

CRANFIELD UNIVERSITY

SANATHANAN RAJAGOPAL

DEVELOPMENT OF A FRAMEWORK TO ESTIMATE THE
SOFTWARE OBSOLESCENCE RESOLUTION COST OF CUSTOM
BUILD REAL-TIME SOFTWARE

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING
Doctor Of Philosophy

PhD Thesis
Academic Year: 2013 - 2022

Supervisor: Prof. John Ahmet Erkoyuncu
Associate Supervisor: Prof. Rajkumar Roy
January 2022

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT AND
MANUFACTURING
Doctor Of Philosophy

PhD Thesis

Academic Year 2013 - 2022

SANATHANAN RAJAGOPAL

DEVELOPMENT OF A FRAMEWORK TO ESTIMATE THE
SOFTWARE OBSOLESCENCE RESOLUTION COST OF CUSTOM
BUILD REAL-TIME SOFTWARE

Supervisor: Prof. John Ahmet Erkoyuncu
Associate Supervisor: Prof. Rajkumar Roy
January 2022

This thesis is submitted in partial fulfilment of the requirements for
the degree of Doctor of Philosophy

© Cranfield University 2022. All rights reserved. No part of this
publication may be reproduced without the written permission of the
copyright owner.

ABSTRACT

Software plays an important role in the defence sector. Almost every project in defence has got software with various degrees of complexity and dependencies. Software obsolescence is one of the key cost drivers which has historically been assessed as part of the maintenance cost due to the lack of data/information and due to the fact that there are no tools, techniques, or framework to help with the software obsolescence management. Therefore it is key for the industry and Defence sector to address software obsolescence and to find an approachable way to counter the effect of software obsolescence.

The aim of this research is to develop a software obsolescence cost analysis framework to predict the software obsolescence resolution cost of custom build real time software at early stages of the product life cycle. For the accomplishment of the aim, an extensive literature review on software obsolescence and its related themes was undertaken, a systematic case study with online survey corroborated the findings and revealed the current practices in the UK Defence sector. Further collaboration with 35 experts from industries led to undertaking a cognitive case study to enable the iterative development of Software Obsolescence Cost Analysis Framework. This was demonstrated by applying the framework to five case studies as part of expert validation.

The novelty of this research is that it tries to solve an issue with significant real-world impact for which very little work exists with lots of real-world inputs from SME and real case studies. Cognitive case study with software experts is unique and has never attempted before to solve this issue. The research comes up with a novel way to quantify the costs for the very first time with strong input from primary data sources. The key limitation of the research is in the unavailability of any historic technical data.

Keywords:

Software Obsolescence, Software Obsolescence Management, Software Obsolescence Resolutions, DMSMS

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my gratitude to all the people who have helped and supported me throughout this project and to all those who collaborated on this project.

I must sincerely express my sincere thank you to my supervisors Professor John Ahmet Erkoyuncu and Professor Rajkumar Roy, and my industry liaison officer Professor Andrew Langridge. Their advice, constructive criticisms, guidance, and patience have been the driving force for the success of this thesis.

It has been a great pleasure working with the staff and colleagues at the School of Aerospace, Transport and Manufacturing Department and their contribution has made this research a rewarding experience. I am grateful to Teresa Bandee and SASSATM Research admin team who has helped and supported me with all the necessary admin requirements throughout my research time in Cranfield.

I am thankful to Dr Tim Sheldon, Head of CAAS (Cost Assurance and Analysis Services), MoD (UK) who approved the funding for this research and for his guidance in the early years of my research, and Dale Shermon, QinetiQ UK who helped secure funding from QinetiQ UK for further progression of this research. I am grateful to all the organisations that collaborated in this research especially MoD DE&S, CAAS, QinetiQ, BAE Systems, ICEAA, SCAF and IOM.

I would like to express gratitude to Professor Peter Sandborn whose guidance on software obsolescence at the early stages of the research was key to the success of this research. My sincere gratitude to members of IOM who supported and collaborated in this study. I am very grateful to John Yeaman, who mentored, guided, and supported me throughout this research.

And finally, I am very grateful to my wife Viji Viswanath for her ongoing support, patience, and love during my time in Cranfield, without which the delivery of this thesis would have been impossible.

Sanathanan Rajagopal

TABLE OF CONTENTS

ABSTRACT	i
ACKNOWLEDGEMENTS.....	iii
LIST OF FIGURES.....	ix
LIST OF TABLES	xiii
LIST OF EQUATIONS.....	xvi
LIST OF ABBREVIATIONS	xvii
1 Research Introduction	21
1.1 Introduction	21
1.2 Research Aims	24
1.3 Current Practice and Challenges	25
1.4 Definitions	27
1.4.1 Bespoke and COTS (Commercial off the shelf) Software	27
1.4.2 Type of software.....	27
1.4.3 Software Maintenance.....	29
1.4.4 Software Obsolescence Mitigation Vs Software Obsolescence Resolution	29
1.4.5 Systems	29
1.5 Scope and out of scope of this research.....	29
1.6 Research Objectives.....	30
1.7 Contribution to knowledge	30
1.8 The Collaborating Organisation	32
1.9 Thesis Structure and Summary	37
1.10 Summary of Observation	39
2 Literature Review	41
2.1 Introduction	41
2.2 Literature Review Strategy.....	41
2.3 Software Obsolescence	44
2.4 Cost Estimate	55
2.4.1 Cost Estimating Techniques.....	58
2.4.2 Software Cost Estimating	61
2.4.3 Software Support Cost Estimate	62
2.5 Key findings from the literature review and research gaps	67
3 Research Methodology	70
3.1 Introduction	70
3.2 Research Aim and Objectives.....	70
3.3 Research Context	71
3.4 Research Purpose	71
3.5 Research Philosophy	72
3.5.1 Qualitative Research	72
3.5.2 Quantitative Research.....	75

3.5.3 Approaches in Qualitative Research	78
3.5.4 Research Evaluations	88
3.6 Research Methodology Adopted.....	88
3.6.1 Selection of Research Approach.....	88
3.6.2 Research Methodology Adopted	89
3.7 Summary of Observation	96
4 Current Practices.....	99
4.1 Introduction	99
4.2 Detailed Research Methodology.....	99
4.2.1 SME Interview.....	99
4.2.2 Case Studies.....	107
4.2.3 Online Survey.....	115
4.3 Summary of Observation	123
5 Cognitive Case Study.....	127
5.1 Introductions	127
5.2 Research Methodology.....	130
5.3 The Cognitive Case Study Experiment	131
5.3.1 Designing the experiments.....	133
5.3.2 Defining the experiment	133
5.3.3 Defining the hypothesis.....	133
5.3.4 Presenting the information and conducting the experiments.....	134
5.3.5 Case Study Analysis	134
5.4 The Experiment	135
5.5 Cognitive Case Study Coding Scheme	149
5.5.1 Micro Strategies	149
5.5.2 Macro Strategies	150
5.5.3 Key Performance Indicators	150
5.6 Cognitive Case Study Analysis	152
5.6.1 The Analysis.....	153
5.6.2 Common Approaches and Differences.....	156
5.6.3 Detailed Analysis.....	158
5.7 Summary of Observation	161
6 Software Obsolescence Cost Analysis Framework	163
6.1 Introduction	163
6.2 Research Methodology for the initial study and assumptions	164
6.3 Detailed Research methodology.....	168
6.3.1 Software obsolescence cost analysis framework.....	170
6.3.2 Software Obsolescence Management Level.....	174
6.3.3 Software Obsolescence Resolution Approach	180
6.3.4 Software obsolescence complexity level.....	191
6.3.5 Software obsolescence key cost drivers	196
6.4 Summary of observations	202

7 Software Obsolescence Cost Analysis Model	205
7.1 Introduction	205
7.2 Detailed Research Methodology	206
7.3 Data Collection Model.....	209
7.3.1 Phase 1 Data Collection.....	211
7.3.2 Phase 2 –Data Collection Model Refinement.....	212
7.3.3 Phase 3 – Data Collection Model Validation	212
7.4 Software Obsolescence Resolution Model	226
7.4.1 Phase 1 Data Collection.....	228
7.4.2 Phase 2 –Software Obsolescence Resolution Model Refinement .	229
7.4.3 Phase 3 – Software Obsolescence Resolution Model Validation ...	229
7.5 Software Obsolescence Cost Model	234
7.6 Software Obsolescence Resolution Project Cost.....	238
7.7 Summary of Observation	241
8 Demonstration	243
8.1 Introduction	243
8.2 Detailed Research Methodology for Software Obsolescence demonstration	243
8.2.1 Summary of the feedback from the demonstration workshop	249
8.3 Software Obsolescence Cost Analysis Framework Demonstration Case Study.	257
8.3.1 Questionnaire Development.....	259
8.3.2 Case Study –Project selections.....	260
8.3.3 Software Obsolescence Cost Analysis Framework Demonstration Case Study.....	262
8.3.4 Case Study Summary	283
8.4 Cross- Case Study Analysis	283
8.4.1 Detailed analysis of the feedback.....	288
8.5 Summary of Observation	298
9 Conclusion.....	299
9.1 Introduction	299
9.2 Research Limitations	302
9.2.1 Scope	302
9.2.2 Access.....	304
9.2.3 Implementation challenge	305
9.2.4 Other limitations	305
9.3 Key Contribution	306
9.4 Future Research	307
REFERENCES.....	309
APPENDICES	319
Appendix A Interview Questions	319
Appendix B Detailed online survey data Analysis	322

Appendix C Source Code for Software Maintenance Database	346
Appendix D Questioner for Demonstration Feedback.....	350
Appendix E Case Study Demonstration Questioner	354
Appendix F The Coding Scheme for Cognitive Case study.	359
Appendix G The day rate escalation table	363

LIST OF FIGURES

Figure 1-1Types of software obsolescence (Javier Romero Rojo, 2011)	23
Figure 1-2 Mapping of Objectives to contribution to knowledge	32
Figure 1-3 Structure of this thesis.....	39
Figure 2-1 Number of publications (both Journals and conference papers) on software obsolescence that was reviewed.....	42
Figure 2-2 Reactive versus proactive strategy (adapted from BS62402)	47
Figure 2-3 Software obsolescence process overview (COG, 2007)	48
Figure 2-4 Hardware reliability curve	49
Figure 2-5 Hardware obsolescence and reliability graph (Javier Romero Rojo, 2007)	50
Figure 2-6 12 steps of estimating by (GAO, 2009)	56
Figure 2-7 Classification of estimating techniques (Niazi et al., 2006)	59
Figure 2-8 Estimating methodology at each stage of the product life cycle adapted from SCEBoK, 2020	60
Figure 3-1 Research approach selection.....	88
Figure 3-2 Research methodology used for this research.....	92
Figure 3-3 Summary of validation and verification.....	96
Figure 4-1 Research methodology followed for SME Interviews	100
Figure 4-2 Detailed analysis of the SME interviews using mind map	105
Figure 4-3 Research methodology followed for the Case studies	108
Figure 4-4 Flow chart to show how project team currently deals with software obsolescence.....	115
Figure 4-5 Research methodology for online survey	115
Figure 4-6 Role of participants	117
Figure 4-7 Response on Software obsolescence strategy	119
Figure 4-8 Monitoring Software Obsolescence.....	120
Figure 5-1 Cognitive case study approach	132
Figure 5-2 Software Development Database front-end	143
Figure 5-3 Software Development Database back end.....	144

Figure 5-4 Software Support Database back end.....	146
Figure 5-5 Software Support Database front-end.....	147
Figure 5-6 Technology selection and deployment.....	155
Figure 5-7 Common approaches and differences in micro and macro strategies adopted by participants.....	157
Figure 5-8 Common approaches and differences in the key performance indicators by the participants.	158
Figure 5-9 Time spent on solving the problem (software obsolescence which was pre-set for this experiment) by each participant and its breakdown.....	159
Figure 5-10 Time spent on selecting, refereeing to technology selection, and referring to software obsolescence by each participant.....	160
Figure 5-11 Technology selection and confirmation by each participant.	161
Figure 6-1 steps/study undertaken to develop the Software obsolescence adopted from (Javier Romero Rojo, 2007).....	164
Figure 6-2 How software obsolescence is managed currently in the industry	166
Figure 6-3 Identification of initial key cost drivers	167
Figure 6-4 Initial understanding and design to estimate the cost of software obsolescence resolutions	168
Figure 6-5 Iterative development of Software Obsolescence Cost Analysis Framework.....	169
Figure 6-6 Software Obsolescence Cost Analysis Framework.....	171
Figure 6-7 Approach undertaken to develop Software Obsolescence Management level	176
Figure 6-8 Software Obsolescence Management Level.....	180
Figure 6-9 Approach undertaken to develop Software Obsolescence Resolution approach.....	183
Figure 6-10 Software Obsolescence Resolution Approach	187
Figure 6-11 Approach undertaken to develop Software Obsolescence Complexity Level.....	192
Figure 6-12 Software Obsolescence Complexity Matrix.....	195
Figure 6-13 Software Obsolescence Key Cost Driver and Cost Elements	198
Figure 6-14 Data flow of the software obsolescence cost analysis framework	199
Figure 6-15 Steps followed to estimate the cost of software obsolescence resolutions (Adopted from Javier Romero Rojo et al, 2010)	201

Figure 7-1 Research methodology followed for the development of SOCAM	207
Figure 7-2 Research Methodology followed to develop Software Obsolescence Data Collection Model.....	210
Figure 7-3 Project data Form.....	214
Figure 7-4 System Information Form.....	215
Figure 7-5 Software sizing form.....	217
Figure 7-6 software development data form.....	218
Figure 7-7 Software Application Data form.....	219
Figure 7-8 Project schedule data form	220
Figure 7-9 Questions on software obsolescence management level	221
Figure 7-10 Software Obsolescence management level inputs form	222
Figure 7-11 Software Obsolescence resolution form.....	223
Figure 7-12 Software Obsolescence probability Matrix form	224
Figure 7-13 Software obsolescence complexity level form.....	225
Figure 7-14 Inputs and output relationships	226
Figure 7-15 Research methodology used to develop Software Obsolescence Resolution Model.....	227
Figure 7-16 Types of nodes used in the resolution model.....	230
Figure 7-17 Example of level one resolution decomposition	231
Figure 7-18 Shows further decomposition of the nodes.	232
Figure 7-19 Example case for the whole software obsolescence resolution profile	233
Figure 7-20 Working of the software obsolescence Cost Model.....	234
Figure 7-21 Resource Loading page of the model	235
Figure 7-22 Cost build sheets.....	236
Figure 7-23 Yearly cost profile.....	237
Figure 7-24 Variables and constraints added to the resolution model.....	238
Figure 8-1 Detailed research methodology for demonstration.....	244
Figure 8-2 Approach taken to complete the demonstration workshop.....	246
Figure 8-3 First iteration of the software obsolescence cost analysis framework	248

Figure 8-4 Software Obsolescence Management Levels	253
Figure 8-5 Software Obsolescence Resolution Approach	254
Figure 8-6 Software Obsolescence Complexity Levels	255
Figure 8-7 Software Obsolescence Key Cost Driver	256
Figure 8-8 Approach undertaken for case study demonstration.....	257
Figure 8-9 Research methodology for Case Study Demonstration	259
Figure 8-10 Software Obsolescence Complexity Matrix Case Study A.....	264
Figure 8-11 Resolution Approach for Application 1	266
Figure 8-12 Resolution Approach for Application 2	270
Figure 8-13 Software Obsolescence Complexity Matrix Case Study B	273
Figure 8-14 Software Obsolescence Complexity Matrix – Case Study C.....	276
Figure 8-15 Software Obsolescence Complexity Matrix -Case Study D.....	278
Figure 8-16 Software Obsolescence Complexity Matrix – Case Study	281
Figure 8-17 Approach taken at the Case Study Demonstration	284

LIST OF TABLES

Table 2-1 Classification of literatures reviewed	43
Table 2-2 Strength, weakness, and application of estimating technique (GAO, 2009)	60
Table 2-3 Commercial tools and models to estimate the cost of software maintenance	64
Table 3-1 Strength and weakness of Qualitative Research (Javier Romero Rojo, 2011)	73
Table 3-2 Differences between positivism and constructivism	74
Table 3-3 Strengths and weakness of quantitative research	76
Table 3-4 Comparison of qualitative and quantitative research based on assumptions, purpose, methods and role	76
Table 3-5 Research strategy and how this was used in this research	78
Table 3-6 Strengths and weakness of survey	83
Table 3-7 Strengths and weakness of interviews	86
Table 3-8 Phases undertaken for this research	89
Table 3-9 Process and tools that was developed as part of the Software Obsolescence Cost Analysis Framework	94
Table 4-1 Demography of experts who participated in the semi – structured interviews	101
Table 4-2 case study with participating project teams and the number of interviews	110
Table 4-3 Role and experience of case study participants	111
Table 4-4 Summaries of the key findings from the case study	112
Table 4-5 List of Websites where the survey was published	116
Table 4-6 Response to element covered in obsolescence management strategy	117
Table 4-7 Planning for Software Obsolescence	119
Table 5-1 Details of the cognitive case study participants	131
Table 5-2 Non-functional requirement	136
Table 5-3 Functional Requirements	137

Table 5-4 Summaries of technology selections used to develop the in-house software	141
Table 5-5 Cognitive Case Study tasking	153
Table 6-1 Details of the workshop participants.....	170
Table 6-2 Number of participants and years of experience	178
Table 7-1 Details of the participants	209
Table 7-2 Mapping of multipliers and factors.....	240
Table 7-3 summaries the key comments from the initial validation exercise are described	241
Table 8-1 Detail of experts who participated in the demonstration workshop.	245
Table 8-2 Summary of the key findings from the demonstration workshop	250
Table 8-3 Experience of experts who participated in the DMSMS workshop .	251
Table 8-4 Experience of experts who participated in ICEAA Workshop	252
Table 8-5 Software project maturity assessment.....	261
Table 8-6 Summary of Case Study A.....	263
Table 8-7 potential obsolescence issue identified within case study A.....	264
Table 8-8 Matrix for Case Study A	265
Table 8-9 Resource type, Rate, Duration and number of resources required for resolution of Application 1.....	267
Table 8-10 Total Resolution Cost for Application 1	268
Table 8-11 Resource type, Rate, Duration, and number of resources required for resolution of Application 2.....	270
Table 8-12 Total Resolution Cost for Application 2	270
Table 8-13 Summary of the Case Study B	272
Table 8-14 potential obsolescence issue identified within case study B.....	274
Table 8-15 Matrix for Case Study B	274
Table 8-16 Summary of Case Study C.....	275
Table 8-17 potential obsolescence issues identified within case study C.....	276
Table 8-18 Matrix for case study C.....	277
Table 8-19 Summary of Case Study D.....	278
Table 8-20 potential obsolescence issue identified within case study D	279

Table 8-21 Matrix for Case Study D	279
Table 8-22 Summary of Case Study E	280
Table 8-23 potential obsolescence issues identified within case study E	282
Table 8-24 Matrix for Case Study E	282
Table 8-25 Summary of the findings against the budget	283
Table 8-26 Roles and experience of experts	284
Table 8-27 Key summary of the findings	285

LIST OF EQUATIONS

Equation 7-1 Cost of each resolution 239

Equation 7-2 Total cost of Software Obsolescence..... 240

LIST OF ABBREVIATIONS

AIS	Automated Information Systems
BAE	British Aerospace
BoE	Basis of Estimate
BS	British Standard
CAAS	Cost Assurances and Analysis Services
CADMID	Concepts Assessment Demonstration Manufacturing In-service and Disposal
CEBoK	Cost Estimating Body of Knowledge
CER	Cost Estimating Relationship
CMMI	Capability Maturity Model Integration
COG	Component Obsolescence Group
COTS	Commercial off the shelves
DE&S	Defence Equipment and Support
DERA	Defence Evaluation and Research Agency
DII	Defence information infrastructure
DII(F)	Defence information infrastructure Future
DMSMS	Diminishing manufacturing sources and material shortages
DoD	Department of Defence
DSTL	Defence Science and Technology Laboratory
DTech	Director of Technology
EVM	Earned Value Management

FP	Function Point
FPGA	Field Programmable Gate Arrays
GAO	Government Audit Office
GEC	General Electric Company plc
GOTS	Government of the shelves
GUI	Graphic user interface
ICE	Independent cost estimate
ICEAA	International Cost Estimating and Analysis Associations
IE8	Internet Explorer 8
IEEE	Institute of Electrical and Electronics Engineers
IFPUG	International Function Point User Group
IOM	International Institute of Obsolescence Management
ISBSG	International software benchmarking standard group
ISO	International organisation for standardisation
ISPA	International Society of Parametric Analysis
ISS/ISTAR	Information, Systems and Software/Intelligence, surveillance, target acquisition and reconnaissance
JSP	Joint service publications
KPI	Key performance information
KPM	Key performance measures
LTPA	Long-Term Partnering Agreement
Max	Maximum

MCDM	Multi Criteria Decision Making
MES	Marconi Electronic Systems
Min	Minimum
ML	Most Likely
MoD	Ministry of Defence
MODAF	MoD architectural framework
NASA	National Aeronautics and Space Administration
NATO	North Atlantic Treaty Organisation
NPV	Net present value
NRE	Non-recurring engineering cost
OEM	Original Equipment Manufacturing
OMS	Obsolescence Management Strategy
ORG	Organisation
OSCAM	Operating and Support Cost Analysis Model
Q&A	Question and Answers
R&D	Research and Development
RaeS	Royal Aeronautical Society
Rx	Receiver
S/W Obs	Software Obsolescence
SCAF	Society of Cost Analysis and Framework
SCEBoK	Software Cost Estimating Body of Knowledge
SDSD	Software Development and support database

SEER	Software Evaluations and Estimation of Resources
SLOC	Source lines of codes
SME	Subject matter expert
SOCAF	Software Obsolescence cost analysis framework
SOCAM	Software Obsolescence cost analysis model
SOMS	Software Obsolescence Management Strategy
SQEP	Suitably Qualified Experience Personal
TOGAF	The open group architecture framework
Tx	Transmitter
UK	United Kingdom
URD	User requirement documents
USA	United states of America
WBS	Work breakdown structure

1 Research Introduction

1.1 Introduction

Software is defined as programmes, procedures, rules, data, and documentation associated with programmable aspects of system hardware and infrastructure (BS EN 62402:2007, 2007). In simple terms, software is a set of programs, supported by a manual, which will make the functionality of system or hardware simpler and easier to use.

