4.419	4.242
Total		4.388	4.250	4.226	4.178

6.4.3.1 Usefulness

Seven questions focused on the dimension of usefulness. For iPlayCode, DK, Duolingo, and C/C++. The mean values of questions relating to usefulness dimension were 4.311, 4.251, 4.210, and 4.210, respectively. Based on the categories presented in Table 6.5, there is on average a strong agreement with this dimension, indicating a high degree of usefulness of these applications. It can be noted that iPlayCode achieved the highest mean, suggesting that most participants agreed that this application had a high level of usefulness and had a positive attitude with regard to using it.

6.4.3.2 Information Quality

Six questions focused on the dimension of information quality. The mean values of questions relating to information quality dimension were 4.376, 4.280, 4.344, and 4.296 for iPlayCode, DK, Duolingo, and C/C++ respectively. iPlayCode again received the best feedback. These results indicate that the participants concurred that both the quality and ease of use of information were good.

6.4.3.3 Interface Quality

With the exception of C/C++, which obtained an ‘agree’ answer on the whole, all the other applications attained an overall ‘strongly agree’ answer with respect to the dimension of interface quality. The mean values of questions relating to interface quality dimension were 4.121 for C/C++, 4.306 for Duolingo, 4.327 for DK, and the highest mean of 4.339 for iPlayCode.

6.4.3.4 Overall Satisfaction

Table 6.5 presents the mean values obtained from the analysis of the data for the questions relating to overall satisfaction. The highest mean score, 4.388, was received by iPlayCode, followed by DK with 4.250, Duolingo with 4.226, and C/C++ with 4.178, indicating participants’ satisfaction and strong agreement with the first three applications and agreement with C/C++.

6.4.4 Case Study 2: Results

A statistical analysis of the responses to the questionnaire related to the rating of the mobile apps SC, Google+, Facebook and Gumtree is presented in Table 6.6.

Table 6:6: Questionnaire scores for the SC, Google+, Facebook and Gumtree applications.

Item		SC	Google+	Facebook	Gumtree
Usefulness		Mean			
1	Overall, I am satisfied with how easy it is to use this interface:	4.452	4.403	4.387	4.468
2	It was simple to use this interface:	4.371	4.419	4.339	4.403
3	I can effectively complete my work using this interface:	4.339	4.339	4.339	4.355
4	I am able to complete my work quickly using this interface:	4.339	4.323	4.323	4.339
5	I am able to efficiently complete my work using this interface:	4.274	4.274	4.258	4.290
6	I feel comfortable using this interface:	4.339	4.323	4.355	4.355
7	It was easy to learn to use this interface:	4.435	4.387	4.419	4.452
Total		4.364	4.353	4.346	4.380
Information quality					
1	Whenever I make a mistake using the interface, I recover easily:	4.403	4.387	4.355	4.387
2	The information provided with this interface is clear	4.403	4.403	4.419	4.419
3	It is easy to find the information I needed	4.290	4.306	4.306	4.290
4	The information provided for the interface is easy to understand	4.323	4.306	4.323	4.339
5	The information is effective in helping me complete the tasks and scenarios	4.242	4.242	4.274	4.274
6	The organization of information on the interface screens is clear:	4.435	4.468	4.435	4.484
Total		4.349	4.352	4.352	4.366
Interface quality					
1	The interface gives error messages that clearly tell me how to fix problems	4.403	4.387	4.371	4.468
2	The interface is pleasant	4.452	4.435	4.435	4.468
3	I like using the interface of this mobile application	4.419	4.387	4.258	4.468
4	This interface has all the functions and capabilities I expect it to have	4.323	4.290	4.339	4.339
Total		4.399	4.375	4.351	4.435
Overall satisfaction					
1	I believe I became productive quickly using this interface:	4.290	4.258	4.323	4.306
2	Overall, I am satisfied with this interface.	4.419	4.403	4.403	4.516
Total		4.355	4.331	4.363	4.411

Table 6.6 lists the mean values of the answers to questions related to the dimension of usefulness for Gumtree, SC, Google+, and Facebook. These values, which are based on the same Likert-scale range that was used for the previous questions, were 4.380, 4.364, 4.353, and 4.346, respectively. Thus, participants expressed strong agreement with the first three, but only agreement regarding the last application.

In terms of information quality, the highest and lowest means were associated with Gumtree (4.366) and SC (4.349), indicating strong agreement.

Regarding the interface quality, the highest mean was also achieved by Gumtree, with 4.435; the lowest mean, 4.351, was associated with Facebook.

On the whole, participants were most satisfied with Gumtree, which received a mean score of 4.411, and they were least satisfied with Google+, which received a mean score of 4.331.

In terms of the Likert-scale range, each of the four application dimensions of usefulness, information quality, interface quality, and overall satisfaction were strongly agreed with by the participants.

6.5 One-way ANOVA Test Results of Both Studies

The analysis requirement indicates that one-way ANOVA is the most appropriate data analysis technique for this thesis (Foster, 2001). This test was used to identify significant differences among the mobile apps in terms of usefulness, information quality, interface quality and overall satisfaction of the users.

Table 6:7: Significance of variance between case studies and existing applications.

Attribute		iPlayCode, DK, Duolingo, C/C++			SC, Google+, Facebook, Gumtree		
		df	F	Sig	df	F	Sig
Usefulness	Between Groups	3	0.402	0.753	3	0.485	0.696
	With Groups	24			24		
	Total	27			27		
Information Quality	Between Groups	3	0.624	0.608	3	0.055	0.982
	With Groups	20			20		
	Total	23			23		
Interface Quality	Between Groups	3	1.638	0.233	3	1.279	0.326
	With Groups	12			12		
	Total	15			15		
Overall Satisfaction	Between Groups	3	0.374	0.777	3	0.207	0.886
	With Groups	4			4		
	Total	7			7		

The significance values (Sig.) based on the ANOVA test for the iPlayCode, DK, Duolingo and C/C++ apps (Table 6.7) are greater than 0.05 for all attributes, indicating that there is no statistically significant difference in usefulness, information quality, interface quality or overall satisfaction between iPlayCode and the other applications (see section A in Appendix G). Similarly, the significance values obtained using the one-way ANOVA test for the SC, Google+, Facebook and Gumtree applications are greater than 0.05 for all attributes (see section B in Appendix G). Hence, no significant difference was found among the SC, Google+, Facebook and Gumtree apps in terms of usefulness, information quality, interface quality or overall satisfaction.

6.6 Summary

The main achievement of this chapter was the implementation of the hierarchical structured design model in the case of two mobile apps. The mobile user interfaces of each of these apps were described and compared to those of other existing mobile apps: the proposed iPlayCode application was compared to DK, Duolingo, C/C++, and the proposed SC was compared to Google+, Facebook and Gumtree. In summary, it can be noted that iPlayCode and SC has an easily navigated structured design, in contrast to other apps, and the structured design of the model was validated by those of other existing mobile apps for all four model screens of S_s , S_i , S_g , and S_{info} .

Furthermore, each mobile app screen was compared and analysed in terms of the number of elements used in the design, such as the total number of image views, text views, buttons and other features. Because the proposed hierarchical structured design supported the model design used for all the apps, all of the apps had most used elements in common. On the other hand, some of the elements differed according to the model screens. For instance, in iPlayCode, the text field was used, whereas there was no text field in the S_s screen of DK, Duolingo or C/C++. In addition, the table element is used in SC and Gumtree, and other elements such as collection views, tab bars, and default cell style are present in all the other apps except SC.

In the first study, the number of available elements in each app differed according to the apps' screens. For example, the highest number of elements in the S_s screen is used in iPlayCode. Duolingo presents the highest number of elements in the S_i screen, and the S_g screen uses the highest number of elements in the iPlayCode app, followed by DK. Likewise, the highest number of used elements in the S_{info} screen is found in the C/C++ app, followed by the DK app.

On the other hand, in the second study, the highest number of elements for S_s was noted in Gumtree, whereas Facebook uses the highest number of elements in S_i , S_g and S_{info} .

The statistical significance of the observed differences in the application interface designs of the independent screens was assessed via the ANOVA test, the t-test and the f-test. The findings revealed that neither single screens (S_s , S_i , S_g , and S_{info}) nor multiple-screens differed significantly in either of the two case studies.

Furthermore, to determine how satisfied users were with the applications' usability, the data derived from the questionnaire were also analysed in this chapter.

The two case studies of iPlayCode and SC did not differ significantly from the other applications under consideration, as shown by the results of the one-way ANOVA test performed on the independent screens of usefulness, information quality, interface quality, and overall satisfaction. Moreover, the assessment of the usability of the application interfaces was conducted based on the qualitative data approach, which took into account the participants' views. On the whole, the research produced two major findings, namely, that navigation, use and learning of iPlayCode and SC did not present any difficulties and that some usability issues existed, as indicated by the results of the qualitative assessment.

Based on the results obtained in the two case studies, the suggested model was confirmed to have less complexity than existing applications with respect to its structural design and number of elements and good usability as a mobile interface design.

Chapter 7: Evaluation of Proposed Approach

7.1 Introduction

Evaluating the status of the two aforementioned UI case studies (Chapter 5) from the viewpoint of the end-users

Assessment of the proposed formal model (FMMUID) that was presented in Chapter 4 is the focus of this thesis. Since this is a new model, it is important to determine how flexible and usable it is. To this end, the assessment process is divided into two phases: the first phase is model validation based on the results derived from the structure design analysis and the count-based method (single and multiple-screens), and the second phase is usability measurement for the two case studies based on data from the questionnaire.

For each of the two case studies, UI complexity was examined using specific research questions and assessment criteria. To enable comparison between the case studies and the existing applications, analysis of the hierarchical structure designs was conducted. In this analysis, particular attention was paid to the presence or absence of the key model screens (S_s , S_i , S_g and S_{info}) in the structure designs, to demonstrate the new model in relation to the existing applications. The screen elements were examined as well to determine how logical and effectively structured the case studies were by comparison to the existing applications. Chapter 5 provided a comprehensive description of the two case studies, which were developed in a way that addressed the research questions in section 1.4.

The structure of the remainder of this chapter is as follows. The assessment of UI complexity is the focus of section 7.2. Single and multiple-screens are assessed in section 7.3. Usability assessment and model validation are addressed in section 7.4 and section 7.5, respectively. An overview of the chapter is provided in section 7.6.

7.2 Complexity

A study by Kong et al. (2009) describes complexity as being dependent on the interactions among the elements of an object, suggesting the existence of myriad associations and interactions among them. When trying to determine complex interactions among multiple elements of an object, it is possible to think of complexity as a measure of difficulty. Given that understanding is derived from various factors and assessed subjectively, the nature of

complexity is not easy to measure or describe. The UI complexity can therefore be thought of as a link to the perspective of the user in terms of the information display and their understanding of the device screen. The productivity of the system is further associated with the complexity of the outcome support and clarification. Navigation (the number of screens determines the depth and breadth), functions, and colours afford the UI designs their complexity.

7.2.1 Navigation

With a greater degree of hierarchy, navigating problems such as selecting the wrong pathway or getting lost become more apparent. A multi-level hierarchical structure necessitates user recollection or identification of a pathway from the current position to the desired destination (Ribeiro et al., 2005). There are several benefits to the adoption of such a multi-level approach. When the screen space allocations are insufficient to meet the needs of the number of components, a degree of depth is required to minimise on-screen overcrowding. Conversely, the use of such complexity increases the error rate and the number of screens required. The effect of using fewer screens on an interface structure is subsequent overcrowding on associated screens. Limitations of navigability were addressed in the proposed model by using the structural depth in the iPlayCode app because the number of screens was small (six), as shown in Figure D.1 in Appendix D, while the structure of the SC model was based on both depth and breadth (balance between depth and breadth) because there were many screens (44), as illustrated in Figure D.2 in Appendix D.

The fewer elements a structure has, the less complex it is. This means that the complexity of a hierarchy structure is amenable to quantification. The number of screens and the number of elements are the two main factors that can increase the complexity of a structure design.

To investigate the proposed model, the researcher applies the proposed model to six existing mobile applications to analyse their design. The results show that all selected mobile applications validate the new model (FMMUID) because they contain the main screens (S_s , S_i , S_g and S_{info}) in the model. In addition, based on these results, the proposed model (FMMUID) can be used as a method for mobile UI design, since it achieved simple and complex UI designs for different mobile applications (see Appendixes A, B and D).

With regard to case study 1, the screens were more logically organised in iPlayCode than in the other applications except that C/C++ had five depth levels with the same number of

screens (6), so it had two options in breadth (see Figure D.1 in Appendix D and Table E.5 in Appendix E). Furthermore, iPlayCode consisted of six screens with six levels in depth and one in breadth, indicating that it had lower complexity.

Regardless of common elements, e.g., text view, image view, and buttons being used, the number of elements on the various screens slightly differ between each case study and existing applications (Figure 6.1). In screen S_s , iPlayCode had the highest number of image views (3) which consisted of application name (iPlayCode), research group name (XDIR), and username icons (see Figure 5.2). However, according to the results obtained in Table E.5 in Appendix E, the Duolingo application has the highest number of elements (22) such as image views in the multiple-screens. This reveals that the distributions of image view elements were different between the screens. Overall, the iPlayCode application contains three in screen S_s and 11 in all screens. It can also be observed from the users' responses to the questionnaire (see Table 6.7) that the questions pertaining to the image views (questions 1, 6, 16, and 19) are ranked as 'strongly agree' by 86.46%, 87.10%, 89.36% and 89.00% of the users, respectively. iPlayCode also had an additional text field component to enhance the application instructiveness and engage users more effectively by prompting them to enter their names for display on the final result screen.

In screen S_i , the number of elements varies from one application to another. Although C/C++ had the lowest number of buttons (5), it had more text views (8) than iPlayCode, which had 1 text view and 7 buttons (see Figure 6.2). For this reason, C/C++ did not surpass iPlayCode because it possessed only two programming languages, whereas the latter had six programming languages and each of them comprised three levels.

Likewise, iPlayCode had the greatest numbers of image views (2) and buttons (4) on screen S_g (see Figure 6.3). When the application is deployed, the first image view relates to the total obtained "Score" in that level, to make the screen more aesthetic to attract the user. The second image view relates to the player's obtained medal (gold, silver or bronze) to increase the competition and challenge, and encourage the user to obtain a higher score (see Figure 5.2). In this situation, the type of medal depends on the score obtained. This screen also includes four buttons. The first button returns the player to the same level to play again. The second button returns the player to the level screen (S_l) to select another level. The third button helps the player return to the main screen (S_i) to choose another programming language. The fourth button takes the player to the final result screen (S_{info}) to display the

scores for all levels (see Figure 5.2). In this context, these buttons give more options for the players to play the game with a flexible UI, which helps them navigate the application effectively. According to Table E.5 in Appendix E, the highest number of used buttons for the Duolingo application is 99. The Duolingo application consists of 2 buttons in S_g (single screen), while it has 62 buttons in S_i . In contrast, the iPlayCode application consists of 4 in S_g and 25 in all screens. In other words, the number of buttons in each application depends on the requirements and functionality of the application and the designer's decisions on how to distribute the elements across the screens.

In terms of screen S_{info} , the fewest elements were displayed by Duolingo and iPlayCode. The main role of S_{info} is to display the final scores for all played levels. In contrast to Duolingo, iPlayCode consists of three image views (see Figure 6.4): one for each medal (bronze, silver and gold) in each level. There are several reasons for awarding medals. The visual incentive of a medal will trigger a user's motivation, and they are used alongside text to output a result. It is also imperative to reward continuous effort, as points are accumulated throughout the game until the score reaches a certain level. Then, users are given medals to mark personal milestones.

In summary, the iPlayCode application interface is not complex, because it has a less complex structure and the number of elements is acceptable, in contrast to the DK, Duolingo and C/C++ applications.

However, the structure design and interface elements in case study two (SC) are analysed separately from the application functions. Figures 6.5-6.8 and Table E.4 in Appendix E; Figure D.2 in Appendix D present the results of the obtained UI elements. Furthermore, the SC application has a more logical structure than other applications, due to balance between depth and breadth. Miller's recommendations have resulted in the use of breadth over depth in most cases, but it is also important to maintain a balance between the two (Miller, 1981). Hence, the SC structure provides a wider range of options (15) in terms of breadth and has an acceptable number of depth levels (12) to maintain the logical coherence, understandability and readability of the structures. The Google+ and Guntree applications consist of the lowest numbers of levels (6 and 10, respectively) because fewer screens are used than for SC and Facebook (see Table E.6 in Appendix E). In other words, whenever the number of screens increases, this UI design becomes more complex.

In terms of the number of elements in single screens, SC had the lowest number of components in the S_s screen compared with the other applications, which included 2 text views, 1 image view and 1 button (see Figures 5.4 and 6.5). However, SC and Google+ had the greatest number of image views (3). The image views in the SC application include the name of the application (SC) and the user's preferred selected images.

The numbers of view elements in screen S_i are shown in Figure 6.6, Google+ includes 2 image views and Gumtree includes 4, while Facebook includes 1 in S_i and 9 collection views in other screens. As a result, Google+, Facebook, and Gumtree use two types of elements to display images or pictures, while SC uses only the image view element; this is why the number of image views in the SC application is the highest.

With respect to the buttons, the SC application includes the highest number of buttons (12) in screen S_i and 29 in all the screens; the buttons in SC are used to navigate to other screens. As mentioned above, SC uses 15 options (breadth). In contrast, Google+, Facebook and Gumtree consist of 74, 96 and 53 buttons, respectively, in the completed applications (see Figure 6.6 and Table E.6 in Appendix E). In this context, the distribution of elements mainly depends on the designer's decisions, application screen functionality and application requirements.