Software plays an important role in Defence. Almost every project in Defence has got software with various degrees of complexity and dependencies. The range of software varies on the different platforms, from major complex weapon system to command and communication system, from mission planning to fighter aircraft. Software is an integral part of every project with various dependencies and complexities. This makes the management and maintenance of software through life extremely important especially the cost associated with the through life management. Software obsolescence is one of the key cost drivers which historically has been assessed as part of the maintenance cost due to the lack of data/information and unavailability of tool, techniques, and framework to assist with the software obsolescence management.

Software obsolescence happens when the original developer and authorised third-party cease to provide support with regular updates, upgrades, and fixes or due to changes in the target environment, systems, and hardware, which makes software unusable (Rajagopal, Erkoyuncu and Roy, 2015). This thesis is based on this definition of software obsolescence. MoD (UK) depends heavily on Commercial off the Shelf (COTS), the software support for which is typically terminated with the launch of new version of the software. As soon as this happens the software becomes obsolete.

Today, the military acquires, test, and fields system of systems mixing COTS and non-COTS products (Merola, 2006). These projects span for decades, and the support and maintenance of these products can exceed the vendor's capacity. Software becomes obsolete when there is a new technological advancement, the

software functionality being not required or other market factors. Vendor stop supporting when they release the new version of the same programme. Software becomes obsolete even when it is completely functional but not useful (Merola, 2006) i.e., the primary function of the software is not needed anymore.

Software gets obsolete generally due to one of the following three main causes (Sandborn, 2007)

1. **Functional Obsolescence:** If there are changes to the hardware, system, or other software in the same system. This will behave like a ripple effect as changes in one software or hardware will affect the next one and so on until it affects the entire system or capability.
2. **Technological Obsolescence:** This happens when vendor stops supporting the products or unavailability of the software in market etc.
3. **Logistical Obsolescence:** This happens when the media or the hard drive for example does not support the software. For example, some of the latest software may not function on old processors.

Other obsolescence such as skills and human obsolescence plays a major role in software obsolescence because it's human nature to move from one place to another or in this case from one project to another or from one company to another. It will be impossible for a vendor to accommodate these migrations. When the employees move, they take their knowledge of the product with them. Most of the safety critical aviation software is written in Ada 95 and there are not enough people that can maintain this software and vendors cannot afford to keep them either.

A recent study on software obsolescence came up with a different classification to P Sandborn's classification. This classification looked at skills and human obsolescence as shown in Figure 1-1

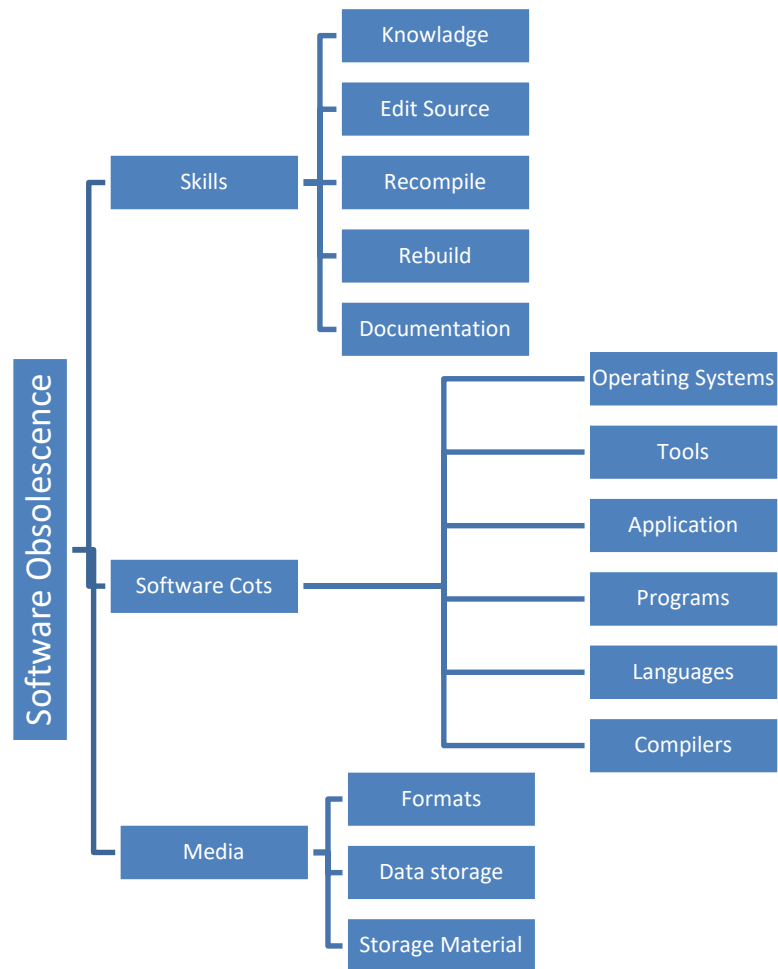


Figure 1-1Types of software obsolescence (Javier Romero Rojo, 2011)

The applicable definition of software obsolescence varies depending on the system that uses the software and where and how that system is being used. COTS software has got both end of sale dates and end of support dates that can be separated by a long period of time (P. Singh and P. Sandborn, 2013). Some researchers argue that software will not become obsolete as it is not affected by degradation and can be easily replicated. They are trying to apply the same reasoning to software obsolescence as mechanical or electrical component obsolescence. When an electronic or mechanical component becomes obsolete and there is no more stock available, the system cannot be maintained according to the original planning. Analogously, the software obsolescence prevents the software from being maintained accordingly (Javier Romero Rojo, 2011)

In the Defence industry it is common to describe the life cycle using the CADMID lifecycle, which is divided into Concept, Assessment, Demonstration, Manufacturing, In-service, and Disposal. Support contracts are generally covered with the in- service phase of the life cycle. These types of contracts are limited to the provision of transactional goods and services such as spares and repairs. As such any obsolescence be that software or hardware obsolescence is resolved reactively and the risk would lay directly on the customer, with payment against each occurrences needed. The cost incurred during these activities is much higher than the original procurement cost (Sandborn, 2007), however for Defence, the main objective is to obtain reliable operational capability at the lowest cost (Redling, 2004) .They undertake something called Value for Money Analysis (JSP 507) during the procurement of the capability however there is little evidence of doing the value for money analysis at the in-service phase. Therefore, the MoD has gradually moved on to a new availability-based contract (Johnsen, Howard and Miemczyk, 2009; P Stein and S Wady, 2008). The essence of this contract is that suppliers are paid for achieving the availability target of the capability in addition to software update/upgrade patch fixes etc. and not just for the spare parts. This transition is shifting the responsibility for managing and resolving both software and hardware obsolescence issue from the customer to the prime contractor who are in a better position to manage them in the most cost-effective manner (Josias, Terpenney J P and McLean K J, 2004; Webb and Bil, 2010). This new scenario has triggered a new need to estimate the NRE (Non-Recurring Engineering) cost of both software and hardware obsolescence at the bidding stage, so it can be included in the support contract. The NRE cost of hardware obsolescence is now well understood, however the same cannot be said of software obsolescence.

1.2 Research Aims

As described above, software obsolescence is very important in any real-time custom build software in the Defence industry. Customers are moving away and diverting the risk management including the obsolescence risk to the Original Equipment Manufacturer (OEM) or solution provider in the availability-based

contract to reduce the impact on the whole life cost. Although, there are numerous studies on hardware obsolescence and the importance of this is well understood, this is not the case with software obsolescence. Therefore, the study on software obsolescence is becoming more important, especially the need to estimate the cost, at very early stages of the project life cycle. In view of this research problem, the aim of this thesis is:

To develop a Software Obsolescence Cost Analysis Framework to predict the Software Obsolescence Resolution cost of custom build software at early stages of the product life cycle

1.3 Current Practice and Challenges

Most of the MoD projects are a combination of integrating old platform with new COTS or vice versa, hence the most challenging software obsolescence management problem is often found at these interfaces. In COTS world, hardware and software depends on each other heavily, software obsolescence is also driven by the changes in hardware improvement. There is a significant investment in both military and commercial world for improving the understanding of the obsolescence management process and issues.

From the case study as described in chapter 4, participant 1 has confirmed that they pass the obsolescence risk to the contractors but are unable to quantify the risk. The expert also confirmed that the Defence project team did not have the capacity or capability to understand and deal with software obsolescence. It was also identified in the case study that organisations have skills gaps to monitor, predict software quality and software obsolescence and therefore this issue of software obsolescence is passed on to the contractors.

Participant 3 advised that the software itself does not get obsolete but everything surrounding the software makes it obsolete. Whereas Participant 4 was of an opinion that software plays an important role in making a system obsolete. Participant 2 and 3 pointed out that the unstable requirement and changes in the Defence security policy plays an important role in getting software obsolete. All but participant 2 agreed that proactive approach to mitigating software

obsolescence is cheaper. Participant 2 advised that reactive approach to mitigate the software is cheaper in cash value but agrees that obsolescence does affect the hardware and therefore it becomes very expensive.

Participants from the case study confirmed that they were not aware of any obsolescence standards, and this was particularly clear, given the inconsistency in the definitions of software obsolescence, Participant 5 and 6 from the industry spoke about planned obsolescence where they talked about releasing a newer version of the software when the current version does not hold ground in the market.

Similar responses were received from different companies who attended the Component Obsolescence Group workshops in early 2015. To a direct question on how to manage software obsolescence, Plant Manager of Company A¹ responded that they do not have a software obsolescence strategy but have developed a strategy for system obsolescence. Team leader Material and Obsolescence of Company B² also commented that they understood that software obsolescence is a substantial issue.

The key issue as per the experts who attended the workshops in identifying the key cost drivers of software obsolescence were: -

- Unavailability of the data related to Software Obsolescence
- Inadequate design documentation and configuration
- Unavailability of historic cost data associated with Software Obsolescence
- Unavailability of Software obsolescence Management plan and strategies

To identify the key cost drivers, we need to clearly differentiate between activities under software maintenance such as updates, patches, and upgrades with software obsolescence. There is a clear relationship between software obsolescence with software design and this will be further explored in this research to identify the key cost drivers. The key challenges however as identified

¹ One of the Largest public transports companies in Europe

² One of the largest Defence Contractor in UK

by the experts from both semi structured interviews and workshops would be the availability of the data to identify potential cost drivers coupled with lack of information on potential resolution approaches and strategies.

1.4 Definitions

The thesis follows standard definitions for several important element throughout this research.

1.4.1 Bespoke and COTS (Commercial off the shelf) Software

According to IBM, bespoke software development is the process of designing, creating, deploying, and maintaining software for specific set of users, functions, or organisations. In contrast to commercial off-the-shelf software (COTS), custom software development aims at a narrowly defined set of requirements. COTS targets a broad set of requirements, allowing it to be packaged and commercially marketed and distributed.

Custom software, on the other hand, is designed for a specific set of needs, such as:

- a field service equipment maintenance program for a manufacturer or
- An online banking app designed for the unique requirements of the bank and its customers.

1.4.2 Type of software

These definitions of the type of software are adapted from the US DoD publication in 2009.

Real time Software

Real time is the most constrained type of software. These are specific solutions limited by system characteristics such as memory size, performance, or battery life. These projects take the most time and effort due to constraints. It is used in domains such as Microcode and Firmware, signal processing, vehicle control/vehicle payload, command and control and communications etc. Examples of real times include field programmable, gate arrays, flight control,

mission control, radar altimeter, network operations, and signal electronics, tracking sensors, encryptions, radio network and propulsion.

Engineering Software

Engineering software operates under less severe constraints than Real-Time Software. This software may take real-time software outputs and further process them to provide human consumable information or automated control of devices, or to perform transformation and aggregation/distribution of data. It is used in domain such as process control, scientific and simulations test, measurement, diagnostic and evaluations. Examples of engineering software includes operating systems, image processing, simulation and modelling, test equipment, file management, artificial intelligence, manufacturing process control etc.

Support Software

Support software assist with operator training and software testing. This type of software has few constraints. It is used in domain such as training software tools. Examples of support software includes computer-based training, compilers, programming aids, code generator, assemblers, courseware, test case generations, liners/loaders, code auditors etc.

AIS (Automated Information Systems)

AIS software provides information processing service to humans or software applications. These applications allow the designated authority to exercise control and have access to typical business/intelligence processes and other types of information access. These systems also include software that facilitates the interface and control among multiple COTS/GOTS software applications. It is used in domains such as mission planning, customs AIS software, enterprise service system and enterprise information system. Examples of AIS software includes scenario generators, target planning, enterprise services management, enterprise resource planning, transaction processing, data warehousing, financial transactions etc.

1.4.3 Software Maintenance

Software maintenance is defined in the IEEE Standard for Software Maintenance (IEEE 14764-2006 - ISO/IEC) as the modification of a software product after delivery to correct faults, to improve performance or other attributes, or to adapt the product to a modified environment

1.4.4 Software Obsolescence Mitigation Vs Software Obsolescence Resolution

Experts from the industry, following the case studies, interviews and online survey agreed that the software obsolescence mitigation is an activity that dictates whether a proactive or reactive approach to obsolescence issue to be taken. However, a software obsolescence resolution is the solution to the obsolescence issue regardless of a reactive or proactive approach.

1.4.5 Systems

(Davis G B, 1997); suggested that systems can be abstract or physical. An abstract system is an orderly arrangement of interdependent ideas or constructs. A physical system is a set of elements that operate together to accomplish an objective. Examples of physical systems are the circulatory system of a body, a school and a computer. A system consists of element that can be identified as belonging together because of a common purpose, goal or objectives.

Based on the above statement the definition of systems used in this research is as follows. "System" in this research consists of a hardware and its associated software, such as command and control systems, radar systems and communication systems etc.

1.5 Scope and out of scope of this research

Software in general takes different types and forms with several applications. Software dominates both commercial and defence world with increasing dependency. As such this research is focused on real time software which are bespoke, or custom build for defence and aerospace.

As such it is important to de-scope some of the elements for this research. De-scoping was undertaken due to the fact that there are several robust research which have already been undertaken, and or due to substantial lack of historic or current data. Following activities are out of scope for this research.

1. Software and hardware integration cost- this is extensively covered and researched in academic and commercial world. There are several tools such as Price TP and SEER cost estimating tools.
2. Software Quality, Reliability and maintenance –related cost drivers – this is again extensively covered and researched, and several commercial tools has got the capability to estimate this cost such as Price TP and SEER.
3. Uncertainty assessment and analysis in the cost model – The framework developed as part of this research does not undertake an uncertainty analysis due to unavailability of any historic or current data on software obsolescence. However, provision has been made in the cost model if the data becomes available in the future.

1.6 Research Objectives

The key research objectives are listed below.

1. To develop a list of sources of software obsolescence for a bespoke defence real time software.
2. To develop key cost drivers and cost elements for software obsolescence resolution.
3. To develop a framework to support the management, identification, and resolution of software obsolescence.
4. To develop a cost model to estimate the cost of software obsolescence resolution for a bespoke real time software.

1.7 Contribution to knowledge

This research has contributed to the understanding of software obsolescence and the factors that affects them. This research introduced a very novel concept of identifying and managing software obsolescence by developing a novel

framework for the management of software obsolescence in addition to developing a model to calculate the cost of software obsolescence resolution cost. The resolution is determined by the selection of resolution approach identified using the framework. This framework can be used at any stage of the development life cycle, be that be at the early bidding stage or at the in-service phase. The key contribution of this research is identified below.

1. This research identified for the very first time the root cause of the Software Obsolescence by deploying Cognitive case study methodology.
2. This research for the very first time developed a novel way to develop a cost estimating process to estimate the cost of software obsolescence resolutions.
3. This research for the very first time identified the key cost drivers and cost elements that are directly linked to software obsolescence resolution.
4. This research developed a novel way to identify, manage and mitigate software obsolescence by the development of Software Obsolescence Cost Analysis Framework.

The Figure 1-2 below shows how each key objective lead to the contribution to knowledge as a result of this research.

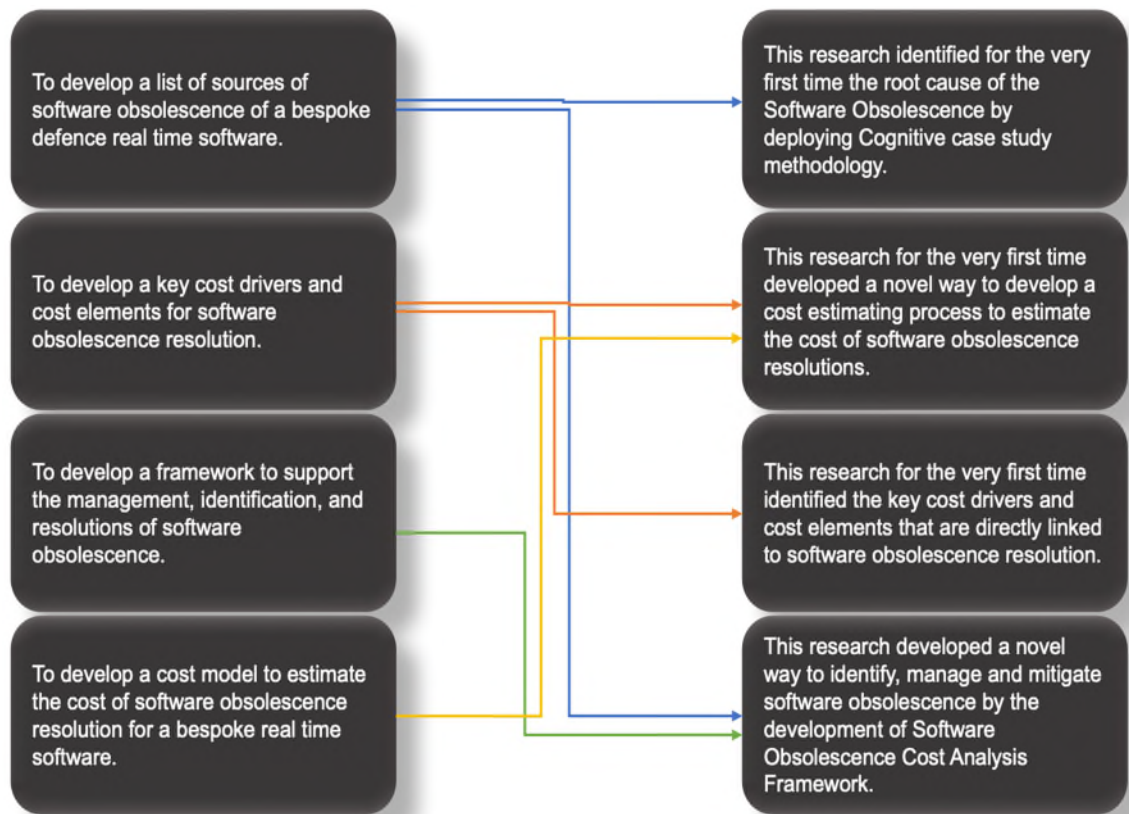


Figure 1-2 Mapping of Objectives to contribution to knowledge

1.8 The Collaborating Organisation

The main organisation that participated in this research project are QinetiQ UK, DE&S – UK Ministry of Defence, CAAS- UK Ministry of Defence, BAE Systems, Component Obsolescence Group (COG) now known as International Institute of Obsolescence Management (IIOM) and Society for Cost Analysis and Forecasting (SCAF). Of these, three of them helped the researcher to understand the current and overall understanding of Software Obsolescence in Defence and aerospace and the three remaining organisations participated in the iterative development of the Software Obsolescence Framework with six project teams from one of the organisations plus one other organisation helped with the validation of the framework.

QinetiQ UK

QinetiQ is a company of scientists and engineers committed to listening, understanding, and responding to customers' needs. QinetiQ is a British

multinational Defence technology company with headquarter in Farnborough, Hampshire. It is one the world's largest Defence contractor measured by 2011 Defence revenues and the sixth largest based in the UK. As a private entity, QinetiQ was created in April 2001; prior to this, it had been part of Defence Evaluation and Research Agency (DERA). A large portion of DERA's assets, sites and employees were transferred to QinetiQ with the other element were incorporated into Defence Science and Technology Laboratory (DSTL) which still remains in government. Former DERA locations, including Farnborough, MoD Boscombe Down, and Malvern have thus become key sites for QinetiQ. In 2006, QinetiQ was floated in the London Stock Exchange. In 2003, QinetiQ signed a 25-year Long-Term Partnering Agreement (LTPA) under which QinetiQ provides UK MoD with innovative and realistic test and evaluation of military and civil platforms, systems, weapons, and component on land, at sea and in the air.

QinetiQ employs more than 6000 people in more than 85 locations worldwide with over £1bn turnover. QinetiQ has registered over 1,300 patents including 300 pending one. QinetiQ's main business is divided into Cyber and Information, Air and Space and Maritime, Land and Weapons.

UK Ministry of Defence (MoD) – DE&S (Defence Equipment and Support)

DE&S is a bespoke trading entity, and arm's length body of the Ministry of Defence. DE&S manage a vast range of complex projects to buy and support all the equipment and services that the Royal Navy, British Army and Royal Air Force need to operate effectively. DE&S works closely with industry, including through partnering agreements and private finance initiatives. DE&S employs approximately 12,000 civil servants and military personnel around the UK and overseas with their headquarters in Bristol with an annual budget of £14 billion. DE&S is responsible for the procurement and support of ships, submarines, aircrafts, vehicles, weapons and supporting services; general requirement including food, clothing, medical supplies, and temporary accommodation; inventory management; British Forces Post Office and Submarine Dismantling Project. The key priorities of DE&S are to provide the right equipment to the armed forces and supporting it, whilst delivering better value for money to the

taxpayer and transforming the organisation so that it can better support the armed forces.

UK Ministry of Defence (MoD) – CAAS (Cost Assurance and Analysis Services)

Cost Assurance and Analysis Services (CAAS) has a proud heritage which dates back over 100 years to World War One when it was formed to prevent the munitions industry from overcharging the Government. The complexities of equipment procurement forced the government at the time to recruit its own technical cost engineers into the Civil Service, to advise commercial colleagues what non-competed contract prices should be paid to industry. This forms the heart of the purpose of CAAS.

Today, the CAAS team of over 300 cost engineers, forecasters, portfolio analysts and accountants provide expert independent costing advice to enable the Ministry of Defence to make effective business decisions on equipment procurement and support. CAAS provides dedicated teams to independently support customers across the MOD, including Defence Equipment & Support, Defence Infrastructure Organisation, Head Office, and the Front-Line Commands.

CAAS is geographically dispersed around the UK with offices close to or within sites of key Defence Industrial Suppliers

There are three key services provided by CAAS:

- 1. Financial Programming and Performance** by providing independent cost estimates through life for Equipment Procurement Plan and for Equipment Support Plan projects which gives a holistic view of equipment costs through life. This supports project teams and their stakeholders in setting realistic and robust budgets. CAAS also provides Assurance of Defence Costings to support the National Audit Office, including submission of Independent Cost Estimates into the Quarterly Review of Programme Costs process and aggregate project and programme data,

including historic performance data to provide a portfolio level view, CAAS also identifies and quantifies portfolio or sector level risks to inform MOD financial programming

2. **Investment Decision Support** by complete cost assurance and project evaluation work to give project stakeholders confidence that key investment decisions are based on sound cost and time foundations. CAAS are mandated for all DE&S project investment decision points for all new and contentious projects and all major projects of significant value.
3. **Contract Management Support** by supporting commercial staff with the management of equipment and support contracts. Cost engineers visit UK and overseas contractors to scrutinise their cost proposals, with cost estimating, modelling and risk analysis helping to assess evidence for negotiating taut fixed prices at contract placement.

BAE System plc

BAE Systems is a global defence, security, and aerospace company with approximately 100,000 employees worldwide. The company delivers a full range of products and services for air, land, and naval forces, as well as advanced electronics, security, information technology solutions and customer support services. In 2009, BAE Systems reported sales of £22.4 billion (US\$ 36.2 billion). BAE Systems is regarded as the second largest global Defence Company based on 2009 revenues (according to the Defence News Annual Ranking, published June 2010). Its headquarters is in Farnborough, Hampshire, England. BAE Systems was formed on 30 November 1999 by the £7.7 billion merger of two British companies, Marconi Electronic Systems (MES), the Defence electronics and naval shipbuilding subsidiary of the General Electric Company plc (GEC), and aircraft, munitions, and naval systems manufacturer British Aerospace (BAE). BAE Systems is involved in several major Defence projects, including the F-35 Lightning II, the Eurofighter Typhoon, the Astute class submarines, and the Queen Elizabeth class aircraft carriers.

Component Obsolescence Group (COG)/ International Institute of Obsolescence Management (IIOM)

The Component Obsolescence Group (COG) now called International Institute of Obsolescence Management (IIOM) is a non-profit making special interest group of like-minded professionals, from all levels of the supply chain and across all industries and relevant government agencies, concerned with addressing and mitigating the effects of obsolescence. COG was founded in the UK in 1997 and formed as IIOM in 2015, but since that time several overseas companies in the USA, mainland Europe and elsewhere have joined. COG Membership has grown to over 200 companies worldwide and holds quarterly meetings throughout the UK, aiming to:

- Provide education and awareness of the factors which affect obsolescence.
- Identify and develop processes for addressing or mitigating the effects of obsolescence.
- Communicate and co-operate with other national and international organisations with similar goals.
- Stimulate discussion and action between members for the members benefit and to communicate this on to the wider world.

Society for Cost Analysis and Forecasting (SCAF)

SCAF was formed in September 1984 following recognition of the need for a forum where those with interests and experience in cost analysis and forecasting could meet to discuss problems and techniques. SCAF was the initiative of the UK members of the International Society of Parametric Analysis (ISPA) with the support of specialist cost groups of the Ministry of Defence (MoD) and the Royal Aeronautical Society (RaeS).

“SCAF in association with industry partners has achieved significant improvement in the fields of parametric analysis and cost forecasting. Our seminars have delivered varied valuable contributions to the development of

forecasting techniques in the areas of bid assessment, through life costing, benefit analysis and reduction in expenditure of economic and financial appraisals". This has led to more rigorous and credible cost analysis supporting through life capability management of major Defence programmes. SCAF has tended towards Defence orientated industries, however, it is the intention to welcome and cover a broad a field as possible.

The objectives of SCAF (Society for Cost Analysis and Forecasting) are to promote the best interests of its members and to improve and disseminate knowledge of cost analysis and forecasting methods.

1.9 Thesis Structure and Summary

This section presents the structure of this thesis, which is illustrated in Figure 1-3. It outlines the activities that led to achieve the research aim.

In Chapter 2, a structured account of existing literature is critically analysed. The two key areas covered in this literature review are obsolescence and cost estimation. The objective is to provide a better understanding about the state of the art in these areas and identify any existing research gap.

Chapter 3 presents the objectives of this research, which have been deduced from the critical analysis of the existing literature. The research methodology developed to achieve these objectives is also presented in this Chapter. A thorough analysis of the possible approaches and strategies to design this research was carried out, and the justification of the methodology selected is based on it.

Chapter 4 presents the current practices in the software obsolescence management and Cost Estimation in the Defence and aerospace at the bidding stage. This chapter investigates the case studies, online survey reports and SME interviews that the researcher has undertaken.

Chapter 5 presents the cognitive case study that was undertaken by the researcher to determine the reasons for software developers to select the

technology that may become obsolete in the future. This chapter also investigates developing a model to identify the root cause of software obsolescence at the development stages of the software.

In Chapter 6 the researcher will discuss the Software Obsolescence Cost Analysis Framework that was developed based on case studies, online survey, SME interviews and Cognitive Case Study. This chapter will cover the key part of the framework such as Software Obsolescence Management level, Software Obsolescence Resolution approach, Software Obsolescence Complexity level and key cost drivers.

In Chapter 7 the research goes through the Software Obsolescence Cost Analysis optimisation tool that takes the inputs from the Cost Analysis Framework and estimates the cost of Software Obsolescence Resolution cost.

Chapter 8 describes the verification and validation of the SOCAF (Software Obsolescence Cost Analysis Framework) and SOCAM (Software Obsolescence Cost Analysis Model). The tool has been implemented in excel and subsequently validated by means of 5 case studies based on current projects in QinetiQ.

And finally, in Chapter 9 the work of this thesis is synthesised, and the implications of the research findings are discussed. The main research contributions are stated, along with the limitations and the future research directions. And finally, the overall conclusions are presented, demonstrating how the aim and the objectives have been achieved.

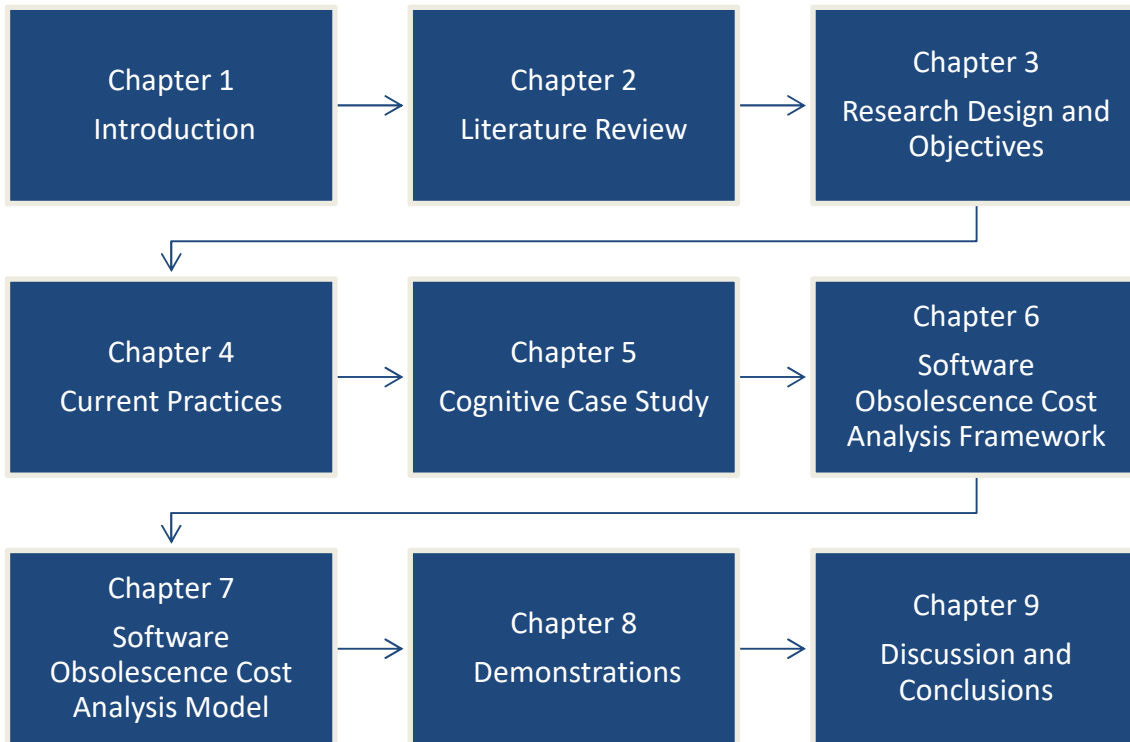


Figure 1-3 Structure of this thesis

1.10 Summary of Observation

Software Obsolescence is one of the key issues in the Defence industry; every system and every manufacturing technique has software embedded in them. It is becoming very important for the industry and Defence sectors to address software obsolescence and find an approachable way to counter the effect of the software obsolescence. It has become important to identify the software obsolescence at a very early stage to effectively manage this issue especially when the project life cycle is being extended to 25- 30 years. Therefore, to find the cost of these issues at a very early stage to inform the funding decisions is of key importance.

Software plays an important role in Defence Manufacturing. Almost every project in Defence manufacturing sector has got software with various degrees of complexity and dependencies. Whilst various research and studies have been conducted on system obsolescence, and tools developed to cost systems and components obsolescence, no major research has been undertaken to develop a framework to estimate cost of software obsolescence. Software becomes

obsolete even when it is completely functional but not useful. This creates a huge cost burden if not checked and controlled proactively.

This research will help in the identification and management of potential software obsolescence in Defence and aerospace and provide recommendation of potential software obsolescence resolution approach and predict the cost of the resolution. This will be undertaken by an extensive literature review to identify potential research gaps that needs to be addressed.

Once the literature review is undertaken it would be necessary to identify how the industry deals with software obsolescence and this will be done by means of case studies on real projects from the industry. This will be complimented by data from structured and semi structured interviews with the industry and academia. A further data collection activity by the means of online survey will be undertaken to compliment the findings from expert interviews and case studies.

A cognitive case study analysis will be undertaken to identify potential root causes of software obsolescence. This is to understand whether a software obsolescence can be identified much early in the software development stage.

All the above data collection method will help in the development of Software Obsolescence Cost Analysis Framework, consisting of several processes and tools to predict the cost of software obsolescence resolution and to manage software obsolescence reactively.

2 Literature Review

2.1 Introduction

This Section provides a literature review to the two areas covered in this research namely software obsolescence and cost estimation. The objective is to provide a better understanding of these areas of the research and to identify any research gaps.

2.2 Literature Review Strategy

A comprehensive investigation has been carried out to identify any publications related to software obsolescence. For this purpose, the main keywords used were “Software Obsolescence”, and “Obsolete Software”. The results were refined using keywords such as ‘Software Maintenance’, ‘Software obsolescence and design’, ‘Software reliability’, ‘Legacy Software’, ‘Obsolescence management’, ‘Hardware obsolescence management’, ‘Functional Obsolescence’, ‘Technological obsolescence’, ‘Cost of software obsolescence’, ‘Software Obsolescence Strategy’, ‘Hardware Obsolescence Cost’ and ‘Software obsolescence management’. Several databases were explored including Scopus, ABI Inform Complete (ProQuest), Web of Science (WoS), Business Source Complete, Electronic Journal Complete (EBSCO), Science Direct and IEEE Explore, in addition to these, literature from organisations such as COG (Component Obsolescence Group), DTech (Director of Technology), Galorath, Price Systems. Google and Google Scholar were also explored.

The title and abstract of papers downloaded were manually explored and analysed to ensure that they are suitable for this research. This literature review concludes that the research on the software obsolescence had commenced during the last 30 years, although the early research was on the software reuse (Standish, 1984), software quality (Born, 1986.) and software reliability (Bev Littlewood, 1987.; Hecht and Hecht, 1986). Later years the research concentrated on obsolescence management (Chou, 2013) and mitigating software obsolescence by service-based model (Romina Torres, 2007). Figure

2-1 below shows the number of publications on software obsolescence that was reviewed.

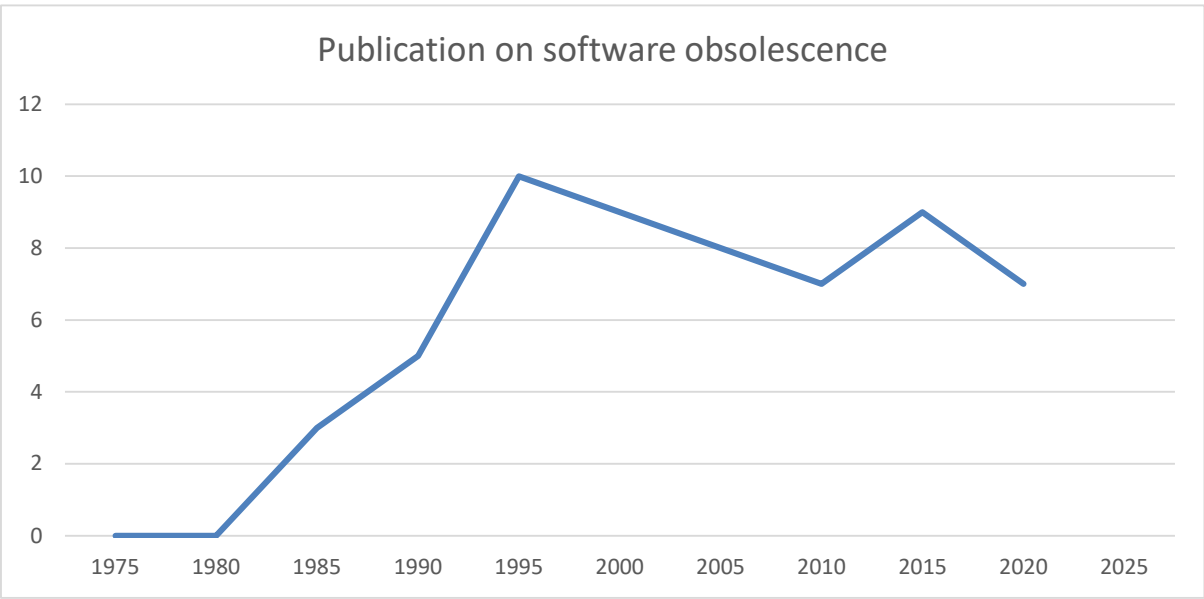


Figure 2-1 Number of publications (both Journals and conference papers) on software obsolescence that was reviewed

All the relevant papers were read and analysed further, which allowed the early identification of trends and key areas that were covered. Those areas were namely “Obsolescence management”, “Obsolescence Costing”, “Software Legacy”, “Software Reliability” and “Software Maintenance” and represented the current research scope within Software Obsolescence.

Table 2-1 below shows the classification of literature that was reviewed. It is worth noting that of the ones reviewed, there were several papers on how to increase the software reliability and maintainability to reduce the effect of obsolescence. This classification also shows that there is no clear trend in a particular area within the software obsolescence topic in recent years. This is explained further in this chapter.

Table 2-1 Classification of literatures reviewed

Year	Authors	Management Obsolescence	Software Obsolescence	Obsolescence Costing	Software Legacy	SW/Sys Reliability	Software Maintenance
2001	Marion, R						
2004	Readling, T						
2004	Neal, T						
2004	Dowling, T						
2006	Sandborn, P and Plunkett, G						
2006	Behbahani, A						
2006	SNEED, H.M.,						
2006	MEROLA, L.,						
2007	Sandborn, P. et al						
2007	Sandborn, P						
2007	Herald, T. et al						
2007	RUMNEY, G.,						
2010	Quyoum et al						
2012	SHAW, H.C., et al						
2012	ROMERO ROJO, F.J., et al						
2012	ROJO, F.R.,						
2012	MENG, X., et al						

2013	TORRES, R.,					
2013	TORRADO, N., et al					
2013	SANDBORN, P.,					
2014	JENAB, K., et al					
2015	Munoz, R.G., et al					
2016	Schmid, E., et al					
2016	Giovannoni, B.J., Boyles, C.					
2017	Li et al					
2018	Gerasimou, S., et al					
2018	Heagney, C.P., Walker, L.J.					
2018	Bowlds, T. T., et al					
2019	Shuai, Y., et al					
2020	Vergara et al					
2020	Ani Zuhra et al					
2021	Zaabr et al					
2022	Bin Wu et al					
2022	Ozkan et al					
2022	Kim S et al					
2022	Arturo E R et al					

2.3 Software Obsolescence

The US Obsolescence management group Diminishing Manufacturing Sources and Material Shortages (DMSMS) underpins a risk-based approach to determine its management priorities for software obsolescence (Jay Mandelbaum, 2012). According to DMSMS the ways to determine software obsolescence management is by approaching the software obsolescence and managing

obsolescence as heavily resource bound. A further study is required as this risk-based approach is difficult to quantify and the probability and the level of risk occurring is difficult to determine at a very early stage.

In the world of emerging dependencies on software, software development companies must cope with the increase in demands of requirement. This is even bigger within the Defence industry given the projects here tend to span for a longer period which makes the requirement unstable. This rapidly changing business environment is challenging traditional requirement engineering (Wnuk, Gorschek and Zahda, 2013). This makes the original requirement obsolete and there by increasing the project time scale, cost or even making the software obsolete prior to in-service. Having a requirement obsolete before the project is delivered, is a factor in identifying the cost of software obsolescence.

Software quality is measured from the foundation laid out by software requirement (Pressman R S and D, 1992). As mentioned earlier, software requirement changes more often and fast that the requirement tend to become obsolete. If this is the case, then the software quality is affected given this is based on the requirement. Obsolescence and market growth rates drive the decision to issue an upgrade, the timing of upgrades being one of the most difficult decisions (Mehra, Seidmann and Mojumder, 2014). A study conducted by Hall and Wilson, 1996; Shows that the experience and views of grassroots practitioner are not listened to and there is a significant bureaucracy in controlling the software quality. There have been several studies undertaken to improve software quality since then. This shows a clear indication about the direct relationship between the requirement and quality which will influence the software's life making it prone to obsolescence.

One way to tackle software obsolescence is to undertake a long-term digital preservation (Doyle, Viktor and Paquet, 2009). What software uses alongside with the system is the data and data is key for the survival of any business. The preservation of data is important for the success of any software business if the software becomes obsolete. The preservation of data can become a very expensive project if not proactively managed.

Managed integration uses a service-based approach to enable inclusion of wide-ranging languages, drivers, interface, protocols, test executives and instrument types into hybrid systems that easily combines new and existing elements (Wubben, 2007.). This is one way of dealing with system and software obsolescence in military test platforms. Weltzin and Schlonsky, 2010 talked about virtualisation of the Linux based test platform to counter obsolescence. This test equipment is used for a very long time and the costs associated with this ageing automated test equipment are exponential. \$50B was spent on DOD related ATE between 1980 and 1992, over the lifetime of this equipment cost are expected exceed \$250B (Weaver et al., 1992).

Reengineering is a process of reverse engineering, programme restructuring, software understanding and translating legacy software into modern programming language. One of the dominant issues in recent years is the need for reengineering the legacy software into modern language to combat the effect of software obsolescence. Ahrens, Prywes and Lock, 1995; came up with different scenarios of the reengineered software process. Whilst scenario gives us an understanding of what process needs to be applied to combat obsolescence, it does not explain the cost associated with this process. Even though this process as explained by the researcher is expensive however it does not quantify the cost.

British Standard 62402 (Obsolescence Management-Application guide, 2007) dictates to have a good software mitigation strategy and obsolescence management plan. JSP 886 vol 7 (JSP 886, 2013) dictates to have a good reactive and proactive strategy to mitigate software obsolescence. **Error! Reference source not found.** shows the difference between the proactive and reactive strategy.