In terms of screen S_g , the SC application consists of the lowest number of buttons (2). On the other hand, the number of table elements is 1 in S_g (single screen), while 14 tables have been used in all screens (multiple-screen). The reason for using the table element is to display more information for the user and help the user navigate the application easily using the scroll-up or scroll-down feature. However, the increased level of scrolling has resulted in degraded speed and retrieval performance levels (Larson & Czerwinski, 1998). Google+, Facebook and Gumtree use menu elements (9), (12) and (39) respectively (see Table E.6 in Appendix E). In addition, one search bar element is used in SC and Google+ (see Figure 6.7), while Facebook and Gumtree use the same number of elements in screen S_i (see Figure 6.6). This highlights that the distribution and use of elements depend on the designer's decisions and the required functionalities.

Regarding screen S_{info} , the highest number of image views in Facebook is 3, whereas the lowest is 1 in Gumtree, followed by 2 in SC and Google+. SC has the lowest number of buttons (1) (see Figure 6.8). However, the SC application consists of the lowest number of elements compared to the other applications for screen S_{info} .

In summary, based on these comparisons and results, the SC application has the fewest elements (e.g., text views, image views and buttons) in screens S_s , S_g , and S_{info} . Additionally, the Gumtree application includes fewer elements in screen S_i ; this is because more options have been included in the main screen for users to select in terms of breadth scale.

7.2.2 Function

Modern systems contain an inherent limitation in an excess of apparent functionality, resulting in a congested interface that is difficult to navigate. Essentially, this is not a limitation of too much functionality per second, but rather of an overly complex interface required by supplementary functionality (Dickinson et al., 2003). By minimising the function count, an interface becomes more user-friendly in terms of access to specific details and tasks in a clear and simple manner, where, on the other hand, expanded functionality increases complexity.

The iPlayCode application includes several extra functions compared with other applications in screens S_s and S_g . For example, in screen S_s , there are five functions (see Figure 6.1): the user number, select language, dialog error box for username (if empty), show the keyboard and navigate to the next screen. The user can change the user interface language before playing and the total number of users. The username can be input using the text box to display the player's name. However, in the Duolingo application, there is only one function available, namely, to change the UI language. Table E.5 in Appendix E illustrates that iPlayCode is less complex than other applications; this is due to the small number of functions (36) used in all screens, in contrast to DK (78) and Duolingo (109). The C/C++ application includes the lowest number of functions (21) due to the number of programming languages used in the application.

Screen S_i has the lowest number of functions with iPlayCode and C/C++ compared to other applications. For example, 7, 16, 65 and 7 functions are used in iPlayCode, DK, Duolingo and C/C++, respectively (see Figure 6.2)

Regarding screen S_g , iPlayCode includes 7 functions (see Figure 6.3): return to the same sub-function to increase the player's score in that sub-function or select another one; go back to the menu of screen levels to select another level; navigate to the final result screen (S_{info}); show achieved scores in the sub-function; display the medal (bronze, silver or gold) depending on the score; navigate to the main screen (S_i) to select another programming

language; finally, display a loading spinner function (see Figure 5.2) on the result screen, which makes the interface more attractive to the user and gives the user a natural sense of delight.

However, based on the results shown in Figure 6.3 and Table E.5 in Appendix E, iPlayCode has the fewest functions compared to the DK and Duolingo applications for screen S_g . The reason behind the high number of functions in screen S_g is to make iPlayCode more attractive and entertain the user. In other words, more features require more functions.

In terms of S_{info} , the number of functions in iPlayCode (9) is lower than in DK (10) and higher than in Duolingo (2) and C/C++ (3) (see Figure 6.4).

In the SC application, the number of functions is less than in existing applications in screens S_s , S_g and S_{info} . The SC and Google+ applications each include 1 function, while Facebook consists of 6 and Gumtree 12 for screen S_s (see Figure 6.5).

The numbers of functions in SC in screens S_g (2) and S_{info} (1) are smaller (see Figures 6.7 and 6.8). In contrast, Gumtree consists of fewer functions (6), followed by SC with 15 in screen S_i (see Figure 6.6). Table E.6 in Appendix E shows that SC includes the smallest number of available functions in all screens (multiple-screens) (35), while the number of functions in Gumtree is 60 (nearly double that in SC), and the highest number of functions is 141 in Google+.

The discussion above indicates that the SC application is less complex than existing applications.

7.2.3 Colour

The application of a quantity of colours of different types benefits the user experience through increasing comprehension and addressing complexity. Adverse or confusing outcomes can arise from the random use of colour, leading to negative user feedback and reduced productivity by preventing the user from concentrating on the task at hand. Additionally, some users may suffer from colour blindness. The other challenge of applying colours is to achieve a bold appearance that is attractive to the eye. The suboptimal use of colour can instigate a high degree of stress or lethargy, giving rise to confusing perceptions on the part of the user.

In the first study, to cope with the complexity of using colour in the user interface, rich colours were used to improve user interaction in the iPlayCode application (see Figure 5.2). However, the highest number of colours used in iPlayCode and C/C++ is 7 in screen S_s (see Figure 6.1). In iPlayCode, the used colours include very pale orange, medium slate blue, deep sky blue, white, light grey and black (see Figure 5.2). The highest number of used colours in the DK application is 12 in screen S_i (see Figure 6.2). Similarly, the DK application includes the highest number of colours (9) in screen S_g (see Figure 6.3). The highest number of colours in the C/C++ application is 9 in screen S_{info} (see Figure 6.4). The iPlayCode application includes the lowest numbers of colours in screens S_i , S_g and S_{info} ; however, for screen S_s , it includes the highest number. Dinulescu (2007) found that the use of more than seven screen colours resulted in a high level of screen complexity. Figures 6.1 to 6.4 show that the highest number of colours in all iPlayCode screens (S_s , S_i , S_g and S_{info}) is 7, which indicates that the iPlayCode application is less complex in terms of colours.

In the second study, S_s consisted of only three main colours in the SC application: green, white and blue; in S_i , S_g and S_{info} , black, white and grey were used to mitigate the colour blindness problem (Figure 5.4). The number of colours was 3 for the SC application. This is higher than that for Facebook, which only used 2, and lowest than those for Google+ (6) and Gumtree (5) in screen S_s (Figure 6.5). However, SC and Google consisted of the lowest number of colours in screen S_i (3) (see Figure 6.6). Likewise, screens S_g and S_{info} for the SC application had the lowest number of colours (3) (see Figures 6.7 and 6.8). Additionally, the total number of colours used in SC was 5, which is lower than for the Google+ (7), Facebook (6) and Gumtree (8) applications (see Table E.6 in Appendix E).

Essentially, four primary origins of UI complexity have been described: 1) the number of screens, 2) the number of elements on each screen, 3) the number of functions on each screen, and 4) the number of colours on each a screen. The interface complexity and comprehensibility may thus be heightened by increasing the amount of these elements. However, there are no minimum or maximum numbers of elements, colours and functions on each screen because our results revealed that there are few elements, colours and functions on some screens and many on other screens. For example, the number of text views in DK is 1 on screen S_s and 11 on screen S_i (see Figures 6.1 to 6.8). The numbers of elements depend on the designer's decisions and the application requirements.

7.3 Assessment of Single and Multiple-screens in the Two Case Studies

Using a count-based method, data were collected by simply counting the elements that were present on the screen. Figures 6.1 to 6.8 present the obtained results from the count-based analysis. The author analysed eight mobile user interface designs to detect their elements or their numbers of components. This included images, buttons, colours, and functions. The screenshots that were evaluated are presented in Appendixes A and B.

As was expected, the on-screen elements only slightly differed between existing applications and the case studies (see Figure 6.1 to 6.8). Most of the elements were present in all screens (such as text views, image views, and buttons), but some differed (rating bar, table, default cell style, collection view, segmented control and video view) among the screens according to screen functionality and application requirements.

For the iPlayCode and SC applications, several important findings were obtained. Among the eight interface designs that the author examined, the single-screen and multiple-screen were the best. The author performed a one-way ANOVA test to determine whether the user interfaces had significant differences. A t-test and f-test were also used for pairwise comparison.

In this thesis, four single screens were examined, which are main factors in the proposed model (FMMUID). These screens are S_s , S_i , S_g and S_{info} (see Equation 4.1).

In study 1, we compared iPlayCode with three existing applications using both single- and multiple-screen comparisons to determine the differences among them using the ANOVA test.

In study 1, after the extraction of the numbers of elements, functions and colours for each app, whether the average numbers of these features differed between UI applications was assessed. (see Table E.2). Table 6.1 shows the comparative analysis based on the one-way ANOVA test with the purpose of identifying any potential differences between the iPlayCode and DK, Duolingo and C/C++ user interface designs. For single screen S_s , using a one-way ANOVA test, the differences between interface applications were not significant, with $0.706 > 0.05$. For screen S_i , again using a one-way ANOVA test, the results were significant, with $0.048 < 0.05$. A paired t-test was conducted to determine which applications were causing the difference; the results revealed that the difference was between iPlayCode and DK, with $0.024 < 0.05$. To determine which application was causing the difference, an f-test was

conducted. The f-test results showed that there was no a significant difference between iPlayCode and DK, with $0.10 > 0.05$.

Regarding screens S_g and S_{info} , the ANOVA test results indicate no significant difference between the UI designs, with 0.866 and $0.536 > 0.05$, respectively.

In study 2, Table 6.2 shows the comparison between SC and three existing applications (Google+, Facebook and Gumtree). The ANOVA test results reveal that there were no significant differences among them for single screens. The p-values for S_s , S_b , S_g and S_{info} were 0.270 , 0.720 , 0.607 and $0.458 > 0.05$, respectively. It could thus be concluded that the four applications did not differ significantly in terms of the single screens.

Furthermore, a one-way ANOVA test was carried out again for two case studies in terms of multiple-screens. These results also indicated that there were no significant differences between the case studies and existing applications, with 0.423 and $0.699 > 0.05$ for the iPlayCode and SC applications, respectively (see Tables F.12 in Appendix F and G.5 in Appendix G).

7.4 Assessment of Usability

To be accepted by users, mobile application UIs must demonstrate, above all, usability. In this work, the questionnaire tool was employed to assess usability by measuring different dimensions of this quality. All eight UI applications, comprising both case studies and existing applications, were tested, and a series of questions was answered by the research participants.

Table 6:5 and Table 6:6 list the means associated with the participants' answers for case study 1 and case study 2, respectively.

In the context of case study 1, the highest mean score for the dimension of usefulness (4.311) was attained by iPlayCode, and 86.0% of the participants expressed satisfaction with the iPlayCode UI. In terms of the dimensions of information quality and interface quality, iPlayCode also achieved the highest average scores of 4.376 and 4.339 , respectively; approximately 87.5% of the participants considered the information quality of the iPlayCode UI to be satisfactory, and over 86.0% considered the quality of the iPlayCode UI to be better than that of the other applications. iPlayCode achieved the highest mean score in overall

satisfaction as well (4.388); approximately 87.8% of participants were satisfied with the iPlayCode UI.

In the context of the second case study, all UI applications had similar mean scores with respect to the dimension of usefulness. Approximately 87.0% of participants considered Gumtree and SC to be useful, with mean scores of 4.380 and 4.364, respectively. Google+ (4.353) and Facebook (4.346) were considered useful by the same proportion of participants but received slightly lower mean scores (4.353 and 4.346, respectively). In regard to information quality, approximately 87.0% of the participants stated that Gumtree and SC had good information quality, with average scores of 4.366 and 4.349, respectively. In terms of the dimension of interface quality, the highest average (4.435) was recorded by Gumtree as well, and over 88.0% of the participants found the interface quality of Gumtree to be good. In second place was SC with an average of 4.399; approximately 88.0% of the participants were satisfied with the quality of the SC UI. In terms of overall satisfaction, Gumtree achieved the highest average (4.411; ~88.0%), followed by Facebook (4.363; 87.0%), SC (4.355; ~87.0%), and Google+ (4.331; >86.0%).

In terms of the questionnaire findings, iPlayCode was not found to differ significantly from the other applications (i.e., DK, Duolingo, and C/C++) with respect to the dimensions of usefulness ($0.753 > 0.05$), information quality ($0.608 > 0.05$), interface quality ($0.233 > 0.05$), or overall satisfaction ($0.777 > 0.05$). Likewise, these dimensions did not exhibit significant differences between SC and the other three applications of Google+, Facebook, and Gumtree ($0.696, 0.982, 0.326$ and $0.886 > 0.05$) (see Table 6.9).

7.5 Validation

The case studies provided strong support for the idea that mobile UIs with good usability can be created with the help of the FMMUID. The results of the user testing were consistent with the analysis results. Furthermore, based on the earlier sections of this work, the hypotheses regarding the lack of difference between the new and the conventional UI designs in terms of usability were validated based on the fact that the p-value obtained in the test statistics was higher than 0.05, indicating no statistical significance.

7.6 Summary and Assessment of Research Questions

The research questions that were outlined in section 1.4 were addressed, as confirmed by the assessment of the research process undertaken in this thesis.

A two-phase process was conducted to assess the effectiveness of the model approach. The first phase focused on the design rationale and on determination of differences in the numbers of elements for different UIs, and the second phase was concerned with user testing of the interfaces and completion of the questionnaire.

Assessment of the new FMMUID approach involved addressing two research questions. The first question sought to determine the UI with the most structurally logical design, the least complexity, and the fewest screen. This question was addressed based on two case studies of application interfaces of different complexity, namely, the simple iPlayCode application and the complex SC application, which use six and forty-four screens, respectively. A comparative analysis of these two case studies and other existing applications was conducted; the results revealed that the well-defined structure, lower complexity and acceptable number of elements on screens made the iPlayCode and SC UI designs effective with regard to the key dimensions assessed. Furthermore, the two case studies did not differ significantly from the other applications as indicated by the outcomes of the ANOVA test, the t-test and the f-test for single and multiple-screens. Therefore, in response to the first question, it could be affirmed that the iPlayCode and the SC are the UIs with the most structurally logical design, limited complexity, and fewest elements on screens. Accordingly, hypotheses H1 and H2 are validated.

The second question was concerned with determining how usable the iPlayCode and SC applications were compared to the other conventional applications with regard to the dimensions of usefulness, information quality, interface quality, and overall satisfaction. On the whole, the results showed that the two UIs were considered to possess good usability. The participants expressed either strong agreement or agreement with the usability properties of the two UIs. Furthermore, the majority of the participants were of the opinion that the iPlayCode and SC applications were easy to use. Therefore, on the basis of the results obtained in response to the second question, it could be affirmed that the proposed UIs did indeed possess good usability; accordingly, hypothesis H3 is validated.

To summarise, the results of the two case studies showed that the new model approach performed well with respect to every aspect considered, demonstrating that it possesses sufficient flexibility and efficiency to be employed in the design and development of mobile UIs.

Chapter 8: Conclusions

In this chapter, the major findings of the research are summarised, and the conclusions that can be drawn from them are presented. Furthermore, the chapter outlines the manner in which the established research aims and objectives have been accomplished and indicates the key contributions made by the research. Additionally, the chapter addresses the research shortcomings and puts forth recommendations for future lines of inquiry.

The development of the FMMUID has enabled successful achievement of the research aim and objectives (section 1.5), as illustrated by the examples and evidence put forward. Sufficient detail has been provided to permit the replication of the approaches employed in this thesis by other researchers for the purposes of mobile UI design and additional investigation. Test results compared with research objectives as set out in section 1.5.

1- Develop a novel approach to UI design.

Chapter 2 describes a number of approaches to interface design, most of which suffer from problems with over-complexity in design, software development and difficulty of providing support and maintenance. They are generally inflexible and therefore difficult to adapt for different applications. In this study, the principle objective is to set out a more creative approach to mobile user interface design using set theory to group elements and components and ranking then according to specific criteria.

Our design methodology, as set out in Chapter 4, meets this principle aim. Another driver behind this project has been the perceived need to build a robust framework for software specification in the development of UI programs.

Set theory in mathematics provides a well established and robust set of tools and a language that can be used in descriptions of the elements of which software is built. The software elements are ranked in a hierarchy that supports effective transition from theory to the practical development and application of the software and permits a holistic description of the system and its functionality. This is the FFMUID framework set out in this paper, which can be used to effectively express, and allow understanding of, all elements in UI software including the most abstract.

The FMMUID framework is capable of describing software of varying levels of intricacy, and provide detailed descriptions of the design elements and their ranking in interface

software. Examples, or prototypes, were chosen to display the essential elements and features of mobile UI software, and two of these presented as practical evaluations.

2- Validate the application of the new approach on case studies (iPlayCode and SC).

The FFMUID model (Chapter 4) has been applied in two practical examples, using iPlayCode and SC for this purpose. In Chapter 5 further examples are presented, using interfaces of varying levels of intricacy, which support the flexibility and usability of the proposed model.

3- Analysing the design structure and UI elements (for case studies and existing applications) in order to measure the complexity and validate the proposed approach.

This objective has been achieved by analysing the FFMUID using case studies and existing applications. The purpose of this analysis to validate and measure the complexity. Comparative analysis was conducted for iPlayCode and SC parameters (S_s , S_i , S_g and S_{info}) and those of other applications as a way of validating the model. To measure the complexity of the UI, a number of metrics were employed; these were associated with hierarchical structure (section 6.2) and number of components, functions and colours (section 6.3).

4- Evaluating the status of the two aforementioned UI case studies from the viewpoint of the end-users.

In Chapter 7 the FFMUID model is assessed in a variety of different applications. It is clear from the results of our tests and the range of environments in which FFMUID can be used successfully that this new approach has real pragmatic applicability. The tests were carried out from the point of view of hands on users of the interfaces and the results are presented in section 7.4.

In data obtained by CSUQ, the tested applications (i.e. iPlayCode and SC) produced very similar results to other software in the evaluation criteria of overall user satisfaction with the software, users' assessment of the interface, the information contained in the system and its functionality.

The thesis has focused on the relevant results obtained and the methods that were created based on those results and the experiments that were performed.

8.1 Result Overview

The research described in this thesis sought to determine whether it was possible to incorporate UI design into a formal model. To this end, a new approach was proposed, and its

advantages were comprehensively discussed. The integration of the UI design process into the suggested method was addressed in detail in Chapter 5.