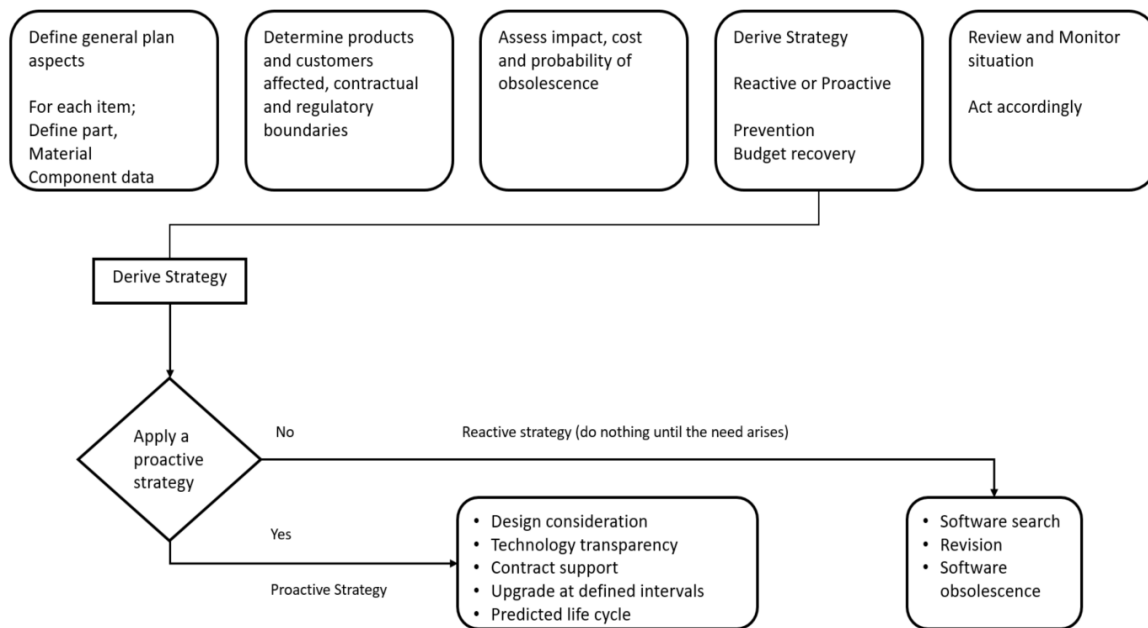


Figure 2-2 Reactive versus proactive strategy (adapted from BS62402)

There are lots of other factors that need to be considered in mitigating software obsolescence, one of the key factors being cost. There are different ways that the software obsolescence can be mitigated such as software license downgrade, redevelopment of software which is then required to requalify or re hosting of software, however, the key constraint to these remains Cost, Resource and Time. Component Obsolescence Group (COG) recommends using risk management-based process (Figure 2-3).

The importance of managing software obsolescence has become far more important in recent times, given the impact of software in nearly every element of systems to the extent that hardware and software drives obsolescence of one another. All the systems that have software content requires software support as a key element of the support and the total life cycle cost of such support will vary according to the system and how the software is procured. To understand and manage software obsolescence, it is important to understand what causes the software obsolescence.

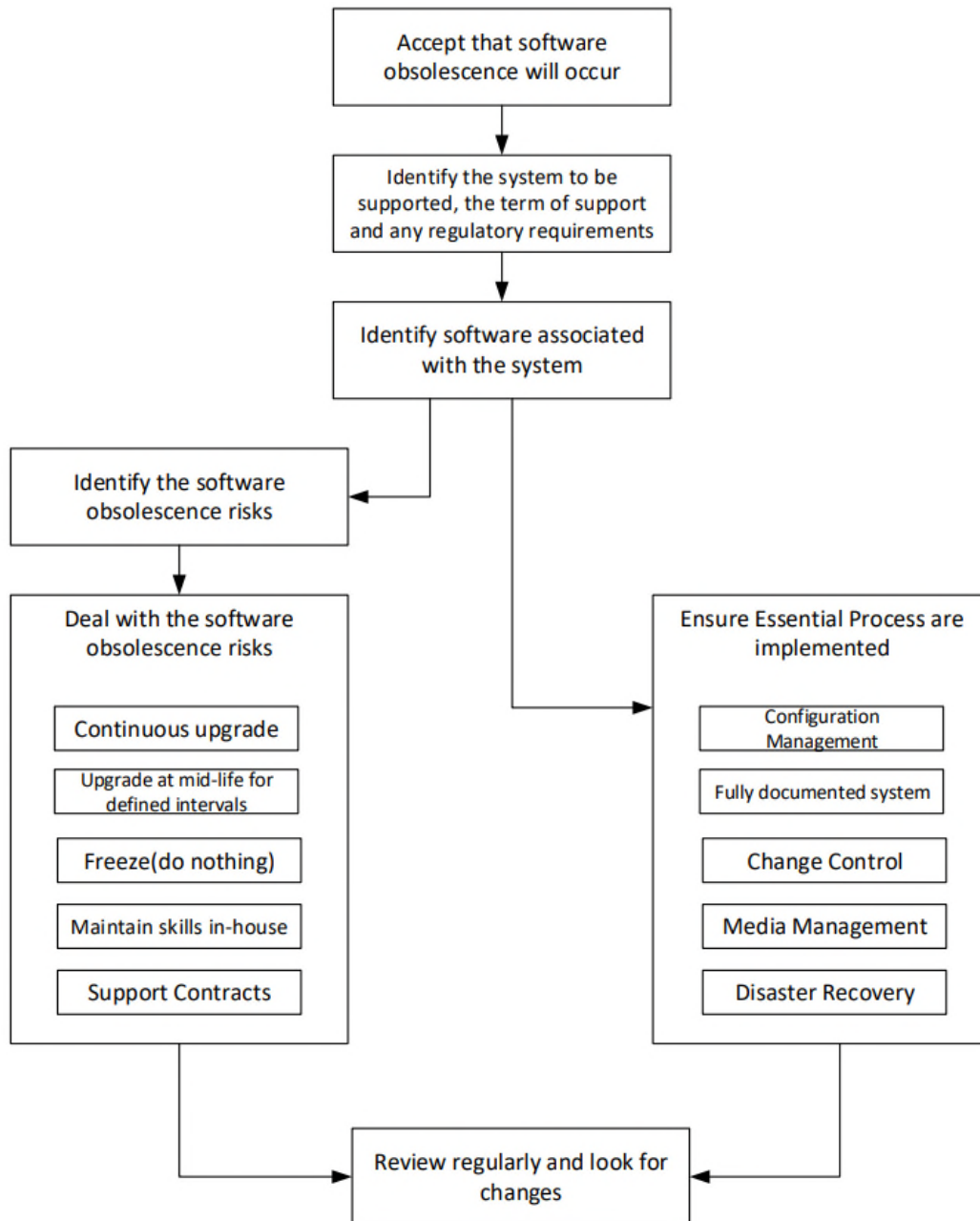


Figure 2-3 Software obsolescence process overview (COG, 2007)

The Defence sector has dealt with obsolescence with a reactive approach (Josias, Terpenney J P and McLean K J, 2004). Several authors (Condra, 1999; Flaherty, 2005; Francis, 2006; Howard, 2002; Josias, Terpenney J P and McLean K J, 2004; Leonard, Wolf and Stolinski, 1988; Marion, 2001; Meyer, Pretorius and Pretorius, 2004; Sandborn, 2007; Torresen and Lovland, 2007.) have advised that it is critical that there needs to be a change from reactive to proactive approach when dealing with obsolescence. The reactive approach is not ideal,

as finding solution with little or no advance warning is an extremely expensive business (Frank and Morgan, 2007; Josias, Terpenney J P and McLean K J, 2004; Trenchard, 2003)].

Yi Chen et al., 2012; suggested that the hardware product life cycle have three stages of failure rates, which is represented in the below figure (Figure 2-4).

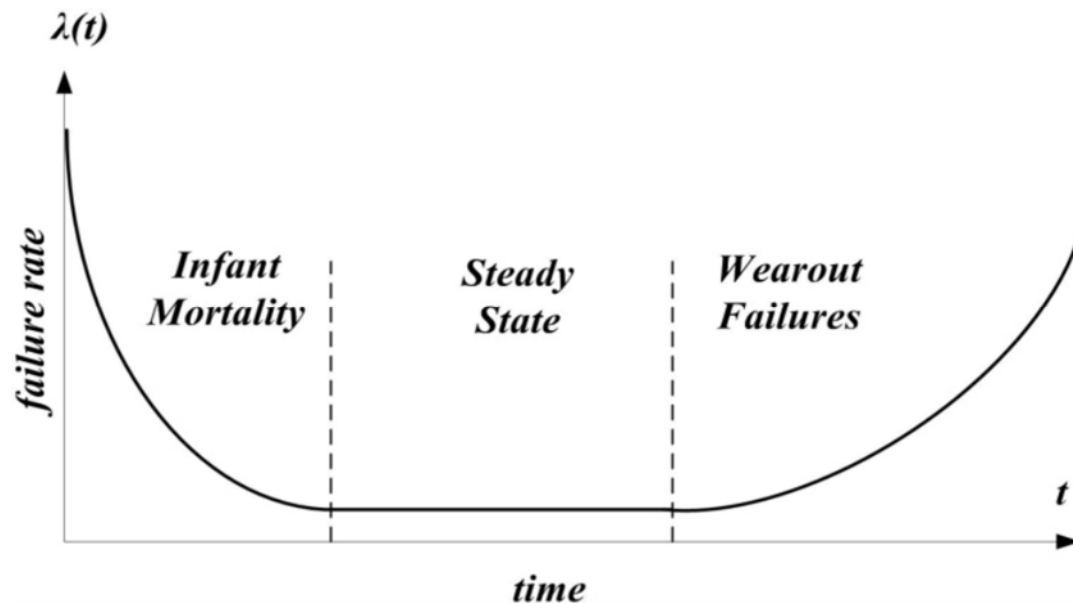


Figure 2-4 Hardware reliability curve

The wear out stage is when the component starts to get obsolete. This is true for almost all the component and hardware elements; however, software behaves differently. Once software is created or compiled on its own, the software tends to last long but there are various other factors that will make the software obsolete such as operating environment, media and skills (Singh and Sandborn, 2005).

Software reliability is one of the key attributes to the software quality. Software does not “wear-out” or “age”, as a mechanical or an electronic system does, the unreliability of software is primarily due to bugs or design faults in the software (Quyoun, Dar and Quadri, 2010). Quyoun, Dar and Quadri, 2010 also suggested that there is no good way of measuring software reliability. However, there have been several studies and research done on different software reliability model (Lyu, 2007; Torrado, Wiper and Lillo, 2013). Li and Pham, 2017, however came up with a model with a testing coverage for software reliability model that

considers not only the Imperfect Debugging (ID) but also the uncertainty of operating environments based on a Non-Homogeneous Poisson Process (NHPP). However, none of these models were able to identify or link the reliability to software obsolescence.

Discussion with the Defence obsolescence forum (Javier Romero Rojo, 2011) showed how the evolution of obsolescence levels differs from implementing a proactive versus reactive approach (Figure 2-5). Even though this graph represents the hardware and component obsolescence, this is very much true for software obsolescence.

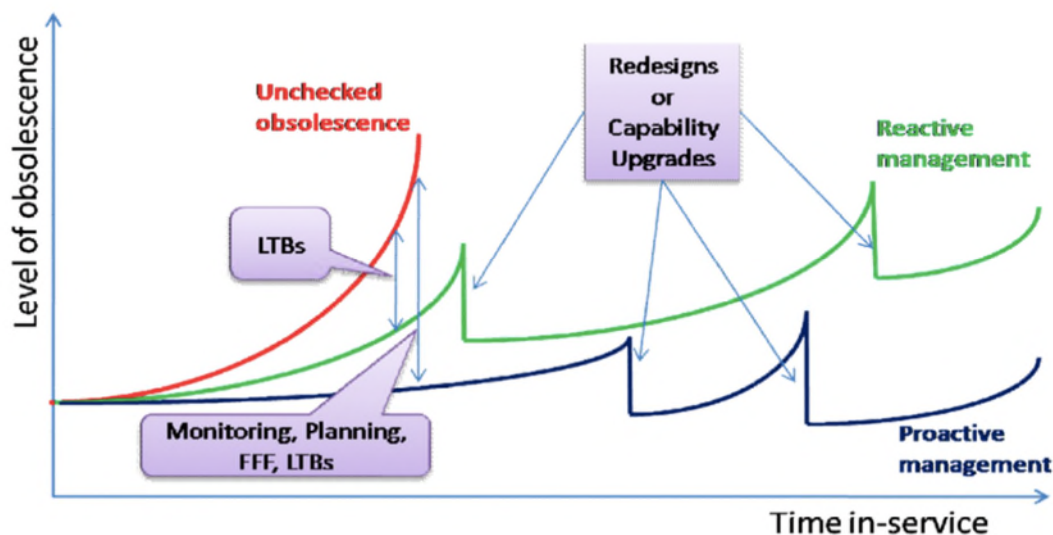


Figure 2-5 Hardware obsolescence and reliability graph (Javier Romero Rojo, 2007)

Project managers and engineers were comfortable to react to obsolescence issues as and when it happened by adopting quick fixes (Howard, 2002; Meyer, Pretorius and Pretorius, 2004). Several authors (Meyer, Pretorius and Pretorius, 2004; Singh, Sandborn and Lorenson, 2002; Singh and Sandborn, 2005) have advised that a proactive approach to deal with obsolescence will need to be used, and better obsolescence management will help in reducing the cost of obsolescence (Herald, Verma and Lechler, 2007)

The complexity of software development and engineering has brought its own problems, and this was recognised during the software crisis in 1968 for the first time at the NATO software engineering conference. Few of the problems faced by software projects are:

- Inability to estimate with accuracy of the Cost, resources and schedule required for a software project.
- Unreliable.
- Delivered late.
- Prohibitive in terms of modification costs.
- Impossible to maintain.
- Performing at an inadequate level.
- Exceeding budget cost.

According to NASA (NASA Cost Estimating Handbook, 2015):

- 55% of software projects exceed budgets by at least 90%.
- Software projects at large companies are not completed by 91%.
- Of the completed projects only 42% of them have original proposed features.
- Historical cost estimates for NASA projects are underestimated by a factor of at least 2.

Today, the military acquires, tests, and fields system of systems mixing COTS and non-COTS products (Merola, 2006). These projects span for decades, and the support and maintenance of these products exceed the vendor's capacity.

Software becomes obsolete when there is a new technological advancement, software functionality not required or other market factors. Vendors stop supporting when they release a new version of the same programme. Software becomes obsolete even when it is completely functional but not useful (Merola, 2006).

Software obsolescence is generally due to one of the three main causes as explained by (Sandborn, 2007).

- **Functional obsolescence:** If there are changes to the hardware, system, or other software within the same system. This will behave like a ripple effect as changes to one software or hardware will affect the next one and so on until it affects the entire system or the capability.
- **Technological Obsolescence:** This happens when vendor stops supporting the products or unavailability of the software in market etc.
- **Logistical Obsolescence:** This happens when the media or the hard drive for example does not support the software. For example, some of the latest software will not function within old processors.

Functional obsolescence is a secondary problem, generally caused by technological advancement elsewhere. Functional obsolescence leaves society with a piece of technology that has gone out of use, leaves the owner of the outgoing technology with reduced opportunity, and may provide another technology's owner with a new opportunity (Bradley and Dawson, 1998). Functional Obsolescence is usually due to the fact that capabilities of operational versions of a system (determined by the level of technical solutions available at creation) in the course of time will eventually cease to meet the needs of users who begin to focus on higher level of technical possibilities provided by new generations of objects of similar purpose (Lomakin and Murav'ev, 2016).

With rapid changes in technology and customer preferences, functional obsolescence of used products poses a serious challenge to resumed remanufacturing. Upgrade remanufacturing is a potential solution for dealing with the problems of functional obsolescence (Wu et al., 2022). Technological obsolescence affects any device that uses some kind of computer program since a software is constantly updated (Vergara et al., 2020). Technology obsolescence, as any kind of obsolescence is unpredictable and largely deals with hardware than software and has been concurred by Zaabar et al., 2021; who revealed that technological obsolescence is a major cause for the electronic waste problem. Kim et al., 2022 analysed and produced a model to predict the technology changes in smartphones.

It is good to have some classification of the causes of software obsolescence, but P Sandborn overlooked human's skill obsolescence factors. Human obsolescence plays a major role, given it's human nature to move from one place to another or in this case from one project to another or from one company to another. It will be impossible for a vendor to accommodate these migrations. When the employees move, they take their knowledge of the product with them. In Defence most of the safety critical aviation software is written in Ada 95, however, there is a lack of adequate skilled resources for the maintenance of these software, in addition to the fact that vendors are unable to afford these SQEP resources either.

A study by (Javier Romero Rojo, 2011) on software obsolescence came up with a different types of software obsolescence, which looked at skills and human obsolescence as shown in Figure 1-1(Javier Romero Rojo, 2011).

The applicable definition of software obsolescence varies depending on the system that use the software and where and how that system is being used. COTS software has got both end of sale dates and end of support dates that can be separated by a long period of time (Sandborn, Prabhakar and Ahmad, 2011).

Some authors argue that software will not become obsolete, as it is not affected by the degradation and can be easily replicated. They are trying to apply the same reasoning to software obsolescence as a mechanical or electrical component obsolescence. When an electronic or mechanical component becomes obsolete and there is no more stock available, the system cannot be maintained according to the original planning. Analogously, the software obsolescence prevents the software from being maintained accordingly (Javier Romero Rojo, 2011).

It is important to identify the root cause of software obsolescence to identify the mitigation actions. The root cause of software obsolescence can be traced to every aspect across the supply chain from support solutions to infrastructure (Sandborn, Prabhakar and Ahmad, 2011)

There is very limited research on the proactive management of software obsolescence, let alone costing of the software obsolescence at the bidding stage. To be proactive in managing software obsolescence, there is a need for planning for software obsolescence. The planning for reactive strategy is based on post obsolescence issue (Jenab et al., 2014), however the planning for obsolescence should start very early in the project life cycle. The planning for software obsolescence should also include the changes to the institutional policies and digital preservations (Formenton, 2020). Effective and efficient obsolescence planning should be able to detect and deal with obsolescence (Jenab et al., 2014), however there is no framework or process related to effectively manage software obsolescence.

Muñoz et al., (2015); investigated how best to monitor and plan for COTS software and presented a framework for aerospace industry. Ozkan and Bulkan, 2022 proposed a discrete- time simulation model by using Multi-objective Evolutionary Algorithms and developed a cost model for COTS. However, the methodology cannot be replicated for a custom-built Defence software, as the characteristics of COTS software is different to custom built software from a customer perspective. Bjoern Bartels et al., (2012), in his book talks about prediction, mitigation and management of obsolescence which can be related to software obsolescence and talks about various ways that software obsolescence can occur.

Author agrees with Shuai, Zheng and Huang, 2018 that majority of the research on software obsolescence is undertaken by the Defence industry to solve the existing issue with software obsolescence and largely focuses on reactive approach with reliance on expert knowledge.

There are no known software obsolescence monitoring tools available in the market, however, there have been several attempts to develop a process of monitoring the software obsolescence. Muñoz et al., (2017); developed a framework to identify the trigger point for software obsolescence in COTS software. Jenab et al., 2014 developed a “Dynamic Obsolescence Model” which investigates all the stages of the product development cycle.

The issue with the first one was that it deals with software obsolescence in COTS software while the latter one, looks at high-level information including upgrade, updates, and bug fixes etc. which is routine maintenance activities to reduce the obsolescence issue. However, software could go obsolete due to various factors both internal and external. Both the models lacked the detail that was needed to recommend the appropriate resolution approach that will have a huge cost implication. Another method proposed by Bowlds, Fossaceca and Iammartino, 2018; was the use of TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) in Multi criteria decision making. Bowlds, Fossaceca and Iammartino, 2018; suggested that multiple technical elements affect the subjectivity of when software becomes obsolete, as no single strategy focusing on a single element can provide a confident risk assessment. The risk of software obsolescence depends on the interactions between several components, rather than a single element.

Several risk-based models were introduced, such as a step-by-step approach for managing software obsolescence (Schmid et al., 2016) which is a risk-based approach. Erkoyuncu, Ononiwu and Roy, 2014 discussed mitigating software obsolescence in oil and gas and proposed a model using resources, materials, and tools. However, these techniques fail to talk about any mitigation action that needs be undertaken to reduce the future software obsolescence issue proactively.

Several other models to predict hardware and material obsolescence have been developed such as E-Force and M-Force (Javier Romero Rojo, 2011) but those models and key cost drivers cannot be mapped or replicated to solve software obsolescence issues or identify key cost drivers. It is understood that Technical and functional obsolescence play a big role leading to additional unplanned cost (Giovannoni and Boyles, 2016).

2.4 Cost Estimate

The second part of the literature review is about cost estimation in general and specifically around software cost estimation. The main cost estimating techniques that are used for Software Cost Estimating is discussed and analysed.

Estimating cost of software is subtly different to the hardware and component cost estimation Rajagopal S, 2020, however, the basic principle of estimating is the same and it follows the best practice as the cost estimating process identified by GAO (GAO, 2009), which is represented in 12 steps as shown in the Figure 2-6 below. There have been several modifications to this model based on each organizational requirement.

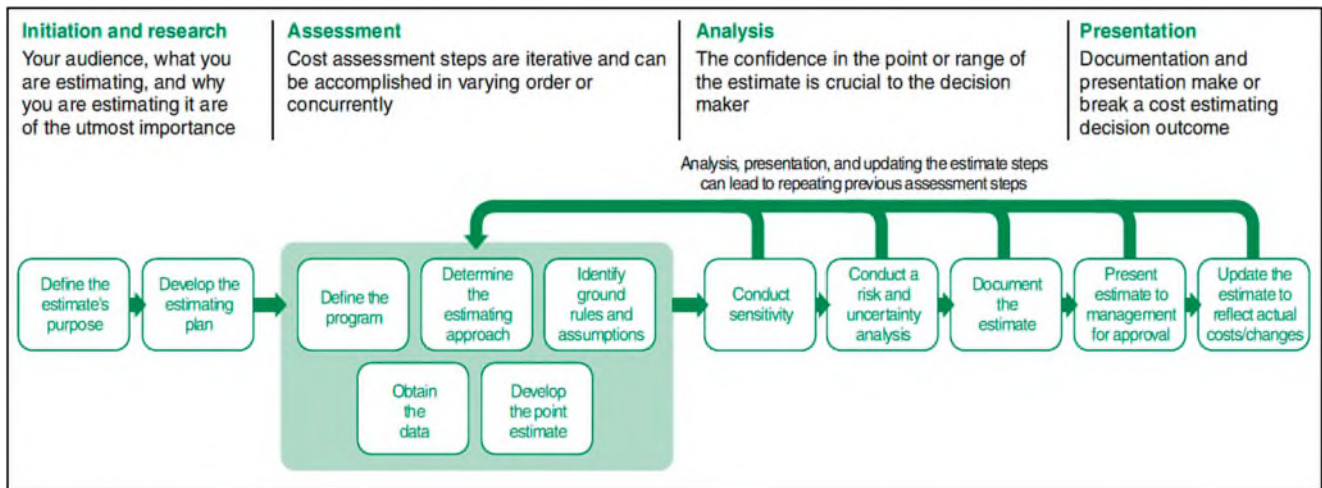


Figure 2-6 12 steps of estimating by (GAO, 2009)

Cost estimating is the determination of quantity and the predicting or forecasting, within the defined scope, of the cost required to construct and equip a facility, to manufacture goods, or to furnish a service (AACE, 2006).

Cost estimating can be defined as the process of collecting and analysing historical data and applying quantitative models, techniques, tools, and databases to predict the future cost of an item, product, program, or task (CEBoK, 2020, Robert F et al, 2019, Aini Zuhra et al, 2020, Arturo E R et al, 2022). Basically, cost estimating is the process of predicting the future based on today's knowledge to help facilitate the successful completion of a program, project, or process. Often the cost estimator will be put in the position of adjusting for new materials, new technology, a new software programming language, a different team of individuals, etc.

The purpose of cost estimating is to translate system/functional requirements associated with programs, projects, proposals, or processes into budget

requirements. It determines and communicates a realistic view of the likely cost outcome of a system or program. This outcome can form the basis of the plan for executing the work to develop, field, and support the system or program (Mostafa R et al, 2021 Aini Zuhra et al, 2020, Arturo E R et al, 2022).

Cost Estimating, essentially predict the future cost of resources, methods and management based on the historical data and experiences (Javier Romero Rojo, 2011). There are many challenges to develop a robust and defendable estimate. Few of them are listed below which are identified by (GAO, 2009; Kingsman and de Souza, 1997; Roy, 2003).

- Unavailability of historical data
- Data not reliable
- Inexperienced Cost Analyst
- Optimism bias
- Overlooking risk and uncertainty
- Unstable requirement etc.

According to the ICEAA's Cost Estimating Body of Knowledge (CEBoK, 2020), There are three primary reasons for estimating the cost of a program.

- **Budgeting:** Program budgets should be built using an Independent Cost Estimate (ICE), not only to give that program a chance for success, but also because valuable agency resources need to be used as effectively as possible. If a program runs over budget, resources will need to be pulled from other programs to cover costs.
- **Planning:** Once budgets are established, planning must occur to lay the foundation for successful project execution. Estimating supports the development of proposal basis of estimate (BoE) documentation, which in turn serves to help establish the integrated baseline. Earned Value Management (EVM) is an approach that tracks cost and performance against this baseline, as measured by performance indices.
- **Trade-offs:** Trade-off studies use a baseline cost estimate and explore the relative cost of changing requirements. The end goal is to achieve the proper balance of maximizing utility and minimizing cost.

Risk plays a big part in all three. Without risk, budgets are likely to be unrealistically low, resulting in immediate shortfalls; baselines are likely to be unrealistically aggressive, resulting in immediate performance deficiencies; and trade-offs are likely to be unrealistically optimistic, resulting in faulty decisions to pursue ultimately unaffordable options.

2.4.1 Cost Estimating Techniques

Cost estimating techniques are the building blocks of a cost estimate. They provide structure that are used to predict future costs based on historical data. Cost estimating techniques rely on statistical properties, logical relationships, and emotional appeal. There are several cost estimating techniques that are described in several literatures (Javier Romero Rojo, 2007; Mostafa et al,2021; Stewart & Shirvan, 2021; Christoph H et al, 2022) such as –

- Parametric
- Bottom Up
- Analogy
- Benchmarking etc.

However, Niazi et al., 2006 broadly classified the estimating technique into two which are more widely used. (Figure 2-7)

- **Qualitative Techniques**
 - o Intuitive – based on the Expert judgement
 - o Analogical – this is comparing the cost against the similar project/product which has been estimated in the past
- **Quantitative Techniques**
 - o Parametric – this is based on Cost Estimating Relationship (CER) which are created using set of dependent and independent variables developed from historical data
 - o Analytical- based on a detailed analysis of the work required into the elementary tasks that constitute the manufacturing process such as bottom-up costing

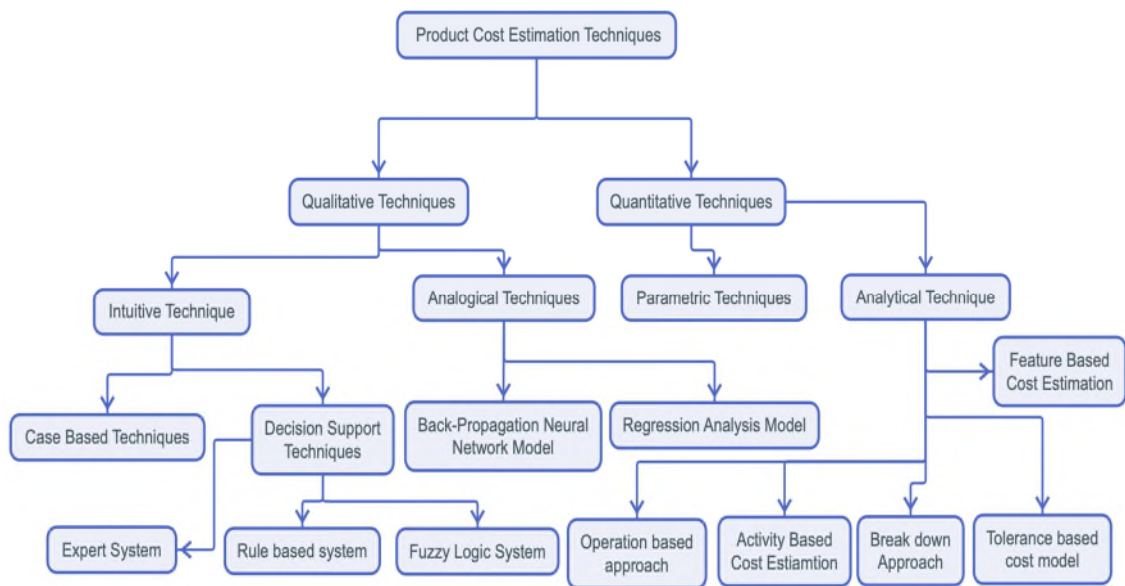


Figure 2-7 Classification of estimating techniques (Niazi et al., 2006)

Each technique can be used at various stages of the project life cycle as described in Figure 2-8. During early stages of the project there is limited availability of data and analogy based or parametric estimate are recommended which can be validated using an expert judgement, however, researcher have used parametric estimate throughout the project life cycle. As the project matures and more information is available a bottom- up estimate or extrapolation from actuals can be used. It is advised to use more than two different approaches to increase the confidence in the estimate (Niazi et al., 2006).

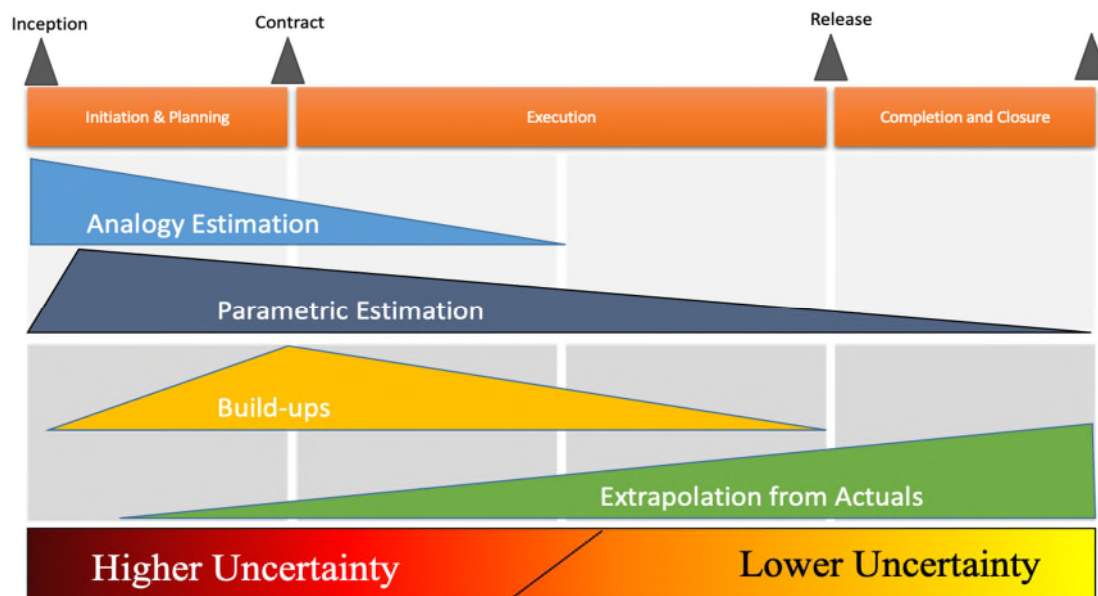


Figure 2-8 Estimating methodology at each stage of the product life cycle adapted from SCEBoK, 2020

The strength, weakness, and the application of each technique is summarized in the Table 2-2

Table 2-2 Strength, weakness, and application of estimating technique (GAO, 2009)

Methods	Strengths	Weakness	Application
Analogy (Extrapolation) (Near Neighbour)	- Requires little data - Based on actual data - Reasonable Quick - Good audit trail	- Subjective adjustments - Accuracy depends on similarity of items - Difficult to assess effects of design change - Blind to cost drivers	- When little data is available - Rough-order-of –magnitude estimates - Cross-check
Parametric	- Reasonably quick - Encourages discipline - Good audit trail - Objective, little bias - Cost driver visibility - Incorporates real-world effects	- Lacks detail - Model investment - Cultural barrier - Needs to understand model's behaviour	- Budgetary estimates - Design-to-cost trade studies - Cross-check - Baseline estimates - Cost goal allocations
Engineering Build-Up	- Easily Audited - Sensitive to labour rates - Tracks vendor quotes - Time honoured	- Requires detailed design - Slow and laborious - Cumbersome	- Production estimating - Software development - Negotiations

2.4.2 Software Cost Estimating

Software Cost Estimation follows the same principle as any other hardware/component Cost Estimation (Rajagopal, S; 2017, CAAS, MoD; 2018). However, since software does not follow the same attributes as hardware it's difficult to understand how each software perform, which makes it difficult to predict the cost (Boehm and Boehm, 1981). Some of the issues with software cost estimation are highlighted below (Boehm and Boehm, 1981; Singh, Singh and Mishra, 2018)

- Lack of data on completed project. It can be argued that this has improved drastically over the year. However, each software project is slightly different to other.
- Estimates are often compiled in a hurry, without an appreciation for the effort required to undertake a credible estimate.
- Unstable requirement – this is since there is technology insertion happening more often than the production of a solution.
- Lack of suitable qualified and experienced people to undertake a software cost estimate.

Software is defined as computer programs, procedures and possibly associated documentation and data pertaining to the operation of a computer system (IEEE 828-2012 STD).

There are some attributes of good software specified by (CEBoK, 2020; Junaid R et al, 2022; Boehm et al, 1984; Tribhuvan T et al, 2018) which are detailed below.

- Delivers required functionality with adequate performance
- Is maintainable, dependable, and acceptable
- Designed to evolve easily to meet changing needs
- Uses system resources efficiently
- Is understandable, usable, and compatible with other systems

The main difference between software and hardware is that there is no production phase in software. Software is typically fully designed, developed, and tested as part of a Research and Development (R&D) phase.

Programming a computer does require intelligence. Indeed, it requires so much intelligence that nobody really does it very well. Programming computer is by far the hardest intellectual task that human beings have ever tried to do (Weinberg, G. M; 1988). And this makes it difficult to estimate the cost even though the cost estimating principle is the same for both hardware and software.

A study undertaken by Standish Group in 1994 and which was revisited in 2000 concluded

- Schedule overruns have significantly decreased from 222% over the original time estimates in 1994 down to 63% in 2000
- Cost overruns have gone from 189% over the original cost estimates in 1994 down to 45% in 2000

This is since there has been evolution of better tools to monitor and control progress and there is a constant innovation in the management process.

Estimating the cost of software development follows the same principle as hardware estimating; and various techniques that were discussed earlier in this chapter can be used for software estimating. However, the key cost driver that effect the cost is different. Some of the key cost drivers that are identified in the literatures are listed below (Wilson R et al, 2022; Poonam R et al,2022; Junaid R et al, 2022; Boehm et al, 1984; Tribhuvan T et al, 2018).

- Software size
- Software Complexity
- Development Environment
- Product characteristics

2.4.3 Software Support Cost Estimate

The impact of software obsolescence issue is higher in the in service or the support phase of the product life cycle, however research has indicated that the

software will become obsolete even before the product is in service (SCEBoK, 2020; Wilson R et al, 2022; Poonam R et al, 2022; Junaid R et al, 2022; Boehm et al, 1984) due to various reasons, such as

- Requirement obsolescence
- Technology insertion
- Long design and development phase etc.

The research will focus on this phase of the software product life cycle due to the above reasons. There has been several research undertaken to estimate the cost of software support cost, obsolescence is one of the key cost drivers in the software maintenance cost estimating (SCEBoK, 2020) yet there is very limited research undertaken to estimate or predict the cost of resolution for software obsolescence. Obsolescence is dealt reactively, and this can be very expensive.

Software maintenance is defined as “the process of modifying a software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment” (IEEE, 1990). However, this definition suggest that the maintenance should start after the product delivery but the planning for software maintenance should start at the design phase. The researcher agrees with Thomas M. Pigoski, 2008; in his book “Practical Maintenance – Best Practice for managing your software investment” where it was noted that “software maintenance is the totality of activities required to provide cost-effective support to a software system. Activities are performed during the pre-delivery stage as well as the post-delivery operations, supportability, and logistics determination. Post-delivery activities include software modification, training and operating a help desk”

There are four types of software maintenance as defined by ISBSG

1. **Perfective Maintenance** – this is the modification of a software application, after delivery to improve performance or maintainability.
2. **Preventative Maintenance** – this is the modification of a software application after delivery to detect and correct latent faults in the software product before they become effective faults

3. **Corrective Maintenance** – the reactive modification of a software product performed after delivery to correct discovered problem.
4. **Adaptive Maintenance**- Enhancements necessary to accommodate changes in the environment in which a software must operate.

However, ISO/IEC 12207 identifies the primary activities of software maintenance as:

1. Process implementation
2. Problem and modification analysis
3. Modification implementation
4. Maintenance review/acceptance
5. Migration and
6. Software retirement

The idea of reuse in software industry is well accepted as a solution to cost savings. The importance of software maintenance is critical if the intention is to gain cost saving through reuse.

There has been several research on the best way to estimate the cost of software maintenance resulting in several models. The key models are highlighted in the Table 2-3 below. (Nguyen, 2010)

Table 2-3 Commercial tools and models to estimate the cost of software maintenance

Model	Maintenance Task Type	Effort estimated for	Modelling Approach	Key Input Metrics	Best Estimation accuracies reported
COCOMO	Regular maintenance, Major enhancement (Adaption and reuse)	Maintenance period, adaptation and reuse project	Linear and Non-linear regression Bayesian Analysis	SLOC added and modified, SU, UNFM, DM, CM,IM	

Knowledge Plan	Regular maintenance, Major enhancement	Maintenance period, adaptation and reuse project	Arithmetic	SLOC or IFPUG's FP of code added, reused, leveraged prototype, base, changed, deleted, system, base	
PRICE-S	Regular maintenance, Major enhancement	Maintenance period, adaptation and reuse project	Arithmetic	SLOC, IFPUG's FP, POP or UCCP of new, adapted, deleted, reused code	
SEER-SEM	Corrective, Adaptive perfective	Maintenance period, adaptation and reuse project	Arithmetic	Maintenance rigour, Annual change rate, Years of maintenance, Effective size (measured in SLOC or IFPUG's FP)	
SLIM	Corrective, Adaptive perfective	Maintenance period	Heuristic	SLOC added and modified	
Basili 1996	Error correction and enhancement	Release	Simple linear regression	SLOC(sum of SLOC added, modified, deleted)	
Basili 1997	Corrective	Component version	Classification (C4.5 algorithm)	Number of function calls, declaration statement, exceptions	76% classification correctness
Niessink and van Vliet 1998	Adaptive(Functional enhancement)	Individual change request	Linear regression	IFPUG's FPA, Maintenance Impact Ratio (MIR)	MMRE=47%, PRED(0.25)=53%
Abran 1995	Adaptive(Functional enhancement)	A set of activities resulting a maintenance work product (Release)	Arithmetic	Extended IFPUG's FPA	
Abran 2002	Adaptive(Functional enhancement)	Release (Project)	Linear and Nonlinear regression	COSMIC-FFP	MMRE =0.25PRED(0.25) =53%

Caivano 2001	Adaptive(renewal e.g. reverse engineering and restoration)	Iterations	Linear regression, Dynamic calibration	SLOC, McCabe's complexity, Halstead's complexity, Number of modules	
Sneed 2004	Error correction ,Routine change, Functional enhancement, technical renovation	Release	Productive index, COCOMO II	SLOC, IFPUG's FP, Object points(Sneed 1995), number of errors, Test coverage, Complexity index, Quality index, Productivity index	
Ramil 200,2003	All task types for evolving the software	Release	Linear regression	Number of systems, modules	MMRE= 19.3%, PRED(0.25)=44%, (Cross-validation)
Sneed 1995	Corrective, Adaptive perfective, Preventive	Individual maintenance task	Productivity index (multiplied by adjusted size)	SLOC, IFPUG's FP, Object points, Data-points, Complexity index, Quality index, Project influence factor	
Briand and Basili 1992a	Corrective, Adaptive (and enhanceive)	Individual change request	Classification	Modification type, source of error, SLOC added, modified, deleted components added, modified, deleted objects(code or design or both)	74%-93%, Classification correctness
Jorgensen 1995	Corrective, Adaptive perfective, Preventive	Individual maintenance task	Linear regression, Neural network, Pattern recognition	SLOC (sum of SLOC added, modified, deleted)	MMRE=100%, PRED(0.25)=26% (Cross-validation)
De Lucia 2005	Corrective	Monthly period effort	Linear regression	Number of tasks, SLOC	MMRE=32.25%, PRED(0.25)=49.32% (Cross-validation)

From the literature review carried out, it can be concluded that none of the current and existing software maintenance cost estimation models have got any attributes or considers the cost of software obsolescence resolution.

2.5 Key findings from the literature review and research gaps

It has been identified through this literature review that there has been very strong and extensive research on component and hardware obsolescence, however, there has been very limited research carried out in the field of Software Obsolescence and estimating the cost of the same. There is very little evidence in the literature of identifying, managing, mitigating, and estimating the cost of the resolution approach for software. This highlights a substantial lack of understanding of the very concept of software obsolescence.

Obsolescence is usually managed on a reactive basis, which often leads to redesigns or loss of capability. A more structured approach to test system architecture with software and hardware best practices can significantly reduce the pain of obsolescence (Robinson et al., 2015). Gerasimou et al., 2018 proposed a modernisation methodology that can help developers of client systems to maintain their source code when these obsolescence issues occur. This methodology applies to programming languages that support library linking which is one way to support one type of software obsolescence issue. Hardware obsolescence is relatively well-understood with sufficient replacement parts and proactive strategy. Software, however, presents a set of challenges that are harder to mitigate, as the detrimental effects of long-term disconnection, software data corruption, and malware are numerous causing potentially devastating obsolescence issues which needs to be managed and mitigated (Jang et al., 2017).

Bartels et al., 2012 in his book *Strategies to the Prediction, Mitigation and Management* explored various types of obsolescence and talked about how to manage, mitigate, and predict obsolescence. However, predicting software obsolescence is something that was not researched in detail. Bartels et al., 2012; did come up with the root cause analysis for software obsolescence, which is further explored and adapted to develop software obsolescence resolution approach further in this thesis.

The literature review provided evidence and understanding about status quo in the field of software obsolescence and the cost estimation for resolution of

software obsolescence. To establish what is required to understand this issue, it was necessary to gain understanding of the current practices in the Defence and aerospace sector by means of case studies, survey etc.

The literature review also suggested that the software obsolescence does not necessarily happen in the in-service or support phase of the project life cycle but could happen any time after the project inception. This needed to be explored further by undertaking a cognitive case study. Majority of the literatures that were reviewed looked at reactive management of software obsolescence by recommending update, upgrade and bug/patch fixing which is a regular day to day maintenance activity to reduce the effect of software obsolescence.

Summary of the research gaps identified are listed below.

- There is a lack of research that highlight the issue of software obsolescence at very early stages of the project life cycle.
- Lack of commonly accepted approaches to mitigate software obsolescence cost effectively.
- Lack of research in identifying cost of software obsolescence resolutions
- Research in software supply chain is limited, further research is required in identification of the integration issues which arise from the complexities of the supply chain.
- Majority of the research is concentrated in the in-service phase of the project. (This highlights that a reactive approach to mitigate software obsolescence is commonly preferred).
- Within industry, software programs such as PriceTrue planning, SEER and OSCAM have highlighted methods to estimate operation and support cost of the software using various matrices and CER's, however, there are no applications to determine the cost of software obsolescence resolutions.