The FMMUID model has been the focus of the thesis; it was proposed as a novel approach to achieving a formal characterisation of the UI design. To demonstrate how this approach can facilitate the integration of UI design and a formal model and how it can benefit the process of UI design, case study examples have been used. To enable UI characterisation, hierarchical structure and set theory language constituted the main components of the formal model. The first phase of the design process involved using the FMMUID to achieve the development of prototypes. The FMMUID has many advantages that link the prototypes to the formal specifications, thus sparing designers the necessity of dealing with complicated formal terminology when ensuring design consistency (Chapter 4). In addition to eliminating the need for expert knowledge about the underlying system, the model reveals the design decisions that have already been made and outlines the designers' comprehension regarding the significance of the prototypes and designs. The accurate model obtained in this way makes it easier for UI designers to communicate with other system development team members (e.g., formal experts, other designers, etc.). Thus, the research affords both a theoretical foundation and a practical method for the design process. The feasibility of the proposed idea was assessed through the development and assessment of two prototypes (Chapter 5).

Both scientific and human assessment methods were employed in the sixth chapter to measure the research contributions. The scientific assessment involved application of the FMMUID to develop two case studies, namely, iPlayCode and SC, followed by comparative analysis of their parameters (S_s , S_i , S_g and S_{info}) and those of other applications as a way of validating the model. Specifically, iPlayCode was compared with DK, Duolingo and C/C++, and SC was compared with Google+, Facebook, and Guntree. To measure the complexity of the UI, a number of metrics were employed; these were associated with hierarchical structure (section 6.2) and number of features, such as components, functions and colours (section 6.3). In this thesis, the results of the assessment confirmed that the proposed model was generally more efficient and flexible compared to existing mobile applications that were functionally similar to the two case studies. Furthermore, with regard to most relevant criteria, the UI design of iPlayCode and SC generally performed better than existing mobile applications, owing to their straightforward hierarchical structure, limited complexity and the acceptable number of elements they displayed on screen (see Figures 6.1 to 6.8; Tables E.1, E.3, E.5 and

E.6 in Appendix E; Figures D.1 and D.2 in Appendix D). The discrepancies between the suggested UIs and existing UI applications were explored through the one-way ANOVA test, the paired t-test and the f-test (section 6.3.1). The p-values indicated by the one-way ANOVA test, the t-test and the f-test were greater than 0.05, indicating that the four parameters of S_s , S_i , S_g , and S_{info} (single screens) and multiple-screens did not differ significantly (see sections 6.3.1.1 and 6.3.1.2). Furthermore, the assessment confirmed that the proposed solution was compatible with real-time applications. In contrast, the purpose of the human assessment was to determine how usable the two case studies were compared to the other UI applications. To this end, a questionnaire was conducted at the University of Huddersfield to evaluate the usability of the UIs of the case studies and those of existing applications. Analysis of the gathered data was conducted, and the one-way ANOVA test was again applied to determine the extent to which the proposed UIs differed from the existing UI applications. The p-value obtained was greater than 0.05, indicating that the dimensions of usability, usefulness, information quality, interface quality and satisfaction did not differ significantly (see Table 6.9).

The outcomes of the research analysis support the use of the proposed model as a design process tool for software engineers and designers. The case study UIs that were created with the proposed model were demonstrated to be more straightforward than several existing UI applications in terms of hierarchical structure design and number of used elements. The model was further validated by the fact that the two case studies possessed acceptable usability regarding the assessed dimensions of usefulness, information quality, interface quality, and overall satisfaction. Hence, UI designers and developers can employ the proposed model to facilitate the process of mobile application development.

8.2 Research Contributions

- A new model rooted in hierarchical structure and set theory and intended to facilitate the creation of usable UIs is the major contribution of the thesis.

A novel formal model constitutes the main contribution of this thesis. From a review of current literature that has focused on formal models of user interface design, it can be pointed out that the relationships among various user interface elements has not been discussed. For this model, formal rules were outlined, composition and integration were addressed, and accuracy in terms of syntax and semantics was demonstrated (Chapter 4). Formal characterisation of the UI design can be achieved by the FMMUID, as proven by

several examples (Chapter 5). The UI composition was defined on the basis of set theory language. Furthermore, to determine how easy UIs were to understand and how efficient they were with respect to structure and usability, a hierarchy diagram was employed. The results are demonstrated the proposed model was flexible enough and compatible with UIs of both lesser and greater complexity. In addition, this research has been able to show the use and application of the advanced model technique by discussing in great length the two case studies of how the model can be used and applied practically. The findings of the research pertaining to the advanced model for user interface design is crucial to those who design mobile user interfaces on key components as they can be applied in the design of the user interfaces as well as in enhancing usability.

- Improvement of software system quality and greater cost-effectiveness can be achieved with the proposed model.

Design models can be formally analysed by integrating UI design into formal models, thus ensuring the accuracy of the developed systems and making them more trustworthy. The findings of the study in view of the advanced approach are significant to the industry sector since examinations to the proposed model in the initial phases of the advancement process can result to identification of possible issues making it easy to address them at an initial stage. In the study, basic set theory presumptions can be directly comprehended by the designers as well as developers and applied. Thus improving the time- and cost-effectiveness of software development and enhancing the quality of the product.

- The proposed model demonstrated flexibility in terms of practical implementation regardless of the designers' or developers' level of knowledge of formal methods.

From a literature review of past studies, it has been found that only one or two case studies have been highlighted as examples. In this specific research, the two case studies have been used for both basic and advanced user interface designs. The proposed model is as flexible as it is appealing to the software developer. It has been demonstrated that the proposed model is compatible with UI designs of both greater and lesser complexity and that its use does not require anything other than straightforward set theory notation. This means that design development can be achieved with this model with no need for in-depth understanding of formal methods. As indicated by the example case studies in Chapter 5, UI design development can be successfully accomplished using the proposed model. The results obtained provide justification for the research (Chapter 6).

- Alternative designs can be supported by the proposed FMMUID model that has been legitimated by the empirical findings.

Alternative designs associated with particular designer requirements are made possible by the model proposed in this thesis for the UI design process. The results obtained from the two case studies involving UIs of greater and lesser complexity confirmed that the model was efficient. Present literature has not referred to any research when it comes to applying existing applications on their models. In this thesis, the detailed analysis was performed where the proposed model was applied on existing UI applications. The results obtained confirmed that the proposed model was effective in the achieve of different designs of greater or lesser complexity. Moreover, the proposed model may also be applicable to the design rationale (Chapter 6).

- Hierarchy diagrams were employed to graphically demonstrate the proposed model and to address various dimensions of design.

On the other hand, current literature on formal models of user interface design has not been featured hierarchical diagrams as mathematical depiction. In this research, hierarchy diagrams were employed with the formal model to afford designers a good understanding of the various dimensions of the design. A hierarchy diagram is useful because it shows how understandable a system or element is and how easy it is to validate with respect to structure or usability. Any new or screen sets may simply be located in our hierarchy by establishing the nature of the behaviour (behaviours) demonstrated. Those associated with the formal specification can be aided by the FMMUID, who can give them a structured means of including a system's UI in their set theory specification. (Chapter 4). The FMMUID is not limited to any particular type of system or style of interface. As part of the background to using the FMMUID, supporting diagrams have also been provided, which allow designers to identify the appropriate category within the hierarchy at any stage in the refinement of their specification. Moreover, the FMMUID introduces a simple structure view that hides all the details and shows only the structure (Chapter 4).

Complexity can be considered a measure of difficulty when the intricate interactions between several elements of an object are explored. Since comprehension has various determinants and its evaluation is subjective in nature, the perceived complexity of a UI is informative about what users think of the information display and the device screen. Moreover, the complexity of the outcome support and elucidation determines how productive the proposed approach is. Four factors have been identified as particularly

influential of how complex a UI design can be; these are navigation (number of screens determines depth and breadth), number of components, number of functions, and screen colours (section 7.2). By enhancing these factors, the interface may be made more complex and difficult to understand.

- Two approaches were employed to measure the complexity and usability of the model.

Prevailing literature about the evaluation of UI design does not measure usability and complexity simultaneously. In this thesis, the first assessment approach involved a comparative analysis of the two case studies and other existing applications in terms of structure and number of elements, functions and colours (see sections 7.2 and 7.3). According to the results obtained, the structure of the proposed model was less complicated than that of the other applications, while the number of elements, functions and colours it contained was similar to those of other applications. The second assessment approach entailed a comparison between the two case studies and other existing application regarding their usability as reflected in four dimensions. This was achieved by means of a questionnaire, the results of which revealed that the case studies and the existing application did not differ significantly in terms of usability (section 7.4).

8.3 Research Limitations

Manual analysis is one of the limitations of the UI analysis approach adopted in the thesis; because it is susceptible to human error, the use of manual analysis can distort the results. Another limitation is the difficulty involved in the processes of design analysis and quantification of complex UI components, both of which consumed considerable time and effort. The mobile applications chosen for purposes of comparison with the case studies in terms of design structure and usability constitute an additional limitation. Three mobile applications based on the iOS platform were selected for each of the two case studies. Despite the fact that the results shed some light on the usability of the case studies, a larger number of mobile applications should be used to gain a clearer picture. In addition, as far as limitations of the study are concerned, one of the key limitation of the study entails the sample since the sample was only made up of those aged from 18-34 years and 35-49 years. As a result, those aged from 49-60 never got a chance to take part in the study. As a result, the extent to which the findings of the study can be said to be generalized to the entire research population is questionable.

8.4 Further Research

The following are believed to be viable lines of inquiry for future research.

- 1- Future research should assess the applicability or viability of the proposed model in the industrial sector. On a pragmatic level, it is hoped that the proposed model will be useful in industrial settings and that it will help enhance the quality of interactive software.
- 2- The development of a more time-effective and less effort-intensive method of design analysis and quantification of UI components is also worth pursuing.
- 3- Currently, only the iOS platform supports the developed prototypes and chosen applications. Therefore, future research should aim to make the proposed model compatible with additional platforms, especially Android and Windows, to attain a more detailed and thorough assessment.
- 4- Future research could attempt the development of a new model of hierarchy levels that could be connected to the FMMUID. To make it possible to determine how complex the UI structure is, the number of levels (depth) and the number of screens at every level (breadth) must be known.
- 5- Future research could also examine the interactive relationships between the UI components (the binary relation is the basis of model development) to facilitate fulfilment of different design requirements and make the design more efficient.

8.5 Published Papers

R. Ihnissi and J. Lu, (2014) "An investigation into the problems of user oriented interfaces in mobile applications," presented at The 2014 World Congress in Computer Science, Computer Engineering, and Applied Computing (WORLDCOMP'14), Las Vegas, USA, 2014, pp. 100-106.

R. Ihnissi and J. Lu, (2015) "An Investigation into Game Based Learning Using High Level Programming Languages". In: Proceedings of the Fifth International Conference on Advanced Communications and Computation. INFOCOMP (2015). IARIA, Brussels, Belgium, pp. 99-104.

R. Ihnissi and J. Lu, (2017). "A Novel Formal Model Approach for Mobile User Interface Design", IEEE transactions on software engineering journal (manuscript under review process).

References

- Abowd, G. D., Coutaz, J., & Nigay, L. (1992). Structuring the Space of Interactive System Properties. *Engineering for Human-Computer Interaction*, 18, 113-129.
- Abrams, M., Phanouriou, C., Batongbacal, A. L., Williams, S. M., & Shuster, J. E. (1999). UIML: an appliance-independent XML user interface language. *Computer networks*, 31(11), 1695-1708.
- Abran, A., Khelifi, A., Suryn, W., & Seffah, A. (2003). Usability meanings and interpretations in ISO standards. *Software Quality Journal*, 11(4), 325-338.
- Abrial, J.-R., & Abrial, J.-R. (2005). *The B-book: assigning programs to meanings*: Cambridge University Press.
- Abrial, J.-R., Butler, M., Hallerstede, S., Hoang, T. S., Mehta, F., & Voisin, L. (2010). Rodin: an open toolset for modelling and reasoning in Event-B. *International Journal on Software Tools for Technology Transfer (STTT)*, 12(6), 447-466.
- Acock, A. C. (2008). *A gentle introduction to Stata (2nd ed.)*. College Station, Texas: Stata press.
- Alemerien, K., & Magel, K. (2014). GUIEvaluator: A Metric-tool for Evaluating the Complexity of Graphical User Interfaces. Paper presented at the SEKE, 13-18.
- Alford, M. (1994). Attacking requirements complexity using a separation of concerns. Paper presented at the Requirements Engineering, 1994., Proceedings of the First International Conference on, 2-5.
- Ali, A. A. (2013). A framework for measuring the usability issues and criteria of mobile learning applications. *Electronic Thesis and Dissertation Repository*. 1184. The University of Western Ontario. <http://ir.lib.uwo.ca/etd/1184>.
- Alroobaea, R., & Mayhew, P. J. (2014). How many participants are really enough for usability studies? Paper presented at the Science and Information Conference (SAI), 2014, 48-56.
- Altaboli, A., & Lin, Y. (2011). Objective and subjective measures of visual aesthetics of website interface design: the two sides of the coin. Paper presented at the International Conference on Human-Computer Interaction, 35-44.
- Alty, J. L. (1992). Can we measure usability. *Proceedings of Advanced Information Systems*, 95-106.

- Andrews, D., Bruun, H., Hansen, B., Larsen, P., & Plat, N. (1996). Information technology—Programming languages, their environments and system software interfaces—Vienna Development Method—Specification Language—Part 1: Base language. International Organization for Standardization, 13817-13811.
- Andriole, S. J., & Adelman, L. (1995). Cognitive systems engineering for user-computer interface design, prototyping, and evaluation: L. Erlbaum Associates Inc.
- Anneberg, L., & Singh, H. (1993). Circuit theoretic approaches to determining software complexity. Paper presented at the Circuits and Systems, 1993., Proceedings of the 36th Midwest Symposium on, 895-898.
- Arthur, R., & Olsen Jr, D. R. (2011). XICE windowing toolkit: Seamless display annexation. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 18(3), 14:1–14:46.
- Ašeriškis, D., Blažauskas, T., & Damaševičius, R. (2017). UAREI: A model for formal description and visual representation/software gamification. *Dyna*, 84(200), 326-334.
- Baecker, R. M. (2014). *Readings in Human-Computer Interaction: toward the year 2000*: Morgan Kaufmann.
- Baier, C., Katoen, J.-P., & Larsen, K. G. (2008). *Principles of model checking*: MIT press.
- Balzert, H., Hofmann, F., Kruschinski, V., & Niemann, C. (1996). The JANUS Application Development Environment-Generating More than the User Interface. Paper presented at the CADUI, 183-207.
- Banati, H., Bedi, P., & Grover, P. (2006). Evaluating web usability from the user's perspective. *Journal of Computer Science*, 2(4), 314-317.
- Basili, V. R. (1980). Qualitative software complexity models: A summary. Tutorial on models and methods for software management and engineering.
- Bastien, J. C. (2010). Usability testing: a review of some methodological and technical aspects of the method. *International journal of medical informatics*, 79(4), e18-e23.
- Bastien, J. C., & Scapin, D. L. (1993). Ergonomic criteria for the evaluation of human-computer interfaces. Inria.

- Beckert, B., & Grebing, S. (2012). Evaluating the Usability of Interactive Verification Systems. Paper presented at the COMPARE.
- Berti, S., Correani, F., Mori, G., Paternò, F., & Santoro, C. (2004). TERESA: a transformation-based environment for designing and developing multi-device interfaces. Paper presented at the CHI'04 extended abstracts on Human factors in computing systems.
- Berti, S., Correani, F., Paterno, F., & Santoro, C. (2004). The TERESA XML language for the description of interactive systems at multiple abstraction levels. Paper presented at the Proceedings Workshop on Developing User Interfaces with XML: Advances on User Interface Description Languages.
- Bertino, E. (1985). Design issues in interactive user interfaces. *Interfaces in Computing*, 3(1), 37-53.
- Bevan, N. (2001). International standards for HCI and usability. *International Journal of Human-Computer Studies*, 55(4), 533-552.
- Bevan, N. K., & Kirakowski, J. (1991a). J. And Maissel, J.(1991). What is usability. Paper presented at the Proceedings of the 4th International Conference on HCI, Stuttgart.
- Bevan, N. K., & Kirakowski, J. M. (1991b). What is usability. Paper presented at the Proceedings of the 4th International Conference on HCI, Stuttgart.
- Bézivin, J., & Gerbé, O. (2001). Towards a precise definition of the OMG/MDA framework. Paper presented at the Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on.
- Biel, B., Grill, T., & Gruhn, V. (2010). Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. *Journal of Systems and Software*, 83(11), 2031-2044.
- Blandford, A. E., Hyde, J. K., Green, T. R., & Connell, I. (2008). Scoping analytical usability evaluation methods: a case study. *Human-Computer Interaction*, 23(3), 278-327.
- Bodart, F., Hennebert, A.-M., Leheureux, J.-M., Provot, I., Sacré, B., & Vanderdonckt, J. (1995). Towards a systematic building of software architecture: The TRIDENT methodological guide Design, Specification and Verification of Interactive Systems' 95, 262-278: Springer.

- Booch, G., Rumbaugh, J., & Jacobson, I. (2005). Unified modeling language User Guide: Addison Wesley Professional.
- Booth, P. (2014). An Introduction to Human-Computer Interaction (Psychology Revivals): Psychology Press.
- Bowen, J. (2008). Formal models and refinement for graphical user interface design. The University of Waikato.
- Bowen, J., & Reeves, S. (2008a). Formal models for user interface design artefacts. *Innovations in Systems and Software Engineering*, 4(2), 125-141.
- Bowen, J., & Reeves, S. (2008b). Refinement for user interface designs. *Electronic Notes in Theoretical Computer Science*, 208, 5-22.
- Bowles, J., & Kloul, L. (2010). Synthesising PEPA nets from IODs for performance analysis. Paper presented at the Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering.
- Bowles, J. K. F., & Bordbar, B. (2007). A formal model for integrating multiple views. Paper presented at the Application of Concurrency to System Design, 2007. ACSD 2007. Seventh International Conference on.
- Brambilla, M., & Fraternali, P. (2014). Interaction flow modeling language: Model-driven UI engineering of web and mobile apps with IFML: Morgan Kaufmann.
- Bramwell, C., Fields, B., & Harrison, M. (1995). Exploring design options rationally Design, Specification and Verification of Interactive Systems' 95 (pp. 134-148): Springer.
- Brinck, T., Gergle, D., & Wood, S. D. (2002). Designing Web sites that work: Usability for the Web: Morgan Kaufmann Publishers.
- Brooks, I. (1993). Object-oriented metrics collection and evaluation with a software process. Paper presented at the Proc. OOPSLA.
- Brown, A. W. (2004). Model driven architecture: Principles and practice. *Software and Systems Modeling*, 3(4), 314-327.
- Buchenau, M., & Suri, J. F. (2000). Experience prototyping. Paper presented at the Proceedings of the 3rd conference on Designing interactive systems: processes, practices, methods, and techniques.
- Buie, E., & Murray, D. (2012). Usability in government systems: User experience design for citizens and public servants: Elsevier.

- Bumbulis, P., Alencar, P. S., Cowan, D. D., & Lucena, C. (1995). Combining formal techniques and prototyping in user interface construction and verification Design, Specification and Verification of Interactive Systems' 95 (pp. 174-192): Springer.
- Bygstad, B., Ghinea, G., & Brevik, E. (2008). Software development methods and usability: Perspectives from a survey in the software industry in Norway. *Interacting with Computers*, 20(3), 375-385.
- Cabot, J., Claris, R., & Riera, D. (2008). Verification of UML/OCL class diagrams using constraint programming. Paper presented at the Software Testing Verification and Validation Workshop, 2008. ICSTW'08. IEEE International Conference on.
- Calvary, G., Coutaz, J., Dâassi, O., Balme, L., & Demeure, A. (2004). Towards a New Generation of Widgets for Supporting Software Plasticity: The "Comet". *Ehci/Ds-Vis*, 3425, 306-324.
- Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., & Vanderdonckt, J. (2003). A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3), 289-308.
- Calvary, G., & Pinna, A.-M. (2008). Lessons of Experience in Model-Driven Engineering of Interactive Systems: Grand challenges for MDE? Paper presented at the First International Workshop on Challenges in Model-Driven Software Engineering (ChaMDE), MODELS.
- Camburn, B. A., Arlitt, R., Perez, K. B., Anderson, D., Choo, P. K., Lim, T., . . . Wood, K. (2017). Design prototyping of systems. *ICED 2017*.
- Card, S. K., Newell, A., & Moran, T. P. (1983). The psychology of human-computer interaction.
- Carr, M., & Verner, J. (1997). Prototyping and software development approaches. Department of Information Systems, City University of Hong Kong, Hong Kong.
- Carroll, J. M. (2013). Human computer interaction-brief intro. *The Encyclopedia of Human-Computer Interaction*, 2nd Ed.

- Carvalho, A. A. A. (2001). Usability Testing of Educational Software: methods, techniques and evaluators. *Actas do 3º Simpósio Internacional de Informática Educativa*, 139-148.
- Casaló, L. V., Flavián, C., & Guinaliu, M. (2010). Generating trust and satisfaction in e-services: the impact of usability on consumer behavior. *Journal of Relationship Marketing*, 9(4), 247-263.
- Cassandras, C. G., & Lafortune, S. (1999). *Introduction to Discrete Event Systems* (The International Series on Discrete Event Dynamic Systems).
- Cataldo, M., De Souza, C. R., Bentolila, D. L., Miranda, T. C., & Nambiar, S. (2010). The impact of interface complexity on failures: an empirical analysis and implications for tool design. School of Computer Science, Carnegie Mellon University, Tech. Rep.
- Catarci, T., Dongilli, P., Mascio, T. D., Franconi, E., Santucci, G., & Tessaris, S. (2004). An ontology based visual tool for query formulation support. Paper presented at the Proceedings of the 16th European Conference on Artificial Intelligence.
- Cerny, T., Chalupa, V., & Donahoo, M. J. (2012). Impact of user interface generation on maintenance. Paper presented at the Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on.
- Chen, C.-Y., & White, B.-K. W. (2013). Hierarchical organization chart for mobile applications: Google Patents.
- Chhillar, U., & Bhasin, S. (2011). A Journey of Software Metrics: Traditional to Aspect-Oriented Paradigm. Paper presented at the 5th National Conference on Computing For Nation Development, 10th-11th March.
- Chignell, M. H. (1990). A taxonomy of user interface terminology. *ACM SIGCHI Bulletin*, 21(4), 27.
- Cimatti, A., Roveri, M., Susi, A., & Tonetta, S. (2011a). Formalizing requirements with object models and temporal constraints. *Software & Systems Modeling*, 10(2), 147-160.
- Cimatti, A., Roveri, M., Susi, A., & Tonetta, S. (2011b). Formalizing requirements with object models and temporal constraints. *Software and Systems Modeling*, 10(2), 147-160.

- Clerckx, T., Winters, F., & Coninx, K. (2005). Tool support for designing context-sensitive user interfaces using a model-based approach. Paper presented at the Proceedings of the 4th international workshop on Task models and diagrams.
- Corporation, M. (2000, 09/05/2017). UI Guidelines vs. Usability Testing. Retrieved 09/05/2017, from <https://msdn.microsoft.com/en-us/library/ms997578.aspx>
- Coskun, E., & Grabowski, M. (2005). Impacts of user interface complexity on user acceptance and performance in safety-critical systems. *Emergency*, 2(1), 3.
- Costabile, M. F. (2001). Usability in the software life cycle. *Handbook of software engineering and knowledge engineering*, 1, 179-192.
- Coutaz, J., & Calvary, G. (2012). *HCI and software engineering for user interface plasticity*: CRC Press.
- Creswell, J. W., & Clark, V. L. P. (2011). *Designing and conducting mixed methods research*.
- Da Silva, P. P. (2000). User interface declarative models and development environments: A survey. Paper presented at the International Workshop on Design, Specification, and Verification of Interactive Systems.
- De Oliveira, R. A. j. (2015). Formal specification and verification of interactive systems with plasticity: applications to nuclear-plant supervision. Université Grenoble Alpes.
- Dickinson, A., Eisma, R., & Gregor, P. (2003). Challenging interfaces/redesigning users. Paper presented at the ACM SIGCAPH Computers and the Physically Handicapped.
- Dillon, A. (2001). The evaluation of software usability *Encyclopedia of human factors and ergonomics*: London: Taylor and Francis.
- Dinulescu, D. C. (2007). User interaction with computerized provider order entry systems: a method for quantitative measurement of cognitive complexity.
- Dix, A. (2016). Human-Computer Interaction. *Encyclopedia of Database Systems*, 1-6.
- Dix, A., Finlay, J. A., & Abowd, D. (1998). G. and Beale, R. *Human-Computer Interaction. 2nd ed. Prentice Hall*.
- Dix, A., Finlay, J., Abowd, G., & Beale, R. (2004). Evaluation techniques. *Human-Computer Interaction*.

- Dix, A. J., & Runciman, C. (1985). Abstract models of interactive systems. *People and Computers: Designing the interface*, 13-22.
- Doherty, G., & Harrison, M. D. (1997). A representational approach to the specification of presentations *Design, Specification and Verification of Interactive Systems' 97* (pp. 273-290): Springer.
- Dubey, S. K., & Gulati, A. (2012). Usability evaluation of software systems using fuzzy multi-criteria approach.
- Duce, D. (1995). Users: Summary of working group discussion. Paper presented at the *Interactive Systems: Design Specification and Verification*.
- Duke, D. J., & Harrison, M. D. (1993). Abstract interaction objects. Paper presented at the *Computer Graphics Forum*.
- Dumas, J. S., & Redish, J. (1999). *A practical guide to usability testing*: Intellect books.
- Duolingo. (2012). Duolingo - Learn Spanish, French and more. Retrieved 17/03/2016, 2015, from <https://itunes.apple.com/gb/app/duolingo-learn-spanish-french-and-more/id570060128?mt=8>
- Edmonds, B. (1995). *What is Complexity?-The philosophy of complexity per se with application to some examples in evolution The evolution of complexity*: Kluwer, Dordrecht.
- Elwert, T., & Schlungbaum, E. (1995). Modelling and generation of graphical user interfaces in the TADEUS approach *Design, Specification and Verification of Interactive Systems' 95* (pp. 193-208): Springer.
- Evans, R. T. (2002). Usability testing and research. *IEEE Transactions on Professional Communication*, 45(2), 151-152.
- Facebook, I. (2008). Facebook. Retrieved from <https://itunes.apple.com/gb/app/facebook/id284882215?mt=8>
- Faconti, G. P., & Paternò, F. (1990). An approach to the formal specification of the components of an interaction.
- Ferré, X., Juristo, N., Windl, H., & Constantine, L. (2001). Usability basics for software developers. *IEEE software*, 18(1), 22-29.
- Ferre, X., Villalba, E., Julio, H., & Zhu, H. (2017). Extending Mobile App Analytics for Usability Test Logging. Paper presented at the *IFIP Conference on Human-Computer Interaction*.

- Fidel, A. (2000). *Discovering statistics using SPSS for windows*: Sage Publications, London, UK.
- Fitzgerald, J., & Larsen, P. G. (2009). *Modelling systems: practical tools and techniques in software development*: Cambridge University Press.
- Foley, J., Kim, W. C., Kovacevic, S., & Murray, K. (1991). UIDE—an intelligent user interface design environment. Paper presented at the Intelligent user interfaces.
- Folmer, E., & Bosch, J. (2004). Architecting for usability: a survey. *Journal of Systems and Software*, 70(1), 61-78.
- Fonseca, J. (2010). W3C Model-Based UI XG Final Report, May 2010.
- Foster, J. J. (2001). *Data Analysis Using SPSS for Windows Versions 8-10: A Beginner's Guide*: Sage.
- Freiberg, M., & Baumeister, J. (2008). A survey on usability evaluation techniques and an analysis of their actual application. Institute of Computer Science, University of Wurzburg, Germany.
- Gajos, K., & Weld, D. S. (2004). SUPPLE: automatically generating user interfaces. Paper presented at the Proceedings of the 9th international conference on Intelligent user interfaces.
- Galitz, W. O. (2007). *The essential guide to user interface design: an introduction to GUI design principles and techniques*: John Wiley & Sons.
- Garlan, D., & Shaw, M. (1993). An introduction to software architecture. *Advances in software engineering and knowledge engineering*, 1(3.4).
- Gena, C., & Torre, I. (2004). The importance of adaptivity to provide onboard services: A preliminary evaluation of an adaptive tourist information service onboard vehicles. *Applied Artificial Intelligence*.
- George, D., & Mallery, M. (2003). *Using SPSS for Windows step by step: a simple guide and reference*.
- Gerhardt-Powals, J., Javecchia, H., Andriole, S., & Miller III, R. (1995). Cognitive redesign of submarine displays. *Cognitive Systems Engineering for User-computer Interface Design, Prototyping, and Evaluation*. Lawrence Erlbaum Associates, Inc., New York, 194-231.

- Gill, N. S., & Grover, P. (2004). Few important considerations for deriving interface complexity metric for component-based systems. *ACM SIGSOFT Software Engineering Notes*, 29(2), 4-4.
- Gilmore, S., Hillston, J., Kloul, L., & Ribaud, M. (2003). PEPA nets: a structured performance modelling formalism. *Performance Evaluation*, 54(2), 79-104.
- Giorgi, G., Guerraggio, A., & Thierfelder, J. (2004). *Mathematics of optimization: smooth and nonsmooth case*: Elsevier.
- Gliem, J. A., & Gliem, R. R. (2003). Calculating, interpreting, and reporting Cronbach's alpha reliability coefficient for Likert-type scales.
- Goh, Y., & Case, K. (2016). *Advances in Manufacturing Technology XXX: Proceedings of the 14th International Conference on Manufacturing Research, Incorporating the 31st National Conference on Manufacturing Research, September 6–8, 2016, Loughborough University, UK (Vol. 3)*: IOS Press.
- Golovine, J. C. R. R. (2013). *Experimental user interface design toolkit for interaction research (IDTR)*.
- González, M. P., Lorés, J., & Granollers, A. (2008). Enhancing usability testing through datamining techniques: A novel approach to detecting usability problem patterns for a context of use. *Information and Software Technology*, 50(6), 547-568.
- Google, I. (2011). *Google+ – interests, communities, discovery*. Retrieved from <https://itunes.apple.com/gb/app/google-interests-communities-discovery/id447119634?mt=8>
- Göransson, B., Gulliksen, J., & Boivie, I. (2004). *The Usability Design Process—Integrating User-centered Systems Design in the Software Development Process Research Section*.
- Grayter, F., & Wallnau, L. (2014). *Essentials of statistics for the behavioral sciences*: Belmont, CA: Wadsworth.
- Griffiths, T., Barclay, P. J., Paton, N. W., McKirdy, J., Kennedy, J., Gray, P. D., . . . da Silva, P. P. (2001). Teallach: a model-based user interface development environment for object databases. *Interacting with Computers*, 14(1), 31-68.
- Grumberg, O., & Long, D. E. (1994). Model checking and modular verification. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 16(3), 843-871.

- Gulliksen, J., Harning, M. B., Palanque, P., Van der Veer, G., & Wesson, J. (2009). Engineering Interactive Systems.
- Gulliksen, J., Harning, M.B., Palanque, P., van der Veer, G.C., Wesson, J. (2007, March 2007). Engineering Interactive Systems-EIS 2007 Joint Working Conferences EHCI 2007, DSV-IS 2007, HCSE 2007, Selected Papers. Paper presented at the Engineering Interactive Systems-EIS 2007 Joint Working Conferences EHCI 2007, DSV-IS 2007, HCSE 2007, Salamanca, Spain.
- Gumtree.com. (2009). Gumtree Classifieds: Buy & Sell - Local for Sale. Retrieved from <https://itunes.apple.com/gb/app/gumtree-classifieds-buy-sell-local-for-sale/id487946174?mt=8>
- Hair, J., Anderson, R., Tatham, R., & Black, W. (2003). Multivariate data analysis. Pearson Education: India.
- Hamborg, K.-C., Vehse, B., & Bludau, H.-B. (2004). Questionnaire based usability evaluation of hospital information systems. *Electronic journal of information systems evaluation*, 7(1), 21-30.
- Han, S. H., Yun, M. H., Kwahk, J., & Hong, S. W. (2001). Usability of consumer electronic products. *International journal of industrial ergonomics*, 28(3), 143-151.
- Harms, I., & Schweibenz, W. (2000). Testing web usability. *Information Management & Consulting*, 15(3), 61-66.
- Hartson, H. R., Andre, T. S., & Williges, R. C. (2001). Criteria for evaluating usability evaluation methods. *International journal of human-computer interaction*, 13(4), 373-410.
- Hayes, P. J., Szekely, P. A., & Lerner, R. A. (1985). Design alternatives for user interface management systems based on experience with COUSIN. Paper presented at the ACM SIGCHI Bulletin.
- Henderson-Sellers, B. (1995). Object-oriented metrics: measures of complexity: Prentice-Hall, Inc.
- Henson, M. C., & Reeves, S. (2000). Investigating Z. *Journal of Logic and Computation*, 10(1), 43-73.