- Bulk of the research is carried out in the in-service phase of the project life cycle.
- It is likely that the bulk of the research is undertaken to solve the software obsolescence reactively.
- Limited or no research is undertaken to proactively predict and resolve software obsolescence issues.

The next chapter in this research describes the objective, aim and development of the research methodology undertaken to develop the software obsolescence cost analysis framework.

3 Research Methodology

3.1 Introduction

The purpose of this chapter is to explain the objectives of this research and to describe how the research and the research methodology was designed.

Section 3.2 discusses the research aims, and objectives are discussed. In Section 3.3 the researcher discussed the various research approaches available regarding research purpose, research strategy, research methodology and selection of the most suitable approach for this research. The main methods for data collection are also discussed along with the risk of validity and generalisability and its mitigation strategy. In section 3.4, the researcher discusses the adopted research methodology describing the three phases of this research and section 3.5 provides a summary of this chapter.

3.2 Research Aim and Objectives

The literature review in chapter 2 helped in the identification of the current trend in the research of software obsolescence along with the cost estimation for the resolution of software obsolescence.

The aim of the research is:

To develop a software Obsolescence Cost Analysis Framework to predict the Software Obsolescence Resolution cost of custom build software at early stages of the product life cycle

The key objectives of this research which relates to address the research gaps in the literature review are:

- To understand the current practices and state of the art in software obsolescence and cost estimation.
- To identify the key cost drivers for software obsolescence resolution of a custom build real time software.
- To develop a framework with process and tools to understand, manage and mitigate software obsolescence and predict the cost of Software Obsolescence Resolutions.

- To develop a cost model to predict the resolution cost of Software Obsolescence in custom build real time software for Defence.

In the below section, the researcher looked at the available research strategies that have helped in the development of the research methodology used for this research.

3.3 Research Context

It is very important to understand the context of this research to map the research methodology accordingly. The main attribute that defined this research context are the research gaps identified through the literature search. From the literature review as described in Chapter 2, it is evident that there is a substantial issue in the lack of research in software obsolescence and accepted approach to mitigate the software obsolescence. There is lack of research in identifying and predicting the cost of software obsolescence resolutions at early stages of the project life cycle. It is also evident that the bulk of the research undertaken is to solve the software obsolescence issue reactively. There are several tools and models that estimate the cost of software maintenance, however there are no tools/model to estimate the cost the software obsolescence resolutions. Hence this research is focused on two different areas; the first part of the research is on software obsolescence and the second part of the research is focused on the predictive analysis of the cost or cost estimation.

3.4 Research Purpose

Looking at the aim, objectives, and the research context the author has used a hybrid of exploratory and explanatory approach to this research. At the initial stages of the research where there is only limited information on software obsolescence available, an exploratory approach was used and when the author wanted to forge the relationship between the software obsolescence and the cost of resolutions an explanatory approach was undertaken.

Based on the research purposes there are three types of research (Robson C, 2002); Exploratory, Descriptive and Explanatory. However, Robson C, 2002, also

identified a fourth type called correlation research which describes the relationship between various aspects of situations.

Exploratory and/or descriptive research purpose is used to develop an initial, rough description of some social phenomenon. This is necessary when little is known about the subject being investigated. This should provide a detailed and accurate picture of the phenomenon. Whereas in explanatory research the researcher allows it to relate an event, observation, or other phenomenon to a general law.

3.5 Research Philosophy

Burns R, 2000; Gummesson E, 1991; Kumar R, 2005 suggested that there are two types of research design based on the investigation point of view and they are qualitative and quantitative, which is also known as flexible and fixed (Johnson and Harris, 2002; Robson C, 2002) . According to Galliers, 1992; Robson C, 2002; Walsh, 2001 Qualitative research is also known as positivistic/scientific whereas quantitative research is also known as naturalistic or interpretive research. The approach taken for this research is described further down in this chapter.

3.5.1 Qualitative Research

Qualitative research strategies are considered especially powerful and appropriate for evaluating process. These are aimed at elucidating and understanding the internal dynamics of operations. They focus on the questions such as “what are the factors that come together to make this situation what it is?” “What are the strengths and weakness of the situations?”

Qualitative research is based on an exploratory approach, where most of the data is collected by interviews, observation, and surveys (Robson C, 2002). The data is collected in the form of words and observation rather than in a numerical format (Johnson and Harris, 2002). The strengths and weakness of qualitative research are shown below in Table 3-1 (Javier Romero Rojo, 2011).

Table 3-1 Strength and weakness of Qualitative Research (Javier Romero Rojo, 2011)

Strength	Weakness
Direct contact with the participants	Time consuming to set up
Studies the subject in its entirety	Problems with validity and reliability
Direct encounter with the world	Problems of anonymity
Allows direct encounter with the world	Possibility of being biased

Positivism and Constructivism

Positivism and/or constructivism are probably the basic representative of a list of alternative paradigms to inform and guide qualitative inquiry (Miles and Huberman, 1994). Positivism has got explanation as a central aim, which means that if we you can relate an event, observation, or other phenomenon to a general law, then you have explained it (Robson C, 2002). Positivists look for the existence of a constant relationship between events. Positivism entails ontological assumptions about an ordered universe made up of discrete and observable events (Balaikie, 2000).

Positivism often involves the use of existing theory to develop a hypothesis to be tested during the research process. The main principle of positivistic research philosophy can be summarised below:

- There is no difference in the logic of investigation across the sciences
- The research is aimed for explanation and prediction
- An inductive reasoning should be used to develop a hypothesis
- Science should be value free, and it should be judged only by logic

Typical research methods used for positivism includes deductive analysis, highly structured, collection of large samples, measurements etc.

Constructivism, within social science there is almost invariably a rejection of the view that 'truth' about the social world can be established by using natural science methods. This is essentially because of the nature of the "subject matter" of social science, expert. People, unlike the object of the natural work, are conscious, purpose actors who have ideas about their world and attach meanings. Constructivism researchers tend to use research methods such as interviews and observations which allow them to acquire multiple perspectives. Positivism or constructivism are probably the basic representatives of a list of alternative paradigms to inform and guide qualitative inquiry (Miles and Huberman, 1994). Table 3-2 below shows difference between Positivism and Constructivism.

Table 3-2 Differences between positivism and constructivism

Issue	Positivism	Constructivism
Human Interest	Should be irrelevant	Are the main drivers of science
Explanations	Must demonstrate causality	Aim to increase general understanding of situations
Research progress	Hypothesis and deductions	Gathering rich data from which ideas are introduced
Concepts	Need to be operationalised so that they can be measured	Should incorporate stakeholders' perspectives
Units of analysis	Should be reduced to simplest terms	May include the complexity of 'whole' situations
Generalisation	Statistical probability	Theoretical abstraction

Sampling requires	Large number selected at random	Small number of cases chosen for specific reasons
-------------------	---------------------------------	---

In this research, researcher has collected the data by means of case studies, surveys, interviews, and observation (Robson C, 2002). The researcher was able to modify the research questions as a result of the information obtained following an iterative process (Easterby-Smith, Thorpe and Lowe, 2002). This research design is suitable for the study of dynamic process where it is aimed to develop or discover new concepts instead of imposing preconceived ideas.

3.5.2 Quantitative Research

Quantitative research methods can be described as the method to describe and measure the level of occurrences based on numbers and calculations. The questions such as how many and how often are asked in quantitative studies. The data collection methods used are based on numbers and mathematical calculations. Quantitative data collection is based on random sampling and structured data collection. Findings of these studies are usually easy to present, summarise and generalise. Robson C, 2002 described as an approach used when the phenomena object can be quantified and hence the data is analysed numerically (Johnson and Harris, 2002). Burns R, 2000 characterise quantitative research into replication, operational definition, hypothesis testing and control. A quantitative approach provides the researcher with full control of the environment and the experimental conditions, while staying detached from it so that any influence in the result is minimised (Robson C, 2002). This approach requires a 'fixed design' to provide the benefits, but it brings a lack of flexibility as a drawback (Robson C, 2002). The main strengths and weakness of the quantitative research are shown in below Table 3-3 (Javier Romero Rojo, 2011).

Table 3-3 Strengths and weakness of quantitative research

Strength	Weakness
Results are replicable	Removed from everyday life
Results are verifiable	Difficult to respond to environment
Offers control and precession	Does not account for people's unique experiences
Illustrates casual effects	Lacks flexibility

(Burns R, 2000) compared the qualitative and quantitative research approaches as shown in the Table 3-4 below based on assumptions, purpose, methods and role.

Table 3-4 Comparison of qualitative and quantitative research based on assumptions, purpose, methods and role.

	Qualitative Research	Quantitative Research
Assumptions	Reality -socially constructed	Facts and data have an object reality
	Variables complex and difficult to measure	Variable can be measured and identified
	Events viewed from informant's perspective	Events viewed from outsider's perspective
	Dynamic quality to life	Static reality to life
Purpose	Interpretation	Prediction
	Contextualisation	Generalisation
	Understanding the perspective of others	Casual explanations

Methods	Data collection using participants observation, unstructured interviews	Testing and measuring
	Concludes with hypothesis and grounded theory	Commences with hypothesis and theory
	Emergence and portrayal	Manipulation and control
	Inductive and naturalistic	Deductive and experimental
	Data analysis by themes from informants' descriptions	Statistical analysis
	Data reported in languages of informants	Statistical reporting
	Descriptive write up	Abstract impersonal write up
Roles of researcher	Research as instrument	Researcher applies formal instrument
	Personal involvement	Detachment
	Emphatic understanding	Objective

Combining the use of qualitative and quantitative approaches can strengthen the research (Greene and Caracelli, 1997; Tashakkori and Teddlie, 2003). (Easterby-Smith, Thorpe and Lowe, 2002) suggested that both qualitative and quantitative approaches are needed to be considered to conduct research. Each approach is undertaken at different phases of this research. Based on the comparative analysis, qualitative research approach is deemed to be the best approach for majority of the research phases. These phases include the current practices in managing software obsolescence, developing the framework and validation of the same, however a quantitative research approach was undertaken where it

was necessary to undertake numerical analysis of the data is required. Therefore, a combination of both types of approach was required to complete the aim of this research.

3.5.3 Approaches in Qualitative Research

There are several research strategies that can be applied for Qualitative research. (Creswell J. W, 1998) identified five strategies namely

- Biography
- Phenomenology
- Grounded theory
- Ethnography
- Case Study.

Table 3-5 below lists out the research strategy with the data collection implications adopted from “collecting Qualitative data: a field manual for applied research” by Guest, Namey and Mitchell, 2013.

Table 3-5 Research strategy and how this was used in this research

Approach	Defining Features	Data Collection Implications	Application in this research
Phenomenology	Focuses on individual experiences, beliefs, and perceptions.	Questions and observations are aimed at drawing out individual experiences and perceptions	This approach is partially used in this research to consider the individual experience in managing software obsolescence however needed to be vary of their perceptions.
		In- depth interviews and focus groups are ideal methods for collecting	

		phenomenological data	
Ethnography	Oriented towards studying shared meanings and practices	Questions and observations are generally related to social and cultural process and shared meanings within a given group of people.	Not particularly used in this research, however there was a need to understand shared meanings and approaches from an organisational point of view. This was applied during initial data collection workshops
	Can have a contemporary or historical focus	Traditionally, it is associated with long-term fieldwork, but some aspects are employed in applied settings	
Inductive Thematic Analysis	Draws on inductive analytic methods	It requires generation of free-flowing data	This research strategy is extensively used in this research by deploying data collection methods such as structure and semi structured interviews, workshops etc. the data thus collected are analysed for common themes,
	Involves identifying and coding emergent themes within data	In-depth interviews and focus groups are the most common data collection techniques associated with this.	

	Most common analytic approach used in qualitative inquiry	Notes from participant observation activities can be analysed using this but interview/focus group data are better	difference, and uniqueness.
Grounded Theory	Inductive data collection and analytic methods	In- depth interviews and focus groups are the most common data collection techniques as associated with this.	This methodology is well used in this research specially to generate the Software Obsolescence Cost Analysis Framework form the data collected using interviews and case studies which was further used to generate an analytical model that is used to predict the cost of software obsolescence resolutions
	Uses systematic and exhaustive comparison of text segments to build thematic structure and theory from a body of text	Sample sizes for grounded theory are more limited than for Inductive Thematic Analysis because the analytic process is more intensive and time consuming.	
Case study	Analysis of one to several cases	Cases are selected based on a unique quality.	This approach is extensively used in this research. A case study

	that are unique with respect to the research topic		was used to understand the current practices and trends in the organisation while dealing with software obsolescence. A cognitive case study was used to identify the root cause of software obsolescence at the development phase of the software and further case study was used to validate the framework and the model
	Analysis primarily focused on exploring the unique quality	Question and observations should focus on, and delve deeply into, the unique feature of interest	
Narrative Analysis	Narratives (storytelling) used as source of data	If generating narratives (through in-depth interviews), then questions/tasks need to be aimed at eliciting stories and the importance those stories, hold for participants, as well as larger cultural meaning.	This research methodology was not used in this research.
Mixed Methods	Defined as integrating quantitative	Collection of qualitative data in a mixed methods	This approach is used in this research as there are limited

	and qualitative research methods in one study	study can be informed from a wide range of theoretical perspectives and analytic approaches	source of quantitative data to formulate a framework or to validate the framework, however lack of quantitative data was
	Two most common designs are sequential and concurrent	Research must specify up front, and in detail, how, when, and why qualitative and quantitative datasets will be integrated.	complimented by collecting qualitative data through interviews and case studies

The major data collection methodology employed in this research are Literature Review, Online Surveys, and Interviews.

Literature Reviews

The literature review should provide the existing ideas and knowledge in the area, as well as the methodologies used (Burns R, 2000). The qualitative approach encourages the search for new ways to look at the data, as occasionally new findings may not be fitted into existing concepts and categories (Javier Romero Rojo, 2011). The literature review is a stimulus for thinking rather than just a way to summarise the previous research done in the area, which can restrict the research to consider only existing concepts and categories (Burns R, 2000).

Surveys

A survey is a research method used for collecting data from a predefined group of respondents to gain information and insight into various topics of interest. They can have multiple purposes, and researchers can conduct it in many ways depending on the methodology selected and the research objectives. Surveys

can be applied to any type of research; its use is not advisable to carry out pure exploratory research. The reason is that standardising the questions to reduce ambiguity is a requirement to ensure reliable data, but it clashes with the nature of an exploratory research. Therefore, surveys are used for descriptive purpose (Robson C, 2002). Data collection methods for surveys can be broadly categorised into two groups- Manual and Electronics. (Siva, Nayak and Narayan, 2019). Electronic Surveys are online surveys and the respondent can complete the questions remotely and the output is in electronic format, however in manual surveys the respondent either receives the questionnaire by post and they fill it in by themselves; face-to face interviews, where the interviewers asks the questions and completes the questionnaire in the presence of the respondent or telephone interviews, where the respondent is contacted by phone and the responses are recorded (Robson C, 2002). According to Robson (2002), the major strengths and weakness are listed below in Table 3-6

Table 3-6 Strengths and weakness of survey

Types	Strengths	Weakness
All types	Simple and straightforward approach to study	Data affected by respondent's characteristics
	Highly adaptable	Respondent may not report beliefs, attitudes etc. accurately
	High amount of data standardisation	
Self-administered surveys such	Can be very efficient – large amount of data at low cost in short time	Low response rate. Sample may not be representative

as online surveys	Allow anonymity- can therefore encourage honest answers, particularly regarding sensitive areas.	Ambiguities and misunderstandings in survey questions may not be detected
		Survey may not be treated seriously – may not be able to detect this
Interviews surveys	Interviewer can ensure questions are clearly understood	May be affected by interviewer's characteristic – could influence response.
	Presence of interviewer can encourage participation and involvement	May be affected by interactions of the interviewer and respondent
		Respondent may consider they are not anonymous and so be less honest and open

Interviews

An interview is a technique which involves asking open ended questions to converse with respondents and collect specific data about a subject. Interviews are very similar to focus groups and surveys when it comes to collecting information but are entirely different in the operation. Focus groups are largely restricted to a small group of people, however interviews are conducted with a sample and the key characteristics they exhibit is their key focus.

There are three major types of interviews: Fully structured, semi- structured and unstructured (Babbie, 2007; Javier Romero Rojo, 2011; Robson C, 2002).

Fully structured interviews

Structured interviews are extremely rigid in their operation and allow very little or no scope of prompting the participants to obtain and analyse the results. Structured interviews employ the use of an interview schedule which contains structured and explicit questions that do not allow room for veering off the topic in questions (Javier Romero Rojo, 2011). Because of these characteristics this approach is mainly suggested for surveys and opinion polls rather than quantitative research (Robson C, 2002).

Semi Structured Interviews

Semi structured interviews offer a good amount of flexibility to the researcher to question the respondents along with maintaining basic interview structure. This gives more flexibility than a fully structured interview and also facilitates building rapport between interviewer and interviewees (Burns R, 2000). However, it would be difficult, and time consuming to compare and analyse data provided by various respondents (Javier Romero Rojo, 2011). The flexibility of semi structured interviews allows the interviewers to pursue a series of less structured questioning and permits the exploration of spontaneous issue raised by the interviewee to be explored (Javier Romero Rojo, 2011). The wording of questions can be flexible and facilitates different levels of languages to be used and clarifications to be made by the interviewer (Berg. B, 2009).

Un-structured Interviews

Unstructured interviews are usually described as conversations held with a purpose in mind which is to collect the data about the research. These interviews have the least number of questions as they lean more towards a normal conversation but with an underlying subject. During unstructured interviews the interviewer and interviewee have a conversation about a specific topic in response to the interviewer asking broad, open-ended questions (Javier Romero Rojo, 2011). There is no set answer to the questions and the interview follows the direction of the participant's responses (Moyle, 2002). These types of

interviews have open-ended questions that enable to go in-depth, clear up any misunderstanding and usually lead to unexpected answers (Robson C, 2002).

The strengths and weakness of each interview style and the aim of this research dictated the data collection method that was used in this research. The Table 3-7 below shows the strengths and weakness of different interviews.

Table 3-7 Strengths and weakness of interviews

Type	Strength	Weakness
Structured Interviews	Structure interviews focus on the accuracy of different responses. Answers obtained can be collectively analysed.	Limited scope of assessment of obtained results
	Large sample of target population can be obtained	The accuracy of information overpowers the detail of information
	The interview procedures are easy	Respondent are forced to select from the provided answers
	Replication across multiple samples is easy	A significant amount of time is required for a structured interview
	It generates reliable results	
Semi Structured	Questions of semi-structured interviews are prepared before the scheduled Interview which provides the research with time to prepare and analyse the questions	Participants can question the reliability factors of these interviews to the flexibility offered

	It is flexible to an extent while maintaining the research guidelines	Comparing two different answers becomes difficult as the guideline for conducting interviews is not entirely followed.
	Researcher can interview or ask questions in the format they prefer	No two question will have the exact same structure and the result will be an inability to compare and infer results
	Reliable data can be collected	
	Flexible structure of the interview	
Unstructured	Helps to gain insight in extreme detail without much conscious effort	As there is no structure to the interview process researcher can take time to execute these interviews
	Participants can clarify all their doubts about the questions and the researcher can take each opportunity to explain the intention for better answers	The absence of standardised set of questions and guidelines indicates that the reliability of unstructured interview is questionable
	There is high flexibility on the questions asked.	The ethics involved in these interviews are considered borderline upsetting.

3.5.4 Research Evaluations

Easterby-Smith, Thorpe and Lowe, 2002 summarises the criteria for validity, reliability, and generalisability as

Validity – does the study clearly gain access to the experience of those in the research setting.

Reliability- is there transparency in how sense was made from the raw data?

Generalisability – do the concepts and constructs derived from the study have any relevance to other settings?

3.6 Research Methodology Adopted

3.6.1 Selection of Research Approach

The above reasoning provided the researcher with the justification of appropriate research methodology selection. The summary of the selection, highlighted in yellow is detailed in the Figure 3-1 below and described further in the section below.

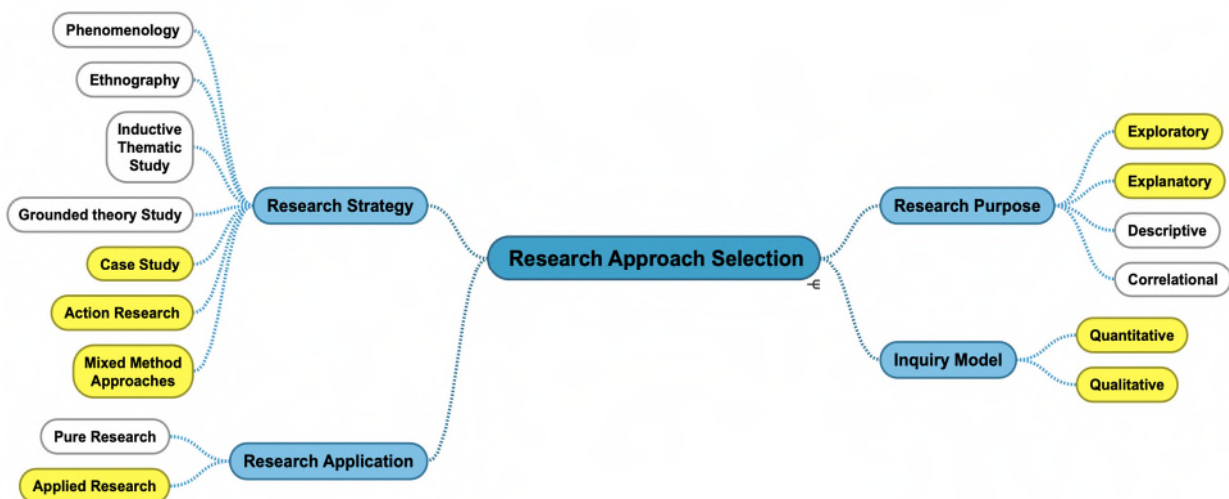


Figure 3-1 Research approach selection

3.6.2 Research Methodology Adopted

Due to the type of information expected to be collected along the research process, an inductive approach has been applied. A case study-based research strategy was deployed to gain the contextual understanding and develop a framework for the software obsolescence cost analysis. This approach will enable the generalisation of the framework developed for the Defence and aerospace domain. The proposed research methodology which is shown in Figure 3-2 below is mapped into four phases. Table 3-8

Table 3-8 Phases undertaken for this research

Phase	Activities
Phase 1: Analysis of current situations	6 x Case Studies Online survey Expert interviews Literature Review
Phase 2: Requirement derivation	Literature Review Data analysis Cognitive Case Studies
Phase 3: Framework development	Process development Tools development
Phase 4: Verification and Validation	Process Validation Tools Validation 5 Case studies

Phase 1: Analysis of Current Situation

This is the very first phase which is dedicated to gain the following

- Gaining the contextual understanding
- Defining the research protocol
- Capturing the current practices on software obsolescence and costing of the same in Defence and aerospace industries

To achieve the above, an extensive literature search on obsolescence, software obsolescence, obsolescence management and software obsolescence costing was undertaken. Interaction with the International Institute of Obsolescence Management (IIOM) allowed gaining a good understanding about the context of this research, however it was soon noted that the expertise in software obsolescence in IIOM was very limited. The enrolment in the IIOM not only allowed gaining access to the latest information about software obsolescence issues in the industry along with possible solutions, but also to come in to contact with current experts in software obsolescence management from various organisations across UK and world-wide. Interview and case study questions was developed by undertaking brainstorming exercise and from literature reviews which was then piloted and validated by industry partners.