- Heymann, M., & Degani, A. (2007). Formal analysis and automatic generation of user interfaces: Approach, methodology, and an algorithm. *Human Factors*, 49(2), 311-330.
- Hillston, J., & Kloul, L. (2006). A function-equivalent components based simplification technique for PEPA models. Paper presented at the European Performance Engineering Workshop.
- Hinton, A., Kwiatkowska, M., Norman, G., & Parker, D. (2006). PRISM: A tool for automatic verification of probabilistic systems. Paper presented at the International Conference on Tools and Algorithms for the Construction and Analysis of Systems.
- Hix, D., Gabbard, J. L., Swan, J. E., Livingston, M. A., Hollerer, T., Julier, S. J., . . . Brown, D. (2004). A cost-effective usability evaluation progression for novel interactive systems. Paper presented at the System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on.
- Hoegh, R. T., Nielsen, C. M., Overgaard, M., Pedersen, M. B., & Stage, J. (2006). The impact of usability reports and user test observations on developers' understanding of usability data: An exploratory study. *International journal of human-computer interaction*, 21(2), 173-196.
- Holzinger, A. (2005). Usability engineering methods for software developers. *Communications of the ACM*, 48(1), 71-74.
- Hood, W. W., & Wilson, C. S. (2002). Solving problems in library and information science using fuzzy set theory. *Library trends*, 50(3), 393.
- Hsieh, S., & Huang, S. (2008). Usability Evaluation: A Case Study. Paper presented at the APIEMS 2008 Proceedings of the 9th Asia Pacific Industrial Engineering & Management Systems Conference.
- Ihnissi, R., & Lu, J. (2014). An investigation into the problems of user oriented interfaces in mobile applications. Paper presented at the Proceedings on the International Conference on Internet Computing (ICOMP), Las Vegas, USA, 2014, pp. 100-106.
- Ihnissi, R., & Lu, J. (2015). An Investigation into Game Based Learning Using High Level Programming Languages In: Proceedings of the Fifth International

- Conference on Advanced Communications and Computation. INFOCOMP (2015). IARIA, Brussels, pp. 99-104.
- Imaz, M., & Benyon, D. (2007). *Designing with blends: Conceptual foundations of human-computer interaction and software engineering methods*: Cambridge, MA: The MIT Press.
- Institution, B. S. (1998). *Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Guidance on Usability*: International Organization for Standardization.
- Iqbal, M., & Warraich, N. F. (2016). Usability evaluation of an academic library website: A case of the University of the Punjab. *Pakistan Journal of Information Management & Libraries (PJIM&L)*, 13.
- ISO, I. (1999). IEC 14598-4: Software Engineering-Product Evaluation-Part 4-Process for Acquirers. International Organization for Standardization, Geneva, Switzerland.
- Iso, I. (2002). IEC 13568: 2002: Information technology—Z formal specification notation—Syntax, type system and semantics. ISO (International Organization for Standardization), Geneva, Switzerland.
- Iso, I., & Std, I. (2001). 9126 Software product evaluation—quality characteristics and guidelines for their use. ISO/IEC Standard, 9126.
- Jacob, R. J. (1983). Using formal specifications in the design of a human-computer interface. *Communications of the ACM*, 26(4), 259-264.
- Jacob, R. J. (2003). *User interface*.
- Jadhav, D., Bhutkar, G., & Mehta, V. (2013). *Usability evaluation of messenger applications for Android phones using cognitive walkthrough*. Paper presented at the Proceedings of the 11th Asia Pacific Conference on Computer Human Interaction.
- Jakob, N. (1993). *Usability engineering*. Fremont, California: Morgan.
- Jakob, N., & Mack Robert, L. (1994). *Usability inspection methods*. CHI Tutorials, 413-414.
- Janssen, C., Weisbecker, A., & Ziegler, J. (1993). Generating user interfaces from data models and dialogue net specifications. Paper presented at the Proceedings of

- the INTERACT'93 and CHI'93 conference on human factors in computing systems.
- Jech, T. (2013). Set theory: Springer Science & Business Media.
- Jeffrey, R., & Chisnell, D. (1994). Handbook of usability testing: how to plan, design, and conduct effective tests: New York John Wiley Sons.
- Jeng, J. (2005). Usability assessment of academic digital libraries: effectiveness, efficiency, satisfaction, and learnability. *Libri*, 55(2-3), 96-121.
- Jennifer, P., Yvonne, R., & Helen, S. (2002). Interaction design: beyond humancomputer interaction). New York. John Wiley & Sons, Inc.
- Jensen, K., & Kristensen, L. M. (2009). Coloured Petri nets: modelling and validation of concurrent systems: Springer Science & Business Media.
- Jensen, K., Kristensen, L. M., & Wells, L. (2007). Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *International Journal on Software Tools for Technology Transfer*, 9(3-4), 213-254.
- Ji, Y. G., Park, J. H., Lee, C., & Yun, M. H. (2006). A usability checklist for the usability evaluation of mobile phone user interface. *International journal of human-computer interaction*, 20(3), 207-231.
- Jordan, P. W. (2002). Designing pleasurable products: An introduction to the new human factors: CRC press.
- Juristo, N. (2009). Impact of usability on software requirements and design Software Engineering (pp. 55-77): Springer.
- Juristo, N., Moreno, A. M., & Sanchez-Segura, M.-I. (2007a). Analysing the impact of usability on software design. *Journal of Systems and Software*, 80(9), 1506-1516.
- Juristo, N., Moreno, A. M., & Sanchez-Segura, M.-I. (2007b). Guidelines for eliciting usability functionalities. *IEEE Transactions on Software Engineering*, 33(11).
- Kalnins, R. D., Markosian, L., Meier, B. J., Kowalski, M. A., Lee, J. C., Davidson, P. L., . . . Finkelstein, A. (2002). WYSIWYG NPR: Drawing strokes directly on 3D models. *ACM Transactions on Graphics (TOG)*, 21(3), 755-762.
- Kang, H. G., & Seong, P. H. (1998). An information theory-based approach for quantitative evaluation of user interface complexity. *IEEE Transactions on Nuclear Science*, 45(6), 3165-3174.

- Karat, C.-M. (1993). Usability engineering in dollars and cents. *IEEE software*, 10(3), 88-89.
- Karray, F., Alemzadeh, M., Saleh, J. A., & Arab, M. N. (2008). Human-computer interaction: Overview on state of the art.
- Katoen, J.-P. (2008). Perspectives in probabilistic verification. Paper presented at the Theoretical Aspects of Software Engineering, 2008. TASE'08. 2nd IFIP/IEEE International Symposium on.
- Kennard, R., & Steele, R. (2008). Application of software mining to automatic user interface generation. Paper presented at the International Conference on Software Methods and Tools.
- Kennard, R. D. (2011). Derivation of a general purpose architecture for automatic user interface generation.
- Kieras, D. (2009). Model-based evaluation. *The Human-Computer Interaction: Development Process*, 294-310.
- Kindersley, D. (2012). DK Quiz Game. Retrieved from <https://itunes.apple.com/gb/app/dk-quiz/id556884950?mt=8>
- Kirakowski, J. (2000). Questionnaires in usability engineering. *Human Factors Research Group*, Cork, Ireland.
- Kohlas, J., Meyer, B., & Schiper, A. (2006). Dependable systems: software, computing, networks: research results of the DICS program (Vol. 4028): Springer.
- Kong, J., Zhang, K., Dong, J., & Xu, D. (2009). Specifying behavioral semantics of UML diagrams through graph transformations. *Journal of Systems and Software*, 82(2), 292-306.
- Kühne, T. (2006). Matters of (meta-) modeling. *Software and Systems Modeling*, 5(4), 369-385.
- Kumari, U., & Upadhyaya, S. (2011). An interface complexity measure for component-based software systems. *International Journal of Computer Applications*, 36(1), 46-52.
- Kwiatkowska, M., Norman, G., & Parker, D. (2007). Stochastic model checking. Paper presented at the International School on Formal Methods for the Design of Computer, Communication and Software Systems.

- Larson, K., & Czerwinski, M. (1998). Web page design: Implications of memory, structure and scent for information retrieval. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.
- Lazar, J., Feng, J. H., & Hochheiser, H. (2010). *Research methods in human-computer interaction*: John Wiley & Sons.
- Lazar, J., Feng, J. H., & Hochheiser, H. (2017). *Research methods in human-computer interaction*: Morgan Kaufmann.
- Lecerof, A., & Paternò, F. (1998). Automatic support for usability evaluation. *IEEE Transactions on Software Engineering*, 24(10), 863-888.
- Lepreux, S., Vanderdonckt, J., & Michotte, B. (2006). Visual design of user interfaces by (de) composition. Paper presented at the International Workshop on Design, Specification, and Verification of Interactive Systems.
- Lewis, C., & Wharton, C. (1997). Cognitive walkthroughs. *Handbook of human-computer interaction*, 2, 717-732.
- Lewis, J. R. (1992). *Psychometric evaluation of the post-study system usability questionnaire: The PSSUQ*. Paper presented at the Proceedings of the Human Factors and Ergonomics Society Annual Meeting.
- Lewis, J. (1993). IBM computer usability satisfaction questionnaires: Psychometric evaluation and instructions for use (Tech. Report 54.786). Boca Raton, FL: IBM Corp.
- Lewis, J. R. (1995). IBM computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *International Journal of Human-Computer Interaction*, 7(1), 57-78.
- Lewis, J. R. (2002). Psychometric evaluation of the PSSUQ using data from five years of usability studies. *International Journal of Human-Computer Interaction*, 14(3-4), 463-488.
- Lewis, J. R. (2006a). Sample sizes for usability tests: mostly math, not magic. *interactions*, 13(6), 29-33.
- Lewis, J. R. (2006b). Usability testing. *Handbook of human factors and ergonomics*, 12, e30.
- Likert, R. (1932). A technique for the measurement of attitudes. *Archives of psychology*.

- Limbourg, Q., & Vanderdonckt, J. (2005). Transformational development of user interfaces with graph transformations *Computer-Aided Design of User Interfaces IV* (pp. 107-120): Springer.
- Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., & Florins, M. (2004). USIXML: A User Interface Description Language Supporting Multiple Levels of Independence. Paper presented at the ICWE Workshops.
- Limbourg, Q., Vanderdonckt, J., Michotte, B., Bouillon, L., & López-Jaquero, V. (2004). USIXML: a language supporting multi-path development of user interfaces. Paper presented at the International Workshop on Design, Specification, and Verification of Interactive Systems.
- Loer, K., & Harrison, M. (2000). Formal interactive systems analysis and usability inspection methods: two incompatible worlds? Paper presented at the International Workshop on Design, Specification, and Verification of Interactive Systems.
- Loer, K., & Harrison, M. (2002). Towards usable and relevant model checking techniques for the analysis of dependable interactive systems. Paper presented at the Automated Software Engineering, 2002. Proceedings. ASE 2002. 17th IEEE International Conference on.
- Loer, K., & Harrison, M. D. (2006). An integrated framework for the analysis of dependable interactive systems (IFADIS): Its tool support and evaluation. *Automated Software Engineering*, 13(4), 469-496.
- Lonczewski, F., & Schreiber, S. (1996). The FUSE-System: an Integrated User Interface Design Environment. Paper presented at the CADUI.
- Long, J. (1989). *Cognitive ergonomics and human-computer interaction* (Vol. 1): Cambridge University Press.
- López-Espin, J., Gallud, J., Lazcorreta, E., Peñalver, A., & Botella, F. (2011). A formal view of distributed user interfaces. Paper presented at the Distributed User Interfaces CHI 2011 Workshop, University of Castilla-La Mancha, Spain.
- Love, S. (2005). *Understanding mobile human-computer interaction*: Butterworth-Heinemann.
- LTD, L. (2011). C/C++ Quiz. Retrieved from <https://itunes.apple.com/us/app/c-c++-quiz/id464252579?mt=8>

- Maalem, S., & Zarour, N. (2016). Challenge of validation in requirements engineering. *Journal of Innovation in Digital Ecosystems*, 3(1), 15-21.
- Macaulay, M. (1996). ASKING TO ASK: THE STRATEGIC FUNCTION OF INDIRECT REQUESTS FOR INFORMATION IN INTERVIEWS.
- Macik, M., Cerny, T., & Slavik, P. (2014). Context-sensitive, cross-platform user interface generation. *Journal on Multimodal User Interfaces*, 8(2), 217-229.
- Macleod, M. (1994). Usability in context: Improving quality of use. Paper presented at the Human Factors in Organizational Design and Management–IV (Proceedings of the International Ergonomics Association 4th International Symposium on Human Factors in Organizational Design and Management, Stockholm.
- Madan, A., & Dubey, S. K. (2012). Usability evaluation methods: a literature review. *International Journal of Engineering Science and Technology*, 4(2), 590-599.
- Maguire, M. (2001). Context of use within usability activities. *International Journal of Human-Computer Studies*, 55(4), 453-483.
- Markopoulos, P., Pycock, J., Wilson, S., & Johnson, P. (1992). Adept-A task based design environment. Paper presented at the System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on.
- Marshall, E., & Boggis, E. (2016). The Statistics Tutor's Quick Guide to Commonly Used Statistical Tests. *University of Sheffield*. Available online: <http://www.statstutor.ac.uk/resources/uploaded/tutorsquickguidetostatistics.pdf>.
- Martelli, S., Nofrini, L., Vendruscolo, P., & Visani, A. (2003). Criteria of interface evaluation for computer assisted surgery systems. *International journal of medical informatics*, 72(1), 35-45.
- Märting, C. (1996). Software Life Cycle Automation for Interactive Applications: The AME Design Environment. Paper presented at the CADUI.
- Martínez-Pérez, B., De La Torre-Díez, I., López-Coronado, M., & Herreros-González, J. (2013). Mobile apps in cardiology. *JMIR mHealth and uHealth*, 1(2).
- Marucci, L., Paterno, F., & Santoro, C. (2003). Supporting Interactions with Multiple Platforms Through User and Task Models. *Multiple User Interfaces, Cross-Platform Applications and Context-Aware Interfaces*, 217-238.
- McCabe, T. J. (1976). A complexity measure. *IEEE Transactions on Software Engineering*(4), 308-320.

- Meedeniya, D. A. (2013). Correct model-to-model transformation for formal verification. University of St Andrews.
- Meixner, G., Paterno, F., & Vanderdonckt, J. (2011). Past, present, and future of model-based user interface development. *i-com* 10 (3): 2-11, 2011.
- Meskens, J., Loskyll, M., Seißler, M., Luyten, K., Coninx, K., & Meixner, G. (2011). GUIDE2ux: a GUI design environment for enhancing the user experience. Paper presented at the Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems.
- Metzker, E., & Offergeld, M. (2001). An interdisciplinary approach for successfully integrating human-centered design methods into development processes practiced by industrial software development organizations *Engineering for Human-Computer Interaction* (pp. 19-33): Springer.
- Miller, D. P. (1981). The depth/breadth tradeoff in hierarchical computer menus. Paper presented at the Proceedings of the Human Factors Society Annual Meeting.
- Mital, A., & Pennathur, A. (2004). Advanced technologies and humans in manufacturing workplaces: an interdependent relationship. *International journal of industrial ergonomics*, 33(4), 295-313.
- Mohamed, I., & Patel, D. (2015). Android vs iOS security: A comparative study. Paper presented at the Information Technology-New Generations (ITNG), 2015 12th International Conference on.
- Moliterni, R. (2008). *Proceedings of the 11th Toulon-Verona International Conference on Quality in Services* (Vol. 44): Firenze University Press.
- Mori, G., Paterno, F., & Santoro, C. (2004). Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Transactions on Software Engineering*, 30(8), 507-520.
- Mori, G., Paternò, F., & Santoro, C. (2003). Tool support for designing nomadic applications. Paper presented at the Proceedings of the 8th international conference on Intelligent user interfaces.
- Morowitz, H. (1995). The emergence of complexity. *Complexity*, 1(1), 4-5.
- Mosbahi, O., Ayed, L. J. B., & Khalgui, M. (2011). A formal approach for the development of reactive systems. *Information and Software Technology*, 53(1), 14-33.

- Moschoyiannis, S., Shields, M. W., & Krause, P. J. (2005). Modelling component behaviour with concurrent automata. *Electronic Notes in Theoretical Computer Science*, 141(3), 199-220.
- Moussa, I., Pacalet, R., Blasquez, J., van Hulst, M., Fedeli, A., Lambert, J.-L., . . . Bricaud, P. (2002). Formal verification techniques: industrial status and perspectives. Paper presented at the Design, Automation and Test in Europe Conference and Exhibition, 2002. Proceedings.
- Muji, M. (2015). Logical Operators for the Data-oriented Design of the User Interfaces. *Procedia Technology*, 19, 810-815.
- Munzner, T. (2000). Interactive visualization of large graphs and networks. Citeseer.
- Myers, B. (1994). Challenges of HCI design and implementation. *interactions*, 1(1), 73-83.
- Myers, B., Hudson, S. E., & Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1), 3-28.
- Myers, B. A. (1992). State of the art in user interface software tools: Carnegie-Mellon University. Department of Computer Science.
- Myers, B. A. (1993). Why are human-computer interfaces difficult to design and implement: DTIC Document.
- Myers, B. A. (1995). User interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 2(1), 64-103.
- Myers, B. A., & Rosson, M. B. (1992). Survey on user interface programming. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.
- Navarre, D., Palanque, P., Bastide, R., Schyn, A., Winckler, M., Nedel, L., & Freitas, C. (2005). A formal description of multimodal interaction techniques for immersive virtual reality applications. *Human-Computer Interaction-INTERACT 2005*, 170-183.
- Navarre, D., Palanque, P., Ladry, J.-F., & Barboni, E. (2009). ICOs: A model-based user interface description technique dedicated to interactive systems addressing usability, reliability and scalability. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(4), 18.

- Nielsen, J. (1993a). Is usability engineering really worth it. *IEEE software*, 10(6), 90-92.
- Nielsen, J. (1993b). *Usability Engineering*, Academic Press pp. 191-194.
- Nielsen, J. (1994a). *Usability engineering*: Elsevier.
- Nielsen, J. (1994b). Usability inspection methods. Paper presented at the Conference companion on Human factors in computing systems.
- Nielsen, J. (1995). Usability inspection methods. Paper presented at the Conference companion on Human factors in computing systems.
- Nielsen, J., & Molich, R. (1990). Heuristic evaluation of user interfaces. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.
- Oates, B. J. (2005). *Researching information systems and computing*: Sage.
- och Dag, J. N., Regnell, B., Madsen, O. S., & Aurnum, A. (2001). An industrial case study of usability evaluation in market-driven packaged software development. Paper presented at the 9th International Conference on Human-Computer Interaction.
- Odeh, S., & Adwan, I. O. (2009). A Usability Testing Approach to Evaluate User-Interfaces in Business Administration. *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, 3(7), 1582-1591.
- Oracle. (2009). *GUIBUILDER Downloads*. Retrieved 13/12/2016, from <https://netbeans.org/projects/guibuilder/downloads>
- Pallant, J., & Manual, S. S. (2010). *A step by step guide to data analysis using SPSS. Berkshire UK: McGraw-Hill Education*.
- Pallant, J. (2013). *SPSS survival manual*: McGraw-Hill Education (UK).
- Paterno, F. (2012). *Model-based design and evaluation of interactive applications*: Springer Science & Business Media.
- Park, J., Han, S. H., Kim, H. K., Cho, Y., & Park, W. (2013). Developing elements of user experience for mobile phones and services: survey, interview, and observation approaches. *Human Factors and Ergonomics in Manufacturing & Service Industries*, 23(4), 279-293.

- Paternò, F., Mancini, C., & Meniconi, S. (1997). ConcurTaskTrees: A diagrammatic notation for specifying task models. Paper presented at the Human-Computer Interaction INTERACT'97.
- Paterno, F., Santoro, C., & Spano, L. D. (2009). MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(4), 19.
- Pedhazur, E. J., & Schmelkin, L. P. (2013). *Measurement, design, and analysis: An integrated approach*: Psychology Press.
- Peñalver, A., Lazcorreta, E., López, J., Botella, F., & Gallud, J. (2012). Schema driven distributed user interface generation. Paper presented at the Proceedings of the 13th International Conference on Interacción Persona-Ordenador.
- Piroi, F. (2004). User interface features in Theorema: A summary. Paper presented at the Mathematical User-Interfaces Workshop.
- Polson, P. G., Lewis, C., Rieman, J., & Wharton, C. (1992). Cognitive walkthroughs: a method for theory-based evaluation of user interfaces. *International Journal of man-machine studies*, 36(5), 741-773.
- Preece, J. (1993). Hypermedia, multimedia and human factors. *Interactive multimedia: Practice and promise*, 135-150.
- Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., & Carey, T. (1994). *Human-computer interaction* Reading, MA: Addison-Wesley.
- Pressman, R. S. (2005). *Software engineering: a practitioner's approach*: Palgrave Macmillan.
- Puerta, A., & Eisenstein, J. (1998). Interactively mapping task models to interfaces in MOBI-D. Paper presented at the Design, Specification and Verification of Interactive Systems.
- Puerta, A., & Eisenstein, J. (2002). XIML: a common representation for interaction data. Paper presented at the Proceedings of the 7th international conference on Intelligent user interfaces.
- Puerta, A. R. (1996). The MECANO Project: Comprehensive and Integrated Support for Model-Based Interface Development. Paper presented at the CADUI.

- Puerta, A. R., & Szekeley, P. (1994). Model-based interface development. Paper presented at the Conference companion on Human factors in computing systems.
- Quesenbery, W. (2004). Balancing the 5Es of Usability. *Cutter IT Journal*, 17(2), 4-11.
- Radatz, J., Geraci, A., & Katki, F. (1990). IEEE standard glossary of software engineering terminology. *IEEE Std*, 610121990(121990), 3.
- Rafe, V., Rahmani, A. T., Baresi, L., & Spoletini, P. (2009). Towards automated verification of layered graph transformation specifications. *IET software*, 3(4), 276-291.
- Rauf, A., Rehman, S. U., Batool, S., & Ali, S. A. (2010). Survey based usability evaluation of MS Word. Paper presented at the User Science and Engineering (i-USER), 2010 International Conference on.
- Ribeiro, L., Dotti, F. L., & Bardohl, R. (2005). A formal framework for the development of concurrent object-based systems *Formal Methods in Software and Systems Modeling* (pp. 385-401): Springer.
- Rieman, J., Clayton, L., & Peter, P. (1993). The cognitive walkthrough method: A practitioner's guide: Technical report, Institute of Cognitive Science, University of Colorado.
- Riihiaho, S. (2000). Experiences with usability evaluation methods. Licentiate thesis. Helsinki University of Technology. Laboratory of Information Processing Science.
- Riza, L. S., Janusz, A., Slezak, D., Cornelis, C., Herrera, F., Benitez, J. M., . . . Stawicki, S. (2015). Package 'RoughSets'.
- Rockinson-Szapkiw, A. (2013). Statistics guide.
- Rogers, Y., Sharp, H., & Preece, J. (2011a). Data Gathering. *Interaction Design: Beyond Human-computer Interaction*, 3rd Edition, 222-268.
- Rogers, Y., Sharp, H., & Preece, J. (2011b). *Interaction design: Beyond human-computer interaction*. Chichester: West Sussex, UK: Wiley.
- Root, R. W., & Draper, S. (1983). *Questionnaires as a software evaluation tool*. Paper presented at the Proceedings of the SIGCHI conference on Human Factors in Computing Systems.
- Rosenthal, D. (1988). A Simple X11 Client Program-or-How hard can it really be to write "Hello, World"? Paper presented at the USENIX Winter.

- Rosnow, R. L., & Rosenthal, R. (1996). *Beginning behavioral research: A conceptual primer*: Prentice-Hall, Inc.
- Ross, T., & Burnett, G. (2001). Evaluating the human-machine interface to vehicle navigation systems as an example of ubiquitous computing. *International Journal of Human-Computer Studies*, 55(4), 661-674.
- Rosson, M. B., & Carroll, J. M. (2002). *Usability engineering: scenario-based development of human-computer interaction*.
- Rubin, J. (1994). *Handbook of Usability Testing. How to Plan. Design. and Conduct Effective Tests*. John Wiley & Sons: Inc.
- Rubin, J., & Chisnell, D. (2008). *Handbook of usability testing: how to plan, design and conduct effective tests*: John Wiley & Sons.
- Rukshan, A., & Baravalle, A. (2012). Automated usability testing: Analysing asia web sites. *arXiv preprint arXiv:1212.1849*.
- Rumbaugh, J., Jacobson, I., & Booch, G. (2004). *Unified modeling language reference manual*, the: Pearson Higher Education.
- Russell, S. (2000). ISO 9000: 2000 and the EFQM excellence model: competition or co-operation? *Total quality management*, 11(4-6), 657-665.
- Ryu, Y. S. (2005). Development of usability questionnaires for electronic mobile products and decision making methods.
- Sahami Shirazi, A., Henze, N., Schmidt, A., Goldberg, R., Schmidt, B., & Schmauder, H. (2013). Insights into layout patterns of mobile user interfaces by an automatic analysis of android apps. Paper presented at the Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems.
- Salman, N. (2006). Complexity metrics as predictors of maintainability and integrability of software components. *Cankaya University Journal of Arts and Sciences*, 1(5).
- Santoro, C. (2005). *A Task Model-based Approach for Design and Evaluation of Innovative User Interfaces*: Presses univ. de Louvain.
- Satyavathy, G., & RachelBlessie, M. (2017). Human-Computer Interaction. *Digital Signal Processing*, 9(1), 4-6.
- Sauro, J., & Lewis, J. R. (2011). When designing usability questionnaires, does it hurt to be positive? Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.

- Schlungbaum, E. (1996). Model-based user interface software tools-current state of declarative models: Georgia Institute of Technology.
- Schnall, R., Cimino, J. J., & Bakken, S. (2012). Development of a prototype continuity of care record with context-specific links to meet the information needs of case managers for persons living with HIV. *International journal of medical informatics*, 81(8), 549-555.
- Schoeller, B., Widmer, T., & Meyer, B. (2006). Making specifications complete through models Architecting Systems with Trustworthy Components (pp. 48-70): Springer.
- Scholtz, J. (2004). Usability evaluation. National Institute of Standards and Technology.
- Seidewitz, E. (2003). What models mean. *IEEE software*, 20(5), 26-32.
- Selic, B. (2003). The pragmatics of model-driven development. *IEEE software*, 20(5), 19-25.
- Shackel, B. (2009). Usability-Context, framework, definition, design and evaluation. *Interacting with Computers*, 21(5-6), 339-346.
- Shackel, B., & Richardson, S. J. (1991). Human factors for informatics usability: Cambridge university press.
- Sharma, A., & Singh, M. (2013). Comparison of the Formal Specification Languages Based Upon Various Parameters. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 11(5), 37-39.
- Sharp, H., Rogers, Y., & Preece, J. (2007). Interaction design: beyond human-computer interaction.
- Shneiderman, B. (2010). Designing the user interface: strategies for effective human-computer interaction: Pearson Education India.
- Shneiderman, B., & Plaisant, C. (2005). Designing the user interface: strategies for effective. *Human-Computer Interaction*.
- Silva, J. R., & Santos, E. A. d. (2004). Applying Petri nets to requirements validation. Paper presented at the IFAC Symposium on Information Control Problems in Manufacturing. Salvador.
- Simon, H. (1969). The Architecture of Complexity: Hierarchic Systems, The Science of the Artificial. MIT Press, Cambridge, MA.

- Sinnig, D. (2004). The complicity of patterns and model-based UI development. Concordia University.
- Šnajberk, J. (2013). Pokročilá interaktivní zobrazování komponentového softwaru.
- Soley, R. (2000). Model driven architecture. OMG white paper, 308(308), 5.
- Spencer, D. (2004). What is usability? University of Melbourne, Melbourne, 108-115.
- Stone, D., Jarrett, C., Woodroffe, M., & Minocha, S. (2005). User interface design and evaluation: Morgan Kaufmann.
- Studio, M. V. (2017, 26/04/2017). Any Developer, Any App, Any Platform. Retrieved 26/04/2017, from <https://www.visualstudio.com/>
- Symantec, N. (2014). Internet security threat report 2014 (Vol. 19).
- Szekely, P. (1995). User interface prototyping: Tools and techniques. Paper presented at the Software Engineering and Human-Computer Interaction.
- Szekely, P., Luo, P., & Neches, R. (1992). Facilitating the exploration of interface design alternatives: the HUMANOID model of interface design. Paper presented at the Proceedings of the SIGCHI conference on Human factors in computing systems.
- Szekely, P., Sukaviriya, P., Castells, P., Muthukumarasamy, J., & Salcher, E. (1996). Declarative interface models for user interface construction tools: the MASTERMIND approach Engineering for Human-Computer Interaction (pp. 120-150): Springer.
- Takahara, Y., & Liu, Y. (2006). Transaction Processing System on Browser-Based Standardized User Interface. Foundations and Applications of Mis: A Model Theory Approach, 225-257.
- Tang, W., Ning, B., Xu, T., & Zhao, L. (2010). Scenario-based modeling and verification for ctcs-3 system requirement specification. Paper presented at the Computer Engineering and Technology (ICCET), 2010 2nd International Conference on.
- Taylor, B., & Heath, A. (1996). The use of double-sided items in scale construction. Center for Research into Elections and Social Trends. Working Paper(37).
- Teoh, K., Ong, T., Lim, P., Liong, R. P., & Yap, C. (2009). Explorations on web usability. American Journal of Applied Sciences, 6(3), 424.

- Thimbleby, H. (2004). User interface design with matrix algebra. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 11(2), 181-236.
- Thimbleby, H., Cairns, P., & Jones, M. (2001). Usability analysis with Markov models. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 8(2), 99-132.
- Thompson, K. M., McClure, C. R., & Jaeger, P. T. (2003). Evaluating federal websites: Improving e-government for the people. *Computers in society: Privacy, ethics, and the Internet*, 400-412.
- Trætteberg, H. (2002). Using user interface models in design *Computer-Aided Design of User Interfaces III* (pp. 131-142): Springer.
- Trivedi, M. C., & Khanum, M. A. (2012). Role of context in usability evaluations: A review. *arXiv preprint arXiv:1204.2138*.
- Trochim, W. (2006). The research methods knowledge base, Retrieved January 11, 2011.
- Tullis, T. S., & Stetson, J. N. (2004). *A comparison of questionnaires for assessing website usability*. Paper presented at the Usability professional association conference.
- Turner, C. W., Lewis, J. R., & Nielsen, J. (2006). Determining usability test sample size. *International encyclopedia of ergonomics and human factors*, 3(2), 3084-3088.
- Utting, M., & Legeard, B. (2010). *Practical model-based testing: a tools approach*: Morgan Kaufmann.
- Van den Bergh, J., Luyten, K., & Coninx, K. (2011). CAP3: context-sensitive abstract user interface specification. Paper presented at the Proceedings of the 3rd ACM SIGCHI symposium on Engineering interactive computing systems.
- Vanderdonckt, J. (1994). *Guide ergonomique des interfaces homme-machine*. Presses Universitaires de Namur.
- Vanderdonckt, J., & Bodart, F. (1996). The “Corpus Ergonomicus”: A Comprehensive and Unique Source for Human-Machine Interface. Paper presented at the Proceedings of the 1st International Conference on Applied Ergonomics.
- Vanderdonckt, J. M., & Bodart, F. (1993). Encapsulating knowledge for intelligent automatic interaction objects selection. Paper presented at the Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems.

- Walker, M., Takayama, L., & Landay, J. A. (2002). High-fidelity or low-fidelity, paper or computer? Choosing attributes when testing web prototypes. Paper presented at the Proceedings of the Human Factors and Ergonomics Society Annual Meeting.
- Warmer, J. B., & Kleppe, A. G. (1998). The object constraint language: Precise modeling with uml (addison-wesley object technology series).
- Weyers, B. (2017). Visual and Formal Modeling of Modularized and Executable User Interface Models The Handbook of Formal Methods in Human-Computer Interaction (pp. 125-160): Springer.
- Wichansky, A. M. (2000). Usability testing in 2000 and beyond. *Ergonomics*, 43(7), 998-1006.
- Wiecha, C., Bennett, W., Boies, S., Gould, J., & Greene, S. (1990). ITS: a tool for rapidly developing interactive applications. *ACM Transactions on Information Systems (TOIS)*, 8(3), 204-236.
- Wing, J. M. (1990). A specifier's introduction to formal methods. *Computer*, 23(9), 8-22.
- Winskel, G. (2010). Set theory for computer science. Unpublished lecture notes.
- Wirth, N. (2002). Program development by Stepwise Refinement Software pioneers (pp. 149-169): Springer.
- Wood, P., Breakwell, G., Hammond, S., & Fife-Schaw, C. (2000). Meta-analysis. *Research methods in psychology*, 2, 414-425.
- Woodcock, J., & Davies, J. (1996). Using Z: specification, refinement, and proof (Vol. 39): Prentice Hall Englewood Cliffs.
- Xchange, T. D. (2005, 25/11/2016). Tcl/Tk. Retrieved 26/04/2017, 2005, from <http://www.tcl.tk/>
- Xing, J. (2004). Measures of information complexity and the implications for automation design: DTIC Document.
- Yang, Y., Tan, Q., & Xiao, Y. (2005). Verifying web services composition based on hierarchical colored petri nets. Paper presented at the Proceedings of the first international workshop on Interoperability of heterogeneous information systems.

- Youxin, M., Xianghai, M., & Weimin, Y. (2009). Component Based Software Reuse Key Technology Research and Design. Paper presented at the Information Technology and Applications, 2009. IFITA'09. International Forum on.
- Yu, S., & Zhou, S. (2010). A survey on metric of software complexity. Paper presented at the Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on.
- Zaharias, P., & Poylymenakou, A. (2009). Developing a usability evaluation method for e-learning applications: Beyond functional usability. *Intl. Journal of Human-Computer Interaction*, 25(1), 75-98.
- Zins, A. H., Bauernfeind, U., Del Missier, F., Venturini, A., & Rumetshofer, H. (2004). An experimental usability test for different destination recommender systems: na.
- Zuse, H. (1993). Criteria for program comprehension derived from software complexity metrics. Paper presented at the Program Comprehension, 1993. Proceedings., IEEE Second Workshop on.

Appendices

Appendix A: Quiz game applications description and screen shoots

1- DK Quiz

DK Quiz is an application that offers an incredible opportunity to practice and develop one's General knowledge skills. The themes included in this quiz application are music, film, history, food, travel, insects, inventions, plants, sports and animals. The addictive DK Quiz game might be played either in a solo mode or turn based challenge mode. Internet connection is not required for playing the solo mode, however, challenge mode requires internet connection as this mode allows the user to go head-to-head with their friend by connecting them through the Facebook.


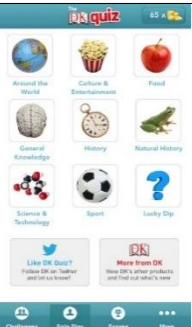

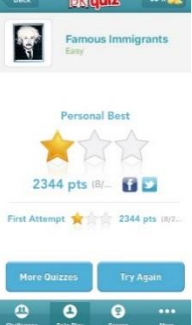
Screen name	Screenshots	Details
	DK Quiz	
S_s		1 image view, 1 text view, 2 button, 4 colours and 2 functions
S_i		11 text view, 1 image view, 12 buttons, 1 tab bar, 12 colours and 16 functions
S_g		3 text view, 1 image view, 2 buttons, 6, 1 rating bar, 9 colours and 5 functions
S_{info}		5 text view, 1 image view, 2 rating bar, 6 buttons, 7 colours and 10 functions

Figure A.1: Screen shots of model screens for DK UI.

2- Duolingo

Duolingo is a free app designed to for people who are avid language learners. Users can learn Spanish, French, German, Portuguese, Italian, Irish, Dutch, Danish, Swedish and English with this app. This app has earned high reputation and become widely popular, The Wall Street Journal stated “Duolingo may hold the secret to the future of education”. The design of language lessons in this app are short, straightforward and lively that allows the users to complete the lessons and reach next level soon. This feature of Duolingo app makes it addictive as the users can level up compete with friends.





Screen name	Screenshots	Details
	Duolingo	
S_s		1 text view, 2 buttons, 2 image view, 3 colours and 3 functions
S_i		62 text views, 2 image view, 62 buttons, 1, tab bar, 5colours and 65 functions
S_g		1 text view, 1 image view, 2 buttons, 6 colours and 3 functions
S_{info}		2 text view, 1 image view, 1 button, 1 table, 5 colour, 2 functions

Figure A.2: Screen shots of model screens for Duolingo UI.

3- C/C++ Quiz

C/C++ Quiz app is intended to help programmers to practice and test their C/C++ programming skills. With more than 500 question, this user-friendly app includes C and C++ MCQs to polish one's skills and enhance their programming knowledge. C/C++ Quiz app covers basic concepts, common errors in codes and code snippets and is very useful tool for candidates preparing for programming Interviews. The user first needs to select the category for starting the quiz, after which 15 random questions are listed to solve. Also, C/C++ quiz app allows users to set or adjust the timings for answering the questions before starting the quiz. On completing the quiz the users can cross check the answers, the app also provides a graphical analysis of the results.

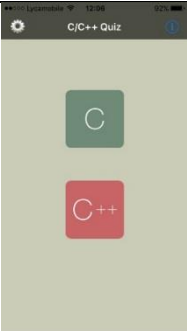
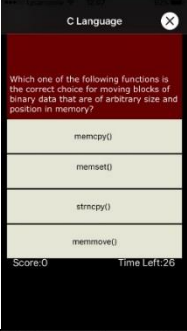
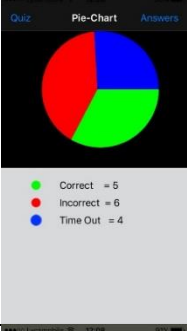
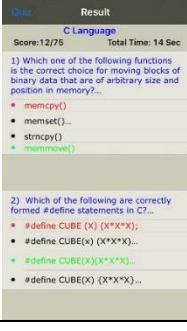
Screen name	Screenshots	Details
	C/C++ Quiz	
S_s		1 a text view, 4 buttons, 7 colours, 4 functions
S_i		8 text views, 5 buttons, 4 colours, 7 function
S_g		4 text views, 2 buttons, 1 chart, 8 colours, 4 functions
S_{info}		19 text views, 1 buttons, 1 table, 9 colour and 3 functions

Figure A.3: Screen shots of model screens for C/C++ UI.