Several documents such as business case, obsolescence management plan, bid documents and other commercial documents from the key stakeholders of the industry partners and MoD which was publicly available was reviewed. Online survey was another way that the researcher collected the data. The online survey was sent across to the defence and aerospace industries.

The research protocol thus developed was used to capture the current practices by means of semi-structured interviews and workshops with relevant key stakeholders such as bid managers, project managers, technical leads, costing individuals from various organisation who participated in this study. Further six cases studies were undertaken on individual large MoD (Ministry of Defence) projects which covered all the domains such as Air, Land, Maritime and ISS/ISTAR Capability areas.

All the data thus collected was then analysed to identify the commonality and differences in them and was mapped using the mind map tool. These initial findings were then transcribed to the project requirement, which was then validated at the SCAF and IIOM workshops, where various industry partners were invited.

Phase 2: Requirement Derivation

This phase of the research is focused on understanding and drawing up the need or requirement for this project. All the data that was collected during the “Analysis of the current situation” phase was analysed and scrutinised in detail. The study started focusing on understanding the commonality in the data that was gathered and which was largely focused on the need for software obsolescence management let alone predicting the cost of resolution of the software obsolescence. The next was to focus on the differences in the data that was collected, with focus on how each company and/or the project team deals with software obsolescence individually.

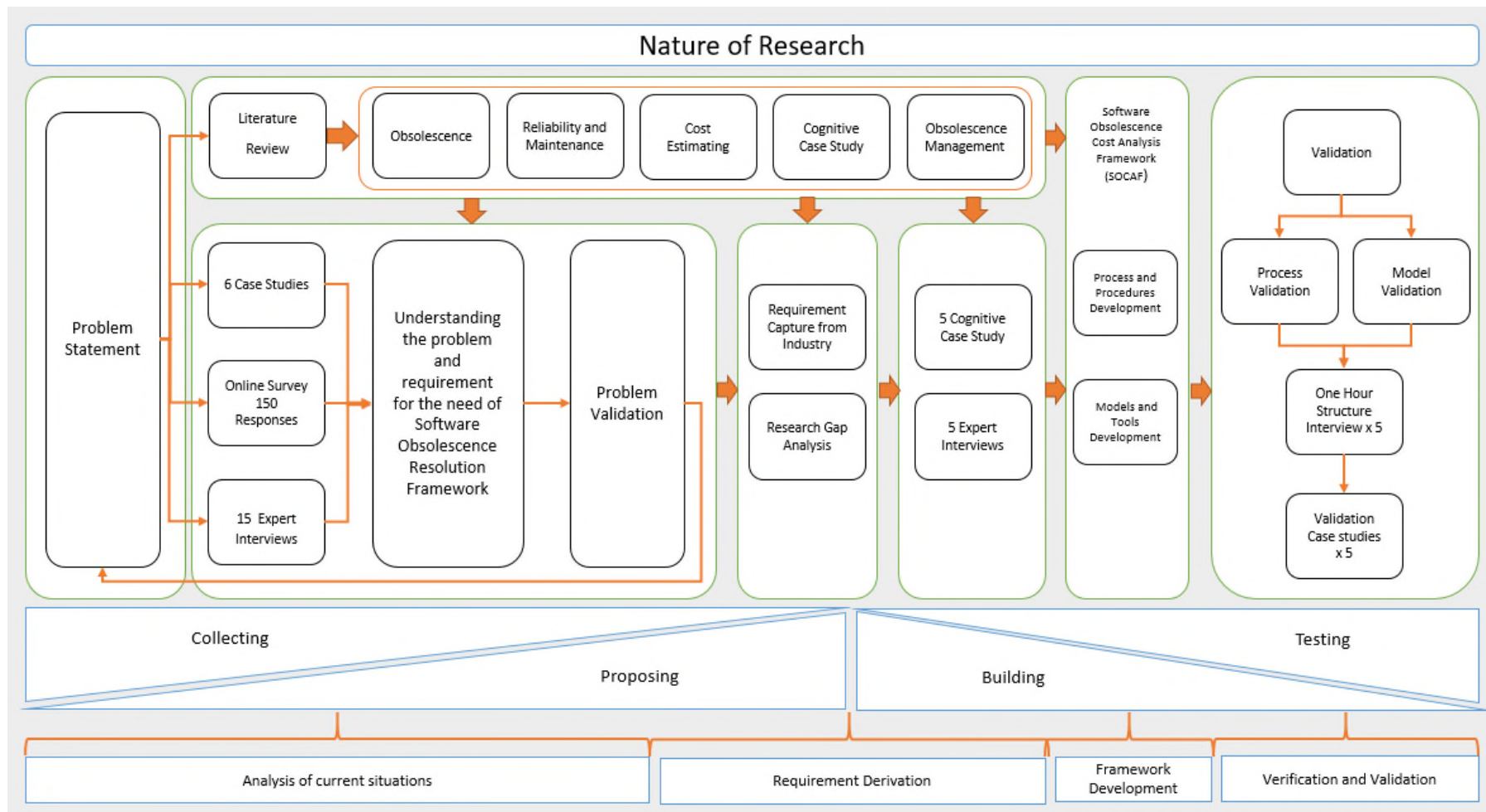


Figure 3-2 Research methodology used for this research

The key commonality that was identified was the recognition within industry on the issue of software obsolescence, however with a serious lack of knowledge and research in identifying, managing, and resolving software obsolescence. The key differences identified was the different approaches that each industry partner undertakes to resolve the software obsolescence issue. The unique aspect in this phase was recognising not having a standardised way to identify, manage and resolve software obsolescence with a clear research gap in estimating the cost of resolution of software obsolescence. It was also noted that majority of all the obsolescence management plans that were reviewed did not have a strategy for software obsolescence mentioned.

During the first phase of the research, it was realised that the technical experts, bid managers and project managers had only limited data on software obsolescence; as such the researcher undertook semi structured interviews of software developers, designers and system engineers following which it became evident that the software developers were able to identify/ predict the software would go obsolete during the development phase software. This was a key source of information that needed to be explored. The researcher wanted to prove this hypothesis and undertook a cognitive case study. Six software developers from Defence and aerospace industry with various levels of expertise were invited for the cognitive case study which was then followed up by unstructured interviews.

These key activities in phase two of this research helped in firming up the requirement for this project. These findings and the requirement for a software obsolescence cost analysis framework not only helped in identifying, managing, and providing resolution for software obsolescence but also to estimate the cost of resolution which was validated with industry partners via workshops through IIOM, SCAF and ICEAA.

Phase 3: Framework Development

This phase of the research is focused on the development of a framework for estimating the cost of software obsolescence resolution, to be used during the life cycle of support contracts at early stages of the project.

The study started focusing on the development of cost estimating framework for software obsolescence. The research protocol was deployed to collect data about the existing cost related documents, tools, company documents, reports, and expert interviews. The data that was collected was not just related to cost but also on how these companies manage their software obsolescence and the mitigation actions they carry out. The data was then analysed with the literature review to develop various aspects of this framework, which was then used to model the cost of different resolution approaches for software obsolescence. The framework consists of several processes and tool sets. The following processes and tools in Table 3-9 below was developed as part of this phase.

Table 3-9 Process and tools that was developed as part of the Software Obsolescence Cost Analysis Framework

Process	Tool
Obsolescence Management Level	Software Obsolescence Data Collection form
Software Obsolescence Resolution Approach	Software Obsolescence Resolution Model
Software Obsolescence Complexity Matrix	Software Obsolescence Resolution Cost Model

The framework underwent an evolutionary development with each iteration being validated by the industry partner who took part in the validation workshop. This approach reduced the subjectivity and bias that could be expected from the experts. A phased development process was used for the development of the framework and tool.

- Exploratory Phase – Framework Development version 1
- Development Phase – Framework Development version 2
- Refinement Phase- Framework Development version 3
- Validation Phase – Framework Development Version 4 and final version

Phase 4: Verification and Validation (Demonstration)

The final phase of this research is about the validation of the framework and the respective prototypes of the tool set that was developed. The validation of the framework and verification of the tool set was undertaken by qualitative and quantitative assessment as explained in Figure 3-3. The validation was initiated with planning meeting with key industry partners, which identified relevant experts in the organisations who could critically appreciate the effectiveness of the methodologies. The methodologies were demonstrated to the experts and feedbacks were captured using questionnaire and semi structured interviews. The same principle was applied to the verification of the tool sets. The observations for both verification and validations were triangulated with literature reviews and previous rough order magnitude estimates from the companies. The first validation was undertaken with BAE systems of an early iteration of the framework; and following feedback the framework incorporated the changes which was again validated at ICEAA, DMSMS and SCAF workshops, allowing the iterative development of the framework and the model. A total of five case studies from five project teams participated in the verification and validation workshops with real data from current and past projects used and analysed. The predicted cost was then compared to the available actual cost. Any gaps in the actual data were covered by expert evaluations from the workshops.

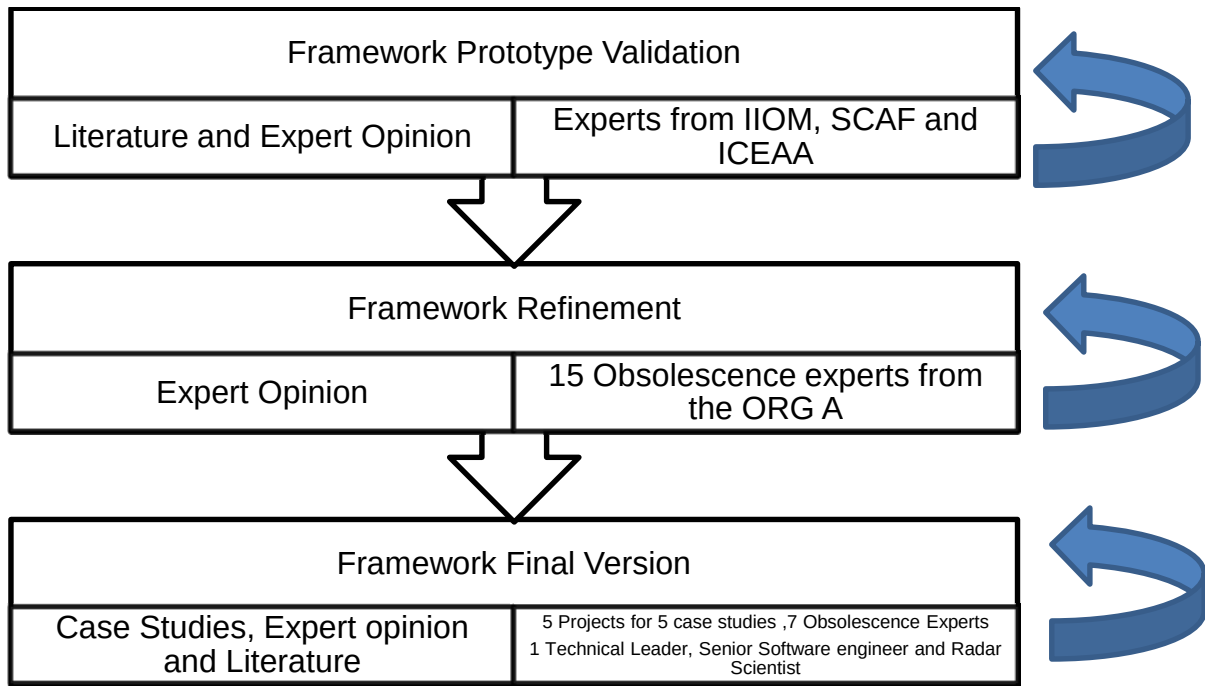


Figure 3-3 Summary of validation and verification

3.7 Summary of Observation

In this chapter the researcher presented the research aims and objectives, the various research methods that were reviewed and the reasoning behind the selection of the appropriate and suitable research method were discussed, alongside various data collection methods reviewed and presented.

In the first section of the chapter the aim of the research was described and how the key objectives of the research were developed. The following section looks at the research gaps that was identified during literature reviews which formulated the need for the research context.

In section 3.4 and 3.5 researcher explored the research purpose and philosophy that was undertaken and explored the strengths and weaknesses of each research method and described the reasons for use of the selected research method.

The data collection methodology that was used in this research was explored in the next section where the use of case studies, interviews, surveys were

explored. In the final section of this chapter the researcher described the research methodology adapted to complete this research.

The following chapter will present the current practices that are in place for managing software obsolescence in UK Defence and Aerospace industries

4 Current Practices

4.1 Introduction

This chapter explores the current practices and challenges within the UK Defence and Aerospace industries in terms of managing Software Obsolescence and estimating the cost of its resolutions.

The first section of this chapter covers the detailed research methodology that was undertaken to gather information on the current practices and challenges of managing software obsolescence in Defence and aerospace industries across UK and in the production of whole life cost estimates.

In the second section, the researcher goes through the data collection methods that were deployed such as Case Studies, SME interviews, Online Surveys, and cognitive Case Study. And the final section describes the summary and the key observations related to current practices of software obsolescence in the UK Defence and aerospace industries.

4.2 Detailed Research Methodology

4.2.1 SME Interview

As described in Chapter 2, currently there is very little information available on software obsolescence, whilst extensive literature search was carried out to get the preliminary information on software obsolescence, it was necessary to capture the information directly from the industries. The research methodology to understand the current practices for managing software obsolescence is shown in the Figure 4-1 below.

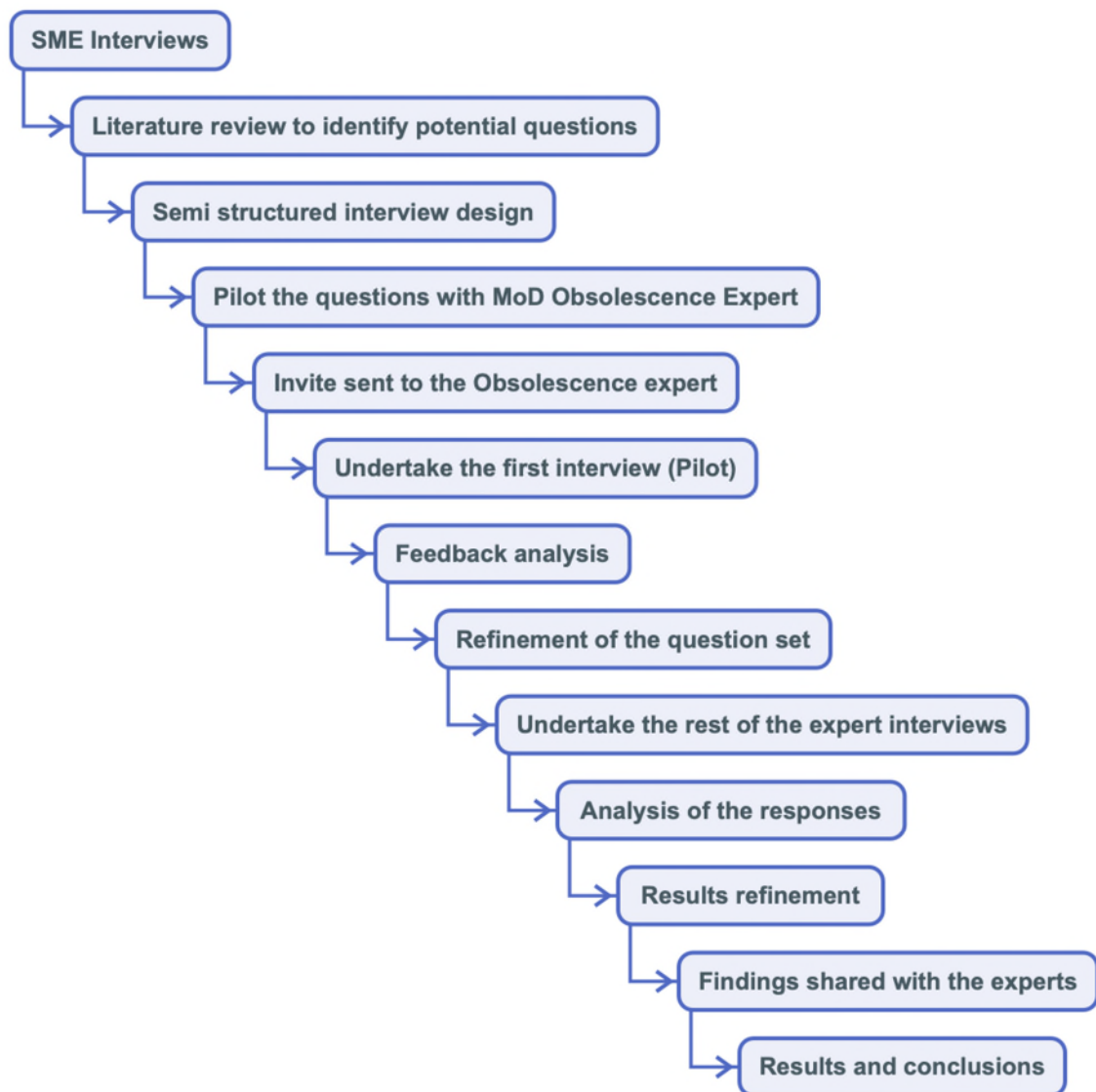


Figure 4-1 Research methodology followed for SME Interviews

A detailed research review was undertaken as described in chapter 2 to identify and design the expert interviews, which allowed the researcher to focus on key topics including definition of obsolescence, obsolescence strategies etc.

To capture the depth of software obsolescence management, its mitigation and analysis, and the current practice in place, a semi structured interview questionnaire was developed (9.4Appendix A). The question set developed consisted of three sections, the first part is covered the Software Obsolescence management, the second part concentrated on the Software quality, standards and maintenance and the final part looked at the cost estimation of software

obsolescence. The examples of questions contained in the questionnaire are listed below.

- What is the difference between software obsolescence and software maintenance?
- What strategies are used to predict and mitigate software obsolescence?
- How do you estimate the cost of software obsolescence resolutions?
- What are the main reasons for software obsolescence?

The questionnaire was piloted on a resident MoD (UK) Obsolescence specialist to understand the feasibility of the questions and to recognise potential issues. Once the question set was complete, invite was sent out to industry specialists. A total of fifteen experts out of 27 invitees from various industrial organisations in the Defence, aerospace, automotive and academia showed interest and allowed to be interviewed. The demography of the participants is listed in the Table 4-1 below. Interviews were carried out over a period of three months taking into consideration the availability of the experts. After each interview, a detailed analysis was carried out based on the information gathered, to gain a more in-depth understanding through subsequent interviews.

Table 4-1 Demography of experts who participated in the semi – structured interviews

No	Organisation	Role	Years of Experience
1	ORG A	Software Engineer	18
2	ORG A	Senior Software Engineer	25
3	ORG B	Software Developer	15
4	ORG C	Software Developer	17
5	ORG B	Cost Analyst	18
6	ORG B	Cost Analyst	13

7	ORG D	Obsolescence Manager	23
8	ORG E (Academia)	Snr Lecturer	19
9	ORG F (Academia)	Lecturer	15
10	ORG G	Senior Consultant	28
11	ORG G	VP UK and Europe	20
12	ORG H	CEO	31
13	ORG H	VP	21
14	ORG I	Management Consultant	18
15	ORG A	Technical Analyst	19

The information collected through these interviews were systematically analysed to identify key issues and ideas. This summary was presented and validated at the final interview with selected experts. With these interviews it was established that experts deal with software obsolescence differently more often, due to the lack of understanding of software obsolescence issues. Lack of consistency and standard was evident that they could follow for long term sustainment of software obsolescence issues/risk. However, it was possible to triangulate some responses from the interviews to the literature search. This helped the researcher to understand some aspects of software obsolescence. Some of the responses from the experts were similar and encouraging, however there was clear indication that further analysis and research was required to understand software obsolescence.

Data Analysis of the SME Interviews

Majority of the projects that the SME worked on had a combination of integrating old platforms with the latest COTS (Commercial off the Shelf) or vice versa, therefore the most challenging software obsolescence management problem was often found at these interfaces among applications or the integrating glue codes. In COTS world, hardware and software depend on each other heavily, software

obsolescence is also driven by the changes in hardware improvement. Experts who were interviewed confirmed that they pass the obsolescence risk to the contractor but are unable to quantify the risk. It was also evident that the project teams have had very limited capacity or capability to understand and deal with software obsolescence. And therefore, noted that they have a massive skills gap to monitor, predict and manage software obsolescence.