Appendix B: Social media applications description and screen shots

1- Google+: interests, discovery, and communities

Google+ offers a great platform that users can use to learn incredible things and enable passionate people to create, and explore their interests. Google+ allows the users to join communities of people around various topics, group things they love into various collections, connect with individuals who share the same interests with them and build an online stream or home that is filled with remarkable contents based on their interests or discoveries.





Screen name	Screenshots	Details
	Google+ - interests, communities, discovery	
S_s		2 text view, 1 image view, 1 button, 6 colours and 1 function
S_i		1 collection view, 9 button, 3 text view, 3 image view, 1 tab bar, 3 colours and 19 functions
S_g		1 collection view, 1 default cell style, 11, 1 search bar, buttons, 5 colours and 16 functions
S_{info}		4 text view, 2 image view, 7 buttons, 1 video view, 1 default cell style, 4 colours and 12 functions

Figure B.1: Screen shots of model screens for Google+ UI.

2- Facebook

Facebook keep friends together as faster than ever, allowing the users to share their thoughts, updates, videos, and photos with their friends, see what their friends share or what they're up to and connect with their friends. Facebook also has salient features such as GPS, location features, as well as other optional features.





Screen name	Screenshots	Details
	Facebook	
S_s		1 Image view, 2 text field, 4 buttons, colours, 6 functions
S_i		4 text view, 1 search bar, 12 buttons, 1 tab bar, 1 image view, 1 video view, 5 colours and 22 functions
S_g		1 default cell style, 4 text view, 2 image view, 1 video view, 1 segmented controls, 9 buttons, 5 colours, 19 functions
S_{info}		1 text view, 3 image view, 1 video view, 14 buttons, 5 colours, 1 segmented controls, 24 functions

Figure B.2: Screen shots of model screens for Facebook UI.

3- Gumtree

Gumtree is an online platform where users can find job opportunities, buy and sell new or used cars, home products or phones and find numerous classified ads of various items of products from across United Kingdom. Gumtree also have local, housing and pets' services among others. Gumtree app allows the users to perform search and customize the searched results, instantly call or text the sellers or send a message and share the classified ads on social media such as Twitter, WhatsApp, Google+ and Facebook for promotion.





Screen name	Screenshots for	Details
	Gumtree	
S_s		7 text view, 1 search bar, 2 image view, 3 buttons, 2 collection view, 1 tab bar, 5 colours and 12 functions
S_i		4 text view, 1 search bar, 2 image view, 2 buttons, 2 collection view, 5 colours and 6 functions
S_g		1 default cell style, 4 buttons, 1 segmented controls, 7 colours, 10 functions
S_{info}		6 text view, 1 image view, 10 buttons, 6, 1 table, colours, 12 functions

Figure B.3: Screen shots of model screens for Gumtree UI.

Appendix C: Questionnaire of usability

Dear participant,

The aim for conducting the research is to investigate the situation of the iPlayCode and SC UIs which are designed by new model in order to measure their usability. The purpose of this questionnaire is to validate the model which has been built.

The information you give will be entirely confidential and will not be shared with any people not directly connected with this research. Please answer honestly and as accurately as you can. Your contribution is much appreciated.

Thank you very much for your assistance and cooperation

Please choose the one most appropriate response to each statement:

Personal information

Please select your gender:

- ☐ Male
- ☐ Female

Please select your age range:

- ☐ 18-34
- ☐ 35-49
- ☐ 50-60

Please select your education:

- ☐ High school diploma
- ☐ Undergraduate degree
- ☐ Postgraduate degree

Please select your experience years:

- ☐ Less than 5 years
- ☐ 6-12 years
- ☐ More than 12 yea

NEXT

Mobile quiz game prototype usability evaluation

1- Overall, I am satisfied with how easy it is to use this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

2- It was simple to use this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

3- I can effectively complete my work using this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

4- I am able to complete my work quickly using this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

5- I am able to efficiently complete my work using this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

6- I feel comfortable using this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

7- It was easy to learn to use this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

8- I believe I became productive quickly using this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

9- The interface gives error messages that clearly tell me how to fix problems:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

10- Whenever I make a mistake using the interface, I recover easily:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

11- The information provided with this interface is clear:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

12- It is easy to find the information I needed:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

13- The information provided for the interface is easy to understand:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

14- The information is effective in helping me complete the tasks and scenarios:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

15- The organization of information on the interface screens is clear:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

16- The interface is pleasant:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

17- I like using the interface of this mobile application:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

18- This interface has all the functions and capabilities I expect it to have:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

19- Overall, I am satisfied with this interface:

	1	2	3	4	5	
strongly disagree	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	strongly agree

What do you think about the App's UI navigation?

Your answer

When using the App's interface, what part was difficult to navigate?

Your answer

What do you suggest to be added/changed in the UI's App?

Your answer

What is your personal comments about the iPlayCode App's interface?

Your answer

BACK

SUBMIT

Appendix D: Comparative analysis of UI design

D.1 Hierarchical structure designs for iPlayCode and existing apps

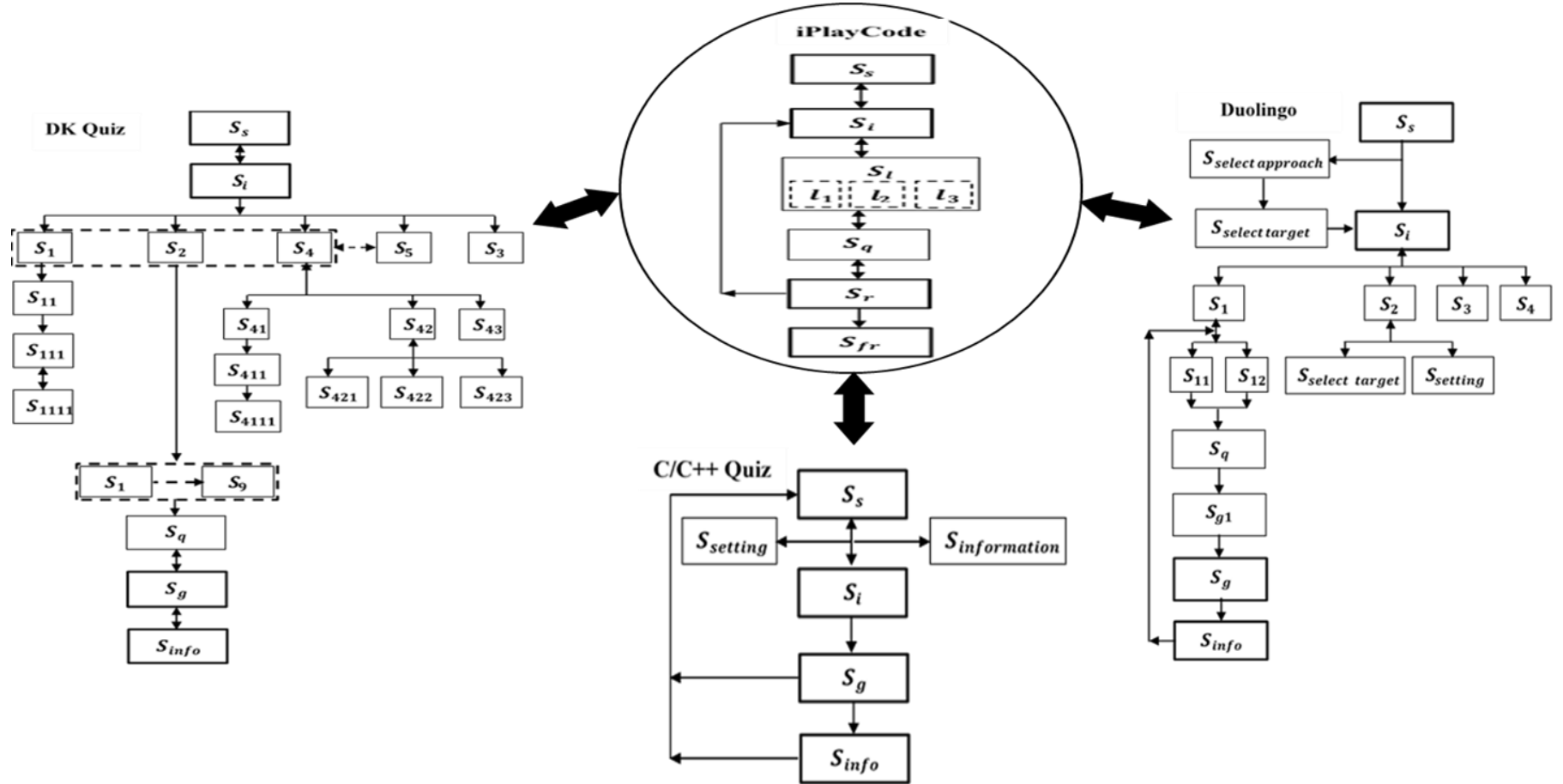


Figure D.1: Comparison of hierarchical structure design of iPlayCode with other mobile applications.

D.2 Hierarchical structure designs for SC and existing apps

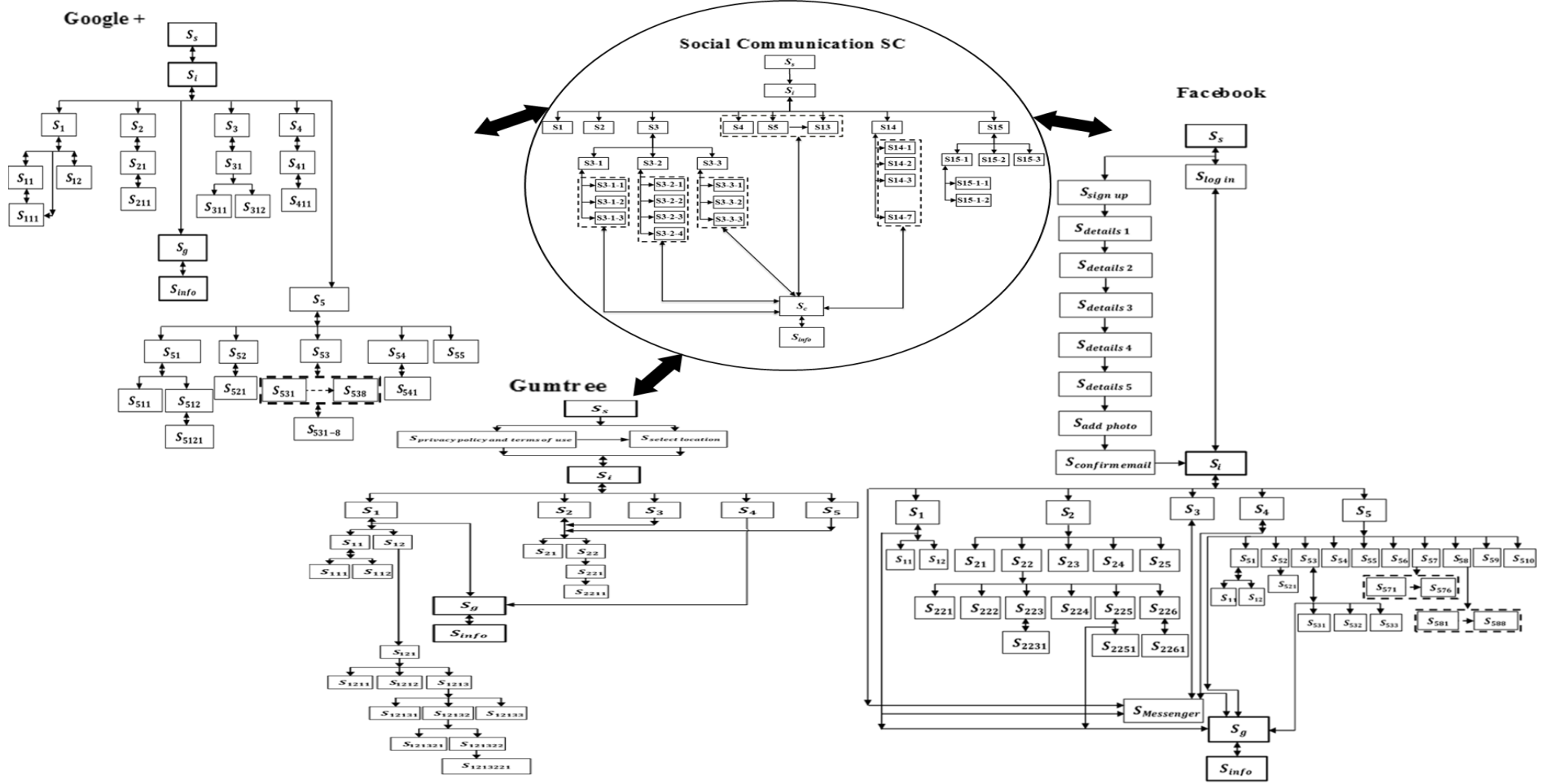


Figure D.2: Comparison of hierarchical structure design of SC with other mobile applications.

Appendix E: Comparative Analysis of UI elements

E.1 Comparative analysis of single screens for quiz game applications

Table E:1: The comparison of UI elements on single screens for quiz game apps.

Variable Screens	iPlayCode	DK Quiz	Duolingo	C/C++ quiz
S_s	1 text view 3 image view 2 buttons 1 text Field 7 Colors 5 functions	1 text view 1 image view 2 button 4 colors 2 function	1 text view 2 image view 2 buttons 3 colors 3 functions	1 text view 4 button 7 colors 4 functions
S_i	1 text view 7 buttons 6 colors 7 function	11 text view 1 image view 12 buttons 1 tab bar 12 colors 16 functions	62 text view 2 image view 62 buttons 1 tab bar 5 colors 65 functions	8 text view 5 buttons 4 colors 7 functions
S_g	2 text view 2 image view 4 buttons 7 colors 7 functions	3 text view 1 image view 2 button 1 rating bar 9 colors 5 functions	1 text view 1 image view 2 buttons 6 colors 3 functions	4 text view 2 buttons 8 colors 4 functions
S_{info}	4 text view 3 image view 1 button 7 colors 9 functions	5 text view 1 image view 6 buttons 2 rating bar 7 colors 10 functions	2 text view 1 image view 1 button 1 table 5 color 2 functions	19 text view 1 button 1 table 9 colors 3 functions

Table E:2: Comparing different average categories of single screens for quiz game apps.

Model screens	Category	iPlayCode	DK	Duolingo	C/C++
S_s	Control	1.5	2	2	4
	Vision	2	1	1.5	1
	Content	0	0	0	0
	Navigation bar	0	0	0	0
	Colour	7	4	3	7
	Function	5	2	3	4
S_i	Control	7	12	62	5
	Vision	1	6	32	8
	Content	0	0	0	0
	Navigation bar	0	1	1	0
	Colour	6	12	5	4
	Function	7	16	65	7
S_g	Control	4	2	2	2
	Vision	2	2	1	4
	Content	0	1	0	0
	Navigation bar	0	0	0	0
	Colour	7	9	6	8
	Function	7	5	3	4
S_{info}	Control	1	6	1	1
	Vision	3.5	3	1.5	19
	Content	0	2	1	1
	Navigation bar	0	0	0	0
	Colour	7	7	5	9
	Function	9	10	2	3

E.2 Comparative analysis of single screens for social media applications

Table E:3: The comparison of UI elements on single screens for social media apps.

Screens	SC	Google+	Facebook	Gumtree
S_s	2 text view 1 image view 1 button 3 color 1 function	2 text view 1 image view 1 button 6 colors 1 function	2 text view 1 image view 4 buttons 2 colors 6 functions	7 text view 2 image view 3 buttons 1 search bar 2 collection view 1 tab bar 5 colors 12 functions
S_i	3 image view 12 buttons 1 table 3 colors 15 functions	3 text view 3 image view 9 buttons 1 collection view 1 tab bar 3 colors 19 functions	4 text view 1 image view 12 buttons 1 video view 1 search bar 1 tab bar 5 colors 22 functions	4 text view 2 image view 2 buttons 2 collection view 1 search bar 5 colors 6 functions
S_g	1 image view 2 buttons 1 search bar 1 table 3 colors 2 functions	9 Buttons 1 search bar 1 default cell style 1 collection view 5 colors 16 functions	4 text view 2 image view 9 buttons 1 default cell style 1 segmented control 1 video view 5 colors 19 functions	4 buttons 1 default cell style 1 segmented control 7 colors 10 functions
S_{info}	2 image view 1 button 1 table 3 colors 1 function	4 text view 2 image view 7 buttons 1 video view 1 default cell style 4 colors 12 functions	1 text view 3 image view 14 buttons 1 segmented control 5 colors 24 functions	6 text view 1 image view 10 buttons 1 table 6 colors 12 functions

Table E:4: Comparing different average categories of single screens for social media apps.

Model screens	Category	SC	Google+	Facebook	Gumtree
S_s	Control	1	1	4	3
	Vision	1.5	1.5	1.5	3
	Content	0	0	0	0
	Navigation bar	0	0	0	0
	Colour	3	6	2	5
	Function	1	1	6	12
S_i	Control	12	9	12	2
	Vision	3	2.33	1.75	2.25
	Content	1	0	0	0
	Navigation bar	0	1	1	0
	Colour	3	3	5	5
	Function	15	19	22	6
S_g	Control	2	9	5	2.5
	Vision	1	0	2	1
	Content	1	0	0	0
	Navigation bar	0	0	0	0
	Colour	3	5	5	7
	Function	2	16	19	10
S_{info}	Control	1	7	7.5	10
	Vision	2	2	2	3.5
	Content	1	0	0	1
	Navigation bar	0	0	0	0
	Colour	3	4	5	6
	Function	1	12	24	12

E.3 Comparative analysis of multiple screens for quiz game applications

Table E:5: Comparing different categories of multiple screens for quiz game apps.

Category		iPlayCode	DK	Duolingo	C/C++
Control					
1	Button	25	71	99	15
2	Check box	0	0	1	0
3	Radio Button	0	0	3	0
4	Text field	1	3	7	0
5	Progress indicators	1	1	1	0
6	Segmented	0	1	0	0
7	Switch	0	1	6	1
Total		3.86	11	16.7	2.3
Vision					
1	Text View	34	39	127	27
2	Image View	9	17	22	0
3	Video view	0	0	1	0
4	Default cell style	0	1	1	0
Total		10.75	14.25	37.75	6.75
Content					
1	Screen	6	30	16	6
2	Table	0	1	1	0
3	Charts	0	0	0	1
4	Alert	1	1	2	1
5	Rating bar	0	4	0	0
Total		1.4	7.2	3.8	1.6
Navigation					
1	Navigation bar	1	1	1	1
2	Tab bar	0	1	1	0
Total		0.5	1	1	0.5
Colour		16	10	19	14
Total		14.75			
Function		36	78	109	21
Total		61			

E.4 Comparative analysis of multiple screens for social media applications

Table E:6: Comparing different categories of multiple screens for social media apps.

Item		SC	Google+	Facebook	Gumtre
Control					
1	Button	29	74	96	53
2	Check box	0	5	2	1
3	Radio Button	0	0	7	1
4	Text field	0	28	17	13
5	Segmented	0	2	3	1
6	Switch	0	34	7	5
7	Data Picker	0	0	2	0
8	Picker	0	0	3	0
Total		3.6	17.9	17.1	9.3
Vision					
1	Text View	11	40	52	24
2	Image View	15	15	15	10
3	Video View	0	1	1	0
4	Activity View	0	1	0	1
5	Collection View	0	2	9	4
6	Default Cell Styles	0	3	12	1
Total		4.3	10.2	14.8	6.3
Content					
1	Screen	44	39	65	27
2	Table	14	1	0	2
3	Action Sheet	0	1	16	0
4	Alert	2	5	1	0
Total		15	11.5	20.5	7.3
Navigation					
1	Navigation bar	1	1	1	1
2	Tool bar	0	1	0	0
3	Tab Bar	0	1	1	1
4	Menu	0	9	12	39
5	Search Bar	1	2	5	3
Total		0.4	2.8	3.8	8.8
Colour		5	7	6	8
Total		6.5			
Function		35	141	128	60
Total		91			

Appendix F: One-way ANOVA test results for iPlayCode, DK, Duolingo and C/C++

A- single screen

1- S_s screen

Table F:1: ANOVA test of S_s screen for iPlayCode and other apps.

SUMMARY						
Groups	Count	Sum	Average	Variance		
iPlayCode	6	15.5	2.583333	8.041667		
DK	6	9	1.5	2.3		
Duolingo	6	9.5	1.583333	1.841667		
C/C++	6	16	2.666667	7.866667		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	7.083333	3	2.361111	0.471	0.706	3.098391
Within Groups	100.25	20	5.0125			
Total	107.3333	23				

2- S_i screen

Table F:2: ANOVA test of S_i screen for iPlayCode and other apps.

SUMMARY						
Groups	Count	Sum	Average	Variance		
iPlayCode	6	21	3.5	12.3		
DK	6	47	7.833333	42.56667		
Duolingo	6	165	27.5	916.3		
C/C++	6	24	4	11.6		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	2323.125	3	774.375	3.152	0.048	3.098391
Within Groups	4913.833	20	245.6917			
Total	7236.958	23				

3- S_g screen

Table F:3: ANOVA test of S_g screen for iPlayCode and other apps.

SUMMARY

Groups	Count	Sum	Average	Variance
iPlayCode	6	20	3.333333	10.26667
DK	6	19	3.166667	10.96667
Duolingo	6	12	2	5.2
C/C++	6	18	3	9.2

ANOVA

Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	6.458333	3	2.152778	0.242	0.866	3.098391
Within Groups	178.1667	20	8.908333			
Total	184.625	23				

4- S_{info} screen

Table F:4: ANOVA test of S_{info} screen for iPlayCode and other apps.

SUMMARY						
Groups	Count	Sum	Average	Variance		
iPlayCode	6	20.5	3.416667	14.64167		
DK	6	28	4.666667	13.46667		
Duolingo	6	10.5	1.75	2.975		
C/C++	6	33	5.5	54.3		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	47.91667	3	15.97222	0.748	0.536	3.098391
Within Groups	426.9167	20	21.34583			
Total	474.8333	23				

➤ t-test and f-test Results for S_i Screen

1- t-test results for S_i

Table F:5: t-test results for S_i screen between iPlayCode and DK apps.

t-test: Paired Two Sample for Means

	<i>iPlayCode</i>	<i>DK</i>
Mean	3.5	7.833333
Variance	12.3	42.56667
Observations	6	6
Pearson Correlation	0.957099	
Hypothesized Mean Difference	0	
df	5	
t Stat	-3.19072	
P(T<=t) one-tail	0.012123	
t Critical one-tail	2.015048	
P(T<=t) two-tail	0.024	
t Critical two-tail	2.570582	

Table F:6: t-test results for S_i screen between iPlayCode and Duolingo apps.

t-test: Paired Two Sample for Means

	<i>iPlayCode</i>	<i>Duolingo</i>
Mean	3.5	27.5
Variance	12.3	916.3
Observations	6	6
Pearson Correlation	0.703638	
Hypothesized Mean Difference	0	
df	5	
t Stat	-2.10602	
P(T<=t) one-tail	0.044535	
t Critical one-tail	2.015048	
P(T<=t) two-tail	0.089	
t Critical two-tail	2.570582	

Table F:7: t-test results for S_i screen between iPlayCode and C/C++ apps.

t-test: Paired Two Sample for Means

	<i>iPlayCode</i>	<i>C/C++</i>
Mean	3.5	4
Variance	12.3	11.6
Observations	6	6
Pearson Correlation	0.535795	
Hypothesized Mean Difference	0	
df	5	
t Stat	-0.36761	
P(T<=t) one-tail	0.364107	
t Critical one-tail	2.015048	
P(T<=t) two-tail	0.728	
t Critical two-tail	2.570582	

Table F:8: t-test results for S_i screen between DK and Duolingo apps.

t-test: Paired Two Sample for Means

	<i>DK</i>	<i>Duolingo</i>
Mean	7.833333	27.5
Variance	42.56667	916.3
Observations	6	6
Pearson Correlation	0.753947	
Hypothesized Mean Difference	0	
df	5	
t Stat	-1.87363	
P(T<=t) one-tail	0.059927	
t Critical one-tail	2.015048	
P(T<=t) two-tail	0.120	
t Critical two-tail	2.570582	

Table F:9: t-test results for S_i screen between DK and C/C++ apps.

t-test: Paired Two Sample for Means

	<i>DK</i>	<i>C/C++</i>
Mean	7.833333	4
Variance	42.56667	11.6
Observations	6	6
Pearson Correlation	0.720039	
Hypothesized Mean Difference	0	
df	5	
t Stat	1.994353	
P(T<=t) one-tail	0.051339	
t Critical one-tail	2.015048	
P(T<=t) two-tail	0.103	
t Critical two-tail	2.570582	

Table F:10: t-test results for S_i screen between Duolingo and C/C++ apps.

t-test: Paired Two Sample for Means

	<i>Duolingo</i>	<i>C/C++</i>
Mean	27.5	4
Variance	916.3	11.6
Observations	6	6
Pearson Correlation	0.739107	
Hypothesized Mean Difference	0	
df	5	
t Stat	2.067058	
P(T<=t) one-tail	0.046795	
t Critical one-tail	2.015048	
P(T<=t) two-tail	0.094	
t Critical two-tail	2.570582	

➤ **f-test results between iPlayCode and Dk apps for Si screen**

Table F:11: f-test results between iPlayCode and Dk apps for S_i screen.

F-test Two-Sample for Variances

	<i>DK</i>	<i>iPlayCode</i>
Mean	7.833333	3.5
Variance	42.56667	12.3
Observations	6	6
df	5	5
F	3.460705	
P(F<=f) one-tail	0.10	
F Critical one-tail	5.050329	

B- Multiple-screens

Table F:12: ANOVA test of multiple-screens for iPlayCode and other apps.

SUMMARY						
Groups	Count	Sum	Average	Variance		
iPlayCode	6	68.50714	11.41786	180.4891		
DK	6	121.45	20.24167	820.5104		
Duolingo	6	187.2643	31.21071	1623.444		
C/C++	6	46.13571	7.689286	67.16926		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	1971.988	3	657.3295	0.977	0.423	3.098391
Within Groups	13458.06	20	672.9031			
Total	15430.05	23				

Appendix G: One-way ANOVA test results for SC, Google+, Facebook and Guntree

A- single screen

1- S_s screen

Table G:1: ANOVA test of S_s screen for SC and other apps.

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
SC	6	6.5	1.083333	1.241667
Google+	6	9.5	1.583333	5.041667
Facebook	6	13.5	2.25	5.575
Guntree	6	24.7	4.116667	18.20167

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	31.73833	3	10.57944	1.408	0.270	3.098391
Within Groups	150.3	20	7.515			
Total	182.0383	23				

2- S_i screen

Table G:2: ANOVA test of S_i screen for SC and other apps.

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
SC	6	34	5.666667	39.06667
Google+	6	34.33	5.721667	52.20082
Facebook	6	42	7	72.8
Guntree	6	16.7	2.783333	5.361667

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	57.21395	3	19.07132	0.450	0.720	3.098391
Within Groups	847.1458	20	42.35729			
Total	904.3597	23				

3- S_g screen

Table G:3: ANOVA test of S_g screen for SC and other apps.

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
SC	6	9	1.5	1.1
Google+	6	30	5	42.4
Facebook	6	31	5.166667	50.96667
Gumtree	6	20.5	3.416667	17.24167

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	52.44792	3	17.48264	0.626	0.607	3.098391
Within Groups	558.5417	20	27.92708			
Total	610.9896	23				

4- S_{info} screen

Table G:4: ANOVA test of S_{info} screen for SC and other apps.

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
SC	6	8	1.333333	1.066667
Google+	6	25	4.166667	21.76667
Facebook	6	38.5	6.416667	82.84167
Gumtree	6	32.5	5.416667	23.44167

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	87.25	3	29.08333	0.901	0.458	3.098391
Within Groups	645.5833	20	32.27917			
Total	732.8333	23				

B- Multiple- screen

Table G:5: ANOVA test of multiple-screens for SC and other apps.

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
SC	6	63.36333	10.56056	167.5925
Google+	6	190.3467	31.72444	2890.901
Facebook	6	190.2633	31.71056	2266.96
Gumtree	6	99.63333	16.60556	453.0419

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	2082.775	3	694.2584	0.480581	0.699	3.098391
Within Groups	28892.47	20	1444.624			
Total	30975.25	23				

Appendix H: One-way ANOVA test analysis for questionnaire

This appendix presents the statistical technique and its output. The output results are used for testing all hypotheses described in Chapter 1.

A- One-way ANOVA test results for iPlayCode and other apps

1- Usefulness attribute

Table H:1: Usefulness attribute results.

SUMMARY						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
iPlayCode	7	30.17742	4.31106	0.016971		
DK	7	29.75806	4.251152	0.044126		
Duolingo	7	29.46774	4.209677	0.054024		
C-C++	7	29.46774	4.209677	0.044918		
ANOVA						
<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	0.048276	3	0.016092	0.402198	0.752704	3.008787
Within Groups	0.960235	24	0.04001			
Total	1.00851	27				

2- Information quality attribute

Table H:2: Information quality attribute results.

Anova: Single Factor

SUMMARY						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
iPlayCode	6	26.25806	4.376344	0.006417		
DK	6	25.67742	4.27957	0.023795		
Duolingo	6	26.06452	4.344086	0.017135		
C-C++	6	25.77419	4.295699	0.028477		
ANOVA						
<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	0.03551	3	0.011837	0.624428	0.607521	3.098391
Within Groups	0.379119	20	0.018956			
Total	0.414629	23				

3- Interface quality attribute

Table H:3: Interface quality attribute results.

Anova: Single Factor

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
iPlayCode	4	17.35484	4.33871	0.02428
DK	4	17.30645	4.326613	0.035618
Duolingo	4	17.22581	4.306452	0.021332
C-C++	4	16.48387	4.120968	0.021072

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	0.125699	3	0.0419	1.638271	0.232731	3.490295
Within Groups	0.306907	12	0.025576			
Total	0.432606	15				

4- Overall satisfaction attribute

Table H:4: Overall satisfaction attribute results.

Anova: Single Factor

SUMMARY

<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>
iPlayCode	2	8.774194	4.387097	0.008325
DK	2	8.5	4.25	0.081296
Duolingo	2	8.451613	4.225806	0.074922
C-C++	2	8.354839	4.177419	0.008325

ANOVA

<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	0.048485	3	0.016162	0.373965	0.777397	6.591382
Within Groups	0.172867	4	0.043217			
Total	0.221351	7				

B- One-way ANOVA test results for SC and other apps

1- Usefulness attribute

Table H:5: Usefulness attribute results.

SUMMARY						
Groups	Count	Sum	Average	Variance		
SC	7	30.54839	4.364055	0.003803		
Google+	7	30.46774	4.352535	0.002725		
Facebook	7	30.41935	4.345622	0.002589		
Gumtree	7	30.66129	4.380184	0.004063		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	0.004794	3	0.001598	0.484962	0.695888	3.008787
Within Groups	0.079084	24	0.003295			
Total	0.083878	27				

2- Information quality attribute

Table H:6: Information quality attribute results.

SUMMARY						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
SC	6	26.09677	4.349462	0.005793		
Google+	6	26.1129	4.352151	0.006703		
Facebook	6	26.1129	4.352151	0.004102		
Gumtree	6	26.19355	4.365591	0.006417		
ANOVA						
<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	0.000954	3	0.000318	0.055262	0.98241	3.098391
Within Groups	0.115071	20	0.005754			
Total	0.116025	23				

3- Interface quality attribute

Table H:7: Interface quality attribute results.

SUMMARY						
Groups	Count	Sum	Average	Variance		
SC	4	17.59677	4.399194	0.003013		
Google+	4	17.5	4.375	0.003707		
Facebook	4	17.40323	4.350806	0.005441		
Gumtree	4	17.74194	4.435484	0.004162		
ANOVA						
Source of Variation	SS	df	MS	F	P-value	F crit
Between Groups	0.015658	3	0.005219	1.278884	0.325995	3.490295
Within Groups	0.048972	12	0.004081			
Total	0.06463	15				

4- Overall satisfaction attribute

Table H:8: Overall satisfaction attribute results.

SUMMARY						
<i>Groups</i>	<i>Count</i>	<i>Sum</i>	<i>Average</i>	<i>Variance</i>		
SC	2	8.709677	4.354839	0.008325		
Google+	2	8.66129	4.330645	0.010536		
Facebook	2	8.725806	4.362903	0.003252		
Gumtree	2	8.822581	4.41129	0.021982		
ANOVA						
<i>Source of Variation</i>	<i>SS</i>	<i>df</i>	<i>MS</i>	<i>F</i>	<i>P-value</i>	<i>F crit</i>
Between Groups	0.006861	3	0.002287	0.207473	0.886453353	6.591382
Within Groups	0.044095	4	0.011024			
Total	0.050956	7				

C- Normal data distribution

Table H:9: Normal data distribution for iPlayCode app.

Mean	4.346774
Standard Error	0.027546
Median	4.370968
Mode	4.451613
Standard Deviation	0.116866
Sample Variance	0.013658
Kurtosis	-0.51736
Skewness	-0.72648
Range	0.370968
Minimum	4.112903
Maximum	4.483871
Sum	78.24194
Count	18

Table H:10: Normal data distribution for DK app.

Mean	4.265233
Standard Error	0.042358
Median	4.314516
Mode	4.451613
Standard Deviation	0.179709
Sample Variance	0.032295
Kurtosis	-1.8245
Skewness	-0.24972
Range	0.467742
Minimum	4
Maximum	4.467742
Sum	76.77419
Count	18

Table H:11: Normal data distribution for Duolingo app.

Mean	4.263441
Standard Error	0.043404
Median	4.346774
Mode	4.451613
Standard Deviation	0.184149
Sample Variance	0.033911
Kurtosis	-1.7822
Skewness	-0.34882
Range	0.451613
Minimum	4
Maximum	4.451613
Sum	76.74194
Count	18

Table H:12: Normal data distribution for C-C++ app.

Mean	4.202509
Standard Error	0.040804
Median	4.137097
Mode	4.032258
Standard Deviation	0.173117
Sample Variance	0.029969
Kurtosis	-1.68736
Skewness	0.323087
Range	0.467742
Minimum	4
Maximum	4.467742
Sum	75.64516
Count	18

Table H:13: Normal data distribution for SC app.

Mean	4.365874
Standard Error	0.014855
Median	4.370968
Mode	4.33871
Standard Deviation	0.064752
Sample Variance	0.004193
Kurtosis	-1.11137
Skewness	-0.31147
Range	0.209677
Minimum	4.241935
Maximum	4.451613
Sum	82.95161
Count	19

Table H:14: Normal data distribution for Google+ app.

Mean	4.354839
Standard Error	0.01475
Median	4.387097
Mode	4.387097
Standard Deviation	0.064292
Sample Variance	0.004133
Kurtosis	-1.00275
Skewness	-0.17059
Range	0.225806
Minimum	4.241935
Maximum	4.467742
Sum	82.74194
Count	19

Table H:15: Normal data distribution for Facebook app.

Mean	4.350594
Standard Error	0.012809
Median	4.33871
Mode	4.33871
Standard Deviation	0.055832
Sample Variance	0.003117
Kurtosis	-0.82121
Skewness	-0.03949
Range	0.177419
Minimum	4.258065
Maximum	4.435484
Sum	82.66129
Count	19

Table H:16: Normal data distribution for Guntree app.

Mean	4.390492
Standard Error	0.017468
Median	4.387097
Mode	4.467742
Standard Deviation	0.076143
Sample Variance	0.005798
Kurtosis	-1.4001
Skewness	0.029779
Range	0.241935
Minimum	4.274194
Maximum	4.516129
Sum	83.41935
Count	19