The experts also concurred that the software on its own does not get obsolete, however some of the experts interviewed also pointed out that software plays an important role in making the whole system obsolete. It was also noted that unstable requirements and changes in the Defence security policy could also trigger software obsolescence. All except two of the SMEs agreed that proactive approach to mitigating software obsolescence is cheaper. The remaining two however, noted that reactive approach to mitigate the software is cheaper in cash value, they also agreed that software obsolescence triggered by hardware does get very expensive.

There is a lack of any obsolescence standards, this was clear from the data collected from the interviews, which also showed lack of consistency in defining software obsolescence. The data suggested that the industry undertakes a planned obsolescence, where they release a newer version of the software when the current version does not hold ground in the market.

Similar responses were received from different companies that attended the Component Obsolescence Group conferences. To a direct question on how to manage software obsolescence, Plant Manager of Company A noted the lack of software obsolescence strategy, although having developed a strategy for system obsolescence. Team leader Material and Obsolescence of organisation B also commented that they understood software obsolescence is a substantial issue.

These interviews and discussions gave a good insight on the current concerns and issues relating to software obsolescence, and how the mitigation actions need to be undertaken. However, the results were not conclusive on the standards followed, mitigation actions, resolution approaches, management and

cost estimation of Software Obsolescence, therefore it was necessary to undertake case study of projects from the Defence and aerospace industries.

Detailed analysis of the interviews is shown in the Figure 4-2 below.

SME interview Analysis- General Information on software obsolescence

It has been identified from the interviews that the industry is keen on understanding the software reliability rather than software obsolescence. They also confirmed that software obsolescence is certainly different to hardware obsolescence, however highly reliable software will have lesser issue with software obsolescence. There are dedicated resources and tools to monitor and manage software reliability, however to manage software obsolescence opinion of experts from ORG E and G indicated that the software will not go obsolete, which is true, if nothing changes in the supporting environment.

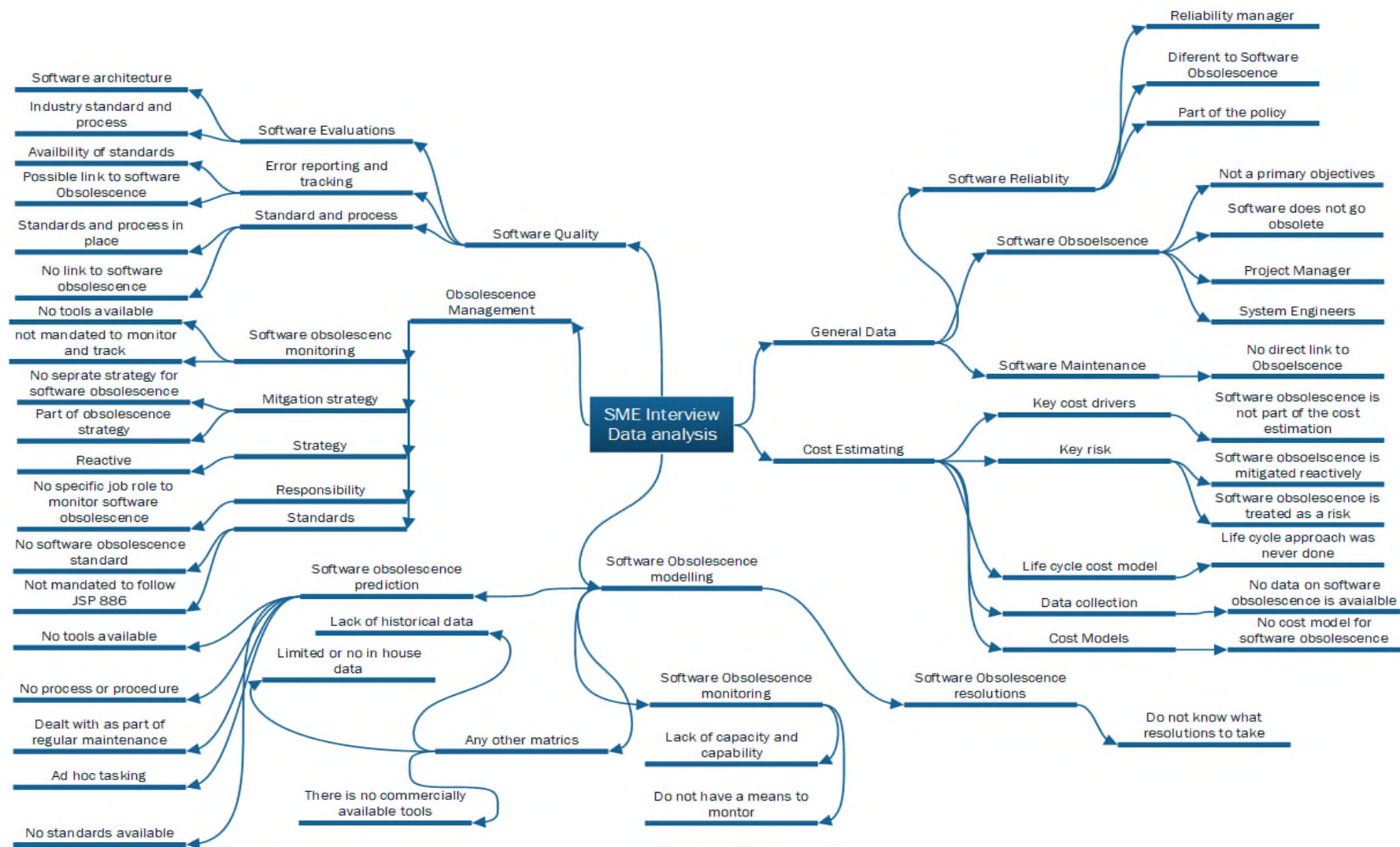


Figure 4-2 Detailed analysis of the SME interviews using mind map

It was also evident from the interviews that there is no direct relationship between software obsolescence and software maintenance. In addition, experts from ORG C and E were of the opinion that software obsolescence is another way the industry makes more profit by planning for obsolescence through making their product obsolete.

SME interview Analysis -Software Quality

When questioned about the software quality and its relationship to software obsolescence it was evident that there is a standard and process that the software developers follow, however this is not related to software obsolescence. Expert from ORG F suggested that error reporting and tracking tool to monitor software quality could be a potential early source of data for software obsolescence, however no relationship was established during the interview.

SME interview Analysis- Software obsolescence Modelling

It was evident from the SME interviews that there is no tool currently available in the market to monitor and manage software obsolescence with a lack of capacity and capability to understand, let alone manage software obsolescence. All 15 experts unanimously confirmed that they are unaware of appropriate resolution technique or approach that they could take to mitigate software obsolescence, however they are aware of reactive and proactive obsolescence strategies. Experts also confirmed the following about obsolescence prediction and data or matrix for software obsolescence.

- No tools are available to predict software obsolescence
- No process and procedure available
- No standard, process and procedure are available; however, MoD (UK) does have JSP 886 but is not mandated
- None or limited data on software obsolescence exists and
- There is a serious lack of historical data

The key findings from these interviews and discussions are listed below.

- Project teams are passing the risk to the contractors/suppliers.
- Lack of skills and resources to predict and monitor software obsolescence.
- Reactive approach often applied to mitigate software obsolescence
- No mandate to use software obsolescence standards.
- Lengthy support contract.
- Unavailability of the data related to software obsolescence.
- Inadequate design documentations and configuration control.
- Unavailability of historic cost data associated with software obsolescence.
- Unavailability of software obsolescence management plan and strategies.
- Inconsistency in obsolescence management plan.

4.2.2 Case Studies

From the industry engagement and interviews, it was evident that the researcher needed more information on the current practices in the industry. As described in Chapter 3, Case studies were undertaken by various methods such as interviews, formal discussions, and engagement with various industry partners. Case studies were carried out with several MoD (UK) projects, which are at the various stages of the project life cycle. Each case study is a project that has got software obsolescence issues/problems to address at various levels. The other selection criteria included the maturity of the project, software dependency, and the value of the project. The research methodology that was followed is shown in the Figure 4-3 below.

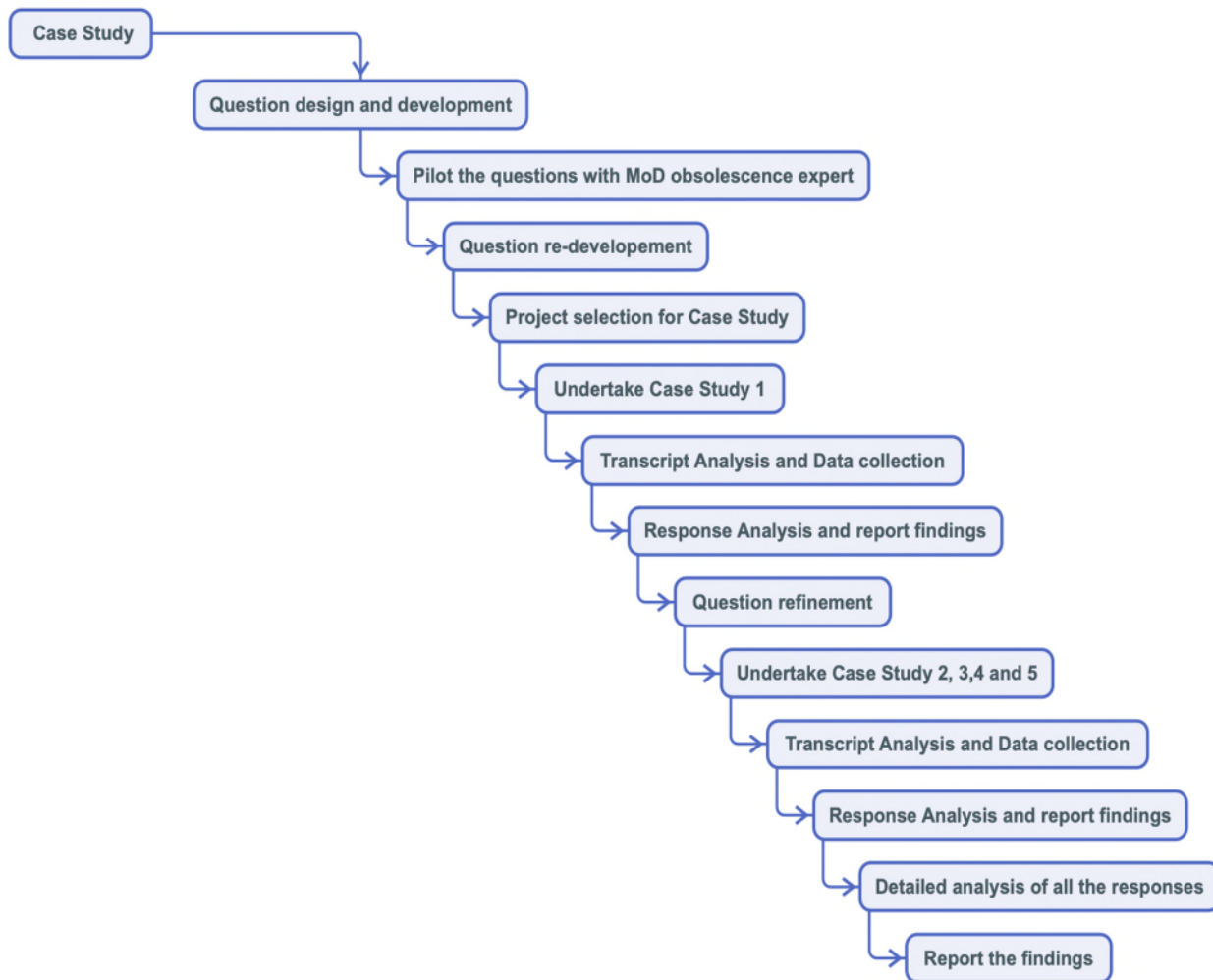


Figure 4-3 Research methodology followed for the Case studies

The first activity was the development of a semi- structured questionnaire based on the information collected during the industrial engagement, interviews, and literature reviews. Due to the exploratory purpose of these interviews a semi-structured questionnaire was developed. The questionnaire was piloted with the project manager of the first project (12 years of experience) and the MoD Obsolescence Manager who was in the post for over 5 years to ensure the quality of the interview questions. Appendix C lists the questionnaires used during the case study interviews. The interview had an average duration of four to five hours. The interviews were broken down into

1. Aims, objectives and purpose of the case study
2. Discussion on the current practices on managing software obsolescence
3. Discussion on the current practices on estimating the cost of software obsolescence resolution

Program managers, project manager, technical leads from the MoD UK projects were selected in the case studies. The selections were made based on the MoD projects. The response was transcribed and analysed, alongside with the documentation provided by these project teams such as logistics and support plan, Obsolescence Management plans and reports etc. They were used to triangulate the information collected during the case study interviews. As a result of the analysis of the responses from each interview, the research protocol was refined and applied to subsequent interviews, following an iterative process. Finally, all the results of the analysis were synthesised, identifying similarities and differences between the information provided by the different experts. The outcome of this study was validated to ensure generalisability and validity of the results.

A total of six case studies the projects within which were entirely from UK MoD, along with 18 interviews were carried out. The transcript of the interviews was analysed, which allowed the identification of the key problems, challenges and issues related to software obsolescence. It also identified a general lack of knowledge on predicting and mitigating software obsolescence. Majority of these projects that were interviewed deals with software obsolescence reactively and recognised that managing software obsolescence proactively will be cost effective.

The key challenges identified by the participating project teams included the difficulty to identify software obsolescence and mitigating these with a suitable strategy. These was in addition to a lack of common definitions, standards and means to monitor software obsolescence and capture data.

Table 4-2 shows the participating project teams from MoD for this case study activity. The total cost of the project was over £1.2Bn.

Table 4-2 case study with participating project teams and the number of interviews.

Project Team	No of Interviews
Project A (ISTAR)	2
Project B (AIR)	2
Project C (ISS)	2
Project D (LAND)	2
Project E (AIR)	2
Project F (HEL)	1

Three of these projects were in the in-service phase, two within assessment phase and 1 in the concept phase. Only four of the projects had any obsolescence management strategy and only one of them have got software obsolescence management strategy; this is understandable given the project came into existence due to the software obsolescence issue that they had with the previous iteration. Whilst, it is not mandated to use software obsolescence standards, three of these projects were informed about JSP 886 and had used it as a reference point. Almost all these projects were keen on mitigating software obsolescence reactively, however recognised that if they had the capacity and capability to, they would have inclined towards engaging software obsolescence proactively. None of these projects maintained the capacity or capability to monitor software obsolescence nor had any historical data to predict the cost of software obsolescence. Table 4-3 below shows the Experience and role of Case Study Participants.

Table 4-3 Role and experience of case study participants

Participants	Experience	Role
Participant 1	25 Years	Senior Support Manager
Participant 2	25 Years	Senior Programme Manager
Participant 3	33 Years	Support Manager
Participant 4	20 Years	Project Manager
Participant 5	10 Years	Software Developer
Participant 6	10 Years	Software Developer
Participant 7	8 Years	Senior ILS Manager
Participant 8	7 Years	Project Manager
Participant 9	5 Years	ILS Manager
Participant 10	5 years	Project Manager
Participant 11	5 years	Project Manager

The case study was largely focused on key topics of discussions such as availability of Obsolescence Management Strategy, Software Obsolescence Management Strategy, availability of software obsolescence management standards, and the strategy that they follow. The case study also looked at monitoring of software obsolescence, availability of historical data on software obsolescence and looks at the cost estimating techniques/methods used to estimate the cost of software obsolescence resolutions. The case study further considered the definition of the software obsolescence for each project.

Project D, E and F being in the in-service phase of the contract were the most appropriate to investigate the software obsolescence issue. Project A and Project C, although being in the Assessment phase; were considered appropriate for the case study given these projects came into existence due to the obsolescence issue with the original delivery.

The Table 4-4 below summarises the findings from the case studies. The detail analysis is described in the next section of this chapter

Table 4-4 Summaries of the key findings from the case study

Issues	Project A	Project B	Project C	Project D	Project E	Project F
CADMID	A	C	A	I	I	I
OMS	√		√	√		√
SOMS			√			
SOM Std			√	√		√
Reactive	√	√	√	√	√	√
Proactive	√		√			√
Monitoring Software Obsolescence	No	No	No	No	No	No
Lack of Definitions	Yes	Yes	Yes	Yes	Yes	Yes
Historical Data	No	No	No	No	No	No
Cost Estimation	No	No	No	No	No	No

Case Study Analysis - Software Obsolescence Management Strategy

Obsolescence management strategy investigates the management, mitigation, planning and resolution of any software obsolescence risk. This document covers every aspect of obsolescence including hardware, software, skills, media, material,

and components etc.

Questions included availability of any obsolescence management strategy and around the strategy usage. Four of the six project teams confirmed having obsolescence management strategy and engaging with any issues according to the management strategy. One of the projects within the in-service phase at the time of the case study did not have an obsolescence strategy. However, even the project teams with an obsolescence management strategy, did not have a specific software obsolescence management strategy, specifically due to not having the capacity or capability to understand and manage software obsolescence. The software obsolescence is therefore captured as a risk and passed on to the OEM. The biggest concern for these projects was the inability to quantify the software obsolescence risk/issue. Table 4.4 shows the detailed responses from each project team.

Only one project (Project C) which was within the Assessment phase managed to maintain a software obsolescence management strategy, however it was noted the team finding it difficult to understand the issue, let alone manage the problem. The lack of capacity and capability led to mitigate the software obsolescence management reactively, which in turn would be expensive to manage.

Case Study Analysis – Software Obsolescence Standards /Strategy

There are no existing specific standards for software obsolescence, whilst for MoD projects JSP 886 is an available standard, however this is an instructional document and is not mandated to use. BS62402 is the British Standard for general obsolescence, however there is limited information on managing and monitoring software obsolescence. IIOM and DMSMS has since released publications on managing software obsolescence, but again they are not a standard accepted by the industry and are not mandated to use.

Project C, D and F were using JSP 886 for guidance, although they were not mandated to use this publication. It was also confirmed in the interviews that there is very little or no guidance on how to manage, monitor and plan for software obsolescence.

Majority of the project teams were dealing with software obsolescence reactively due

to the lack of any guidance or standard, however Project A and Project C were trying to understand a better way to manage software obsolescence proactively. They documented in their Obsolescence management strategy that they will deal with software obsolescence proactively. This was the only example of proactive management documented.

Case Study Analysis - Cost Estimation of Software Obsolescence

In this section, researcher was looking for any standard definition for software obsolescence that the project teams were using. The researcher wanted to explore the possibilities of any commercial or in house tools/techniques or methodology for monitoring software obsolescence and the availability of any historical data by which a Cost model for the resolution of software obsolescence could be developed.

It came as a no surprise that none of the project teams in the case study were monitoring and managing software obsolescence systematically. The flow chart Figure 4-4 below models how each project team's deal with software obsolescence. It was also evident that there is no standard definition for software obsolescence to which the project teams are working. It was also noted that there is a lack of any cost model by which they could predict the cost of resolution of software obsolescence, largely due to lack of any historical data to predict the future.

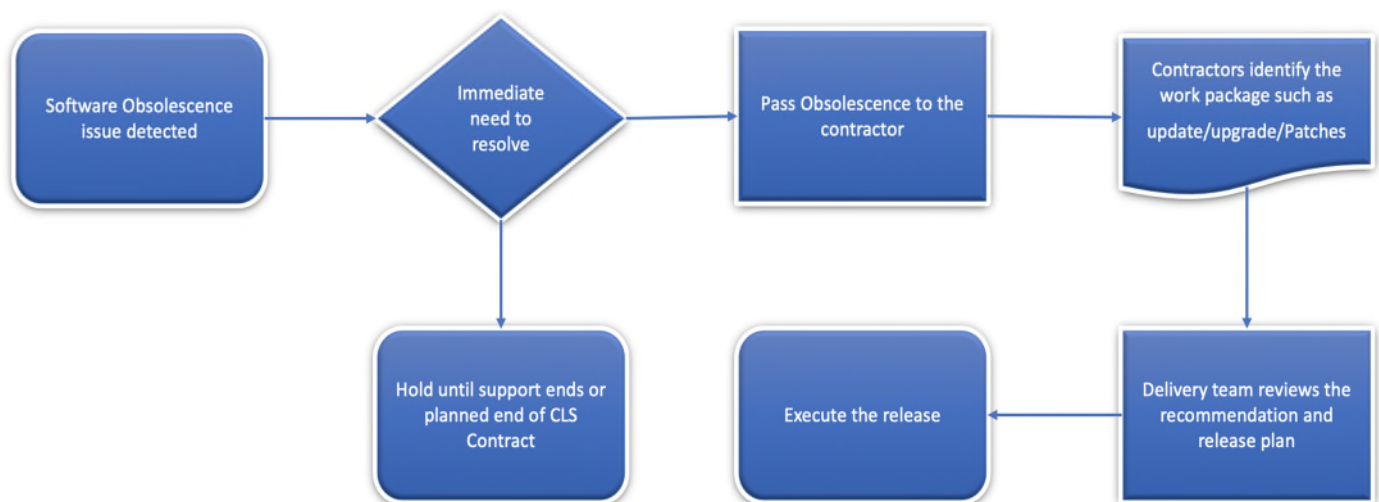


Figure 4-4 Flow chart to show how project team currently deals with software obsolescence.

4.2.3 Online Survey

The Case Studies, SME interviews and Industry engagement did provide sufficient information on the current practices of software obsolescence. However, the data collected needed to be validated.

The questions for online survey were designed and developed from the data that was collected from the case studies, SME interviews, industry engagement and literature search. The methodology used for online survey is described in Figure 4-5.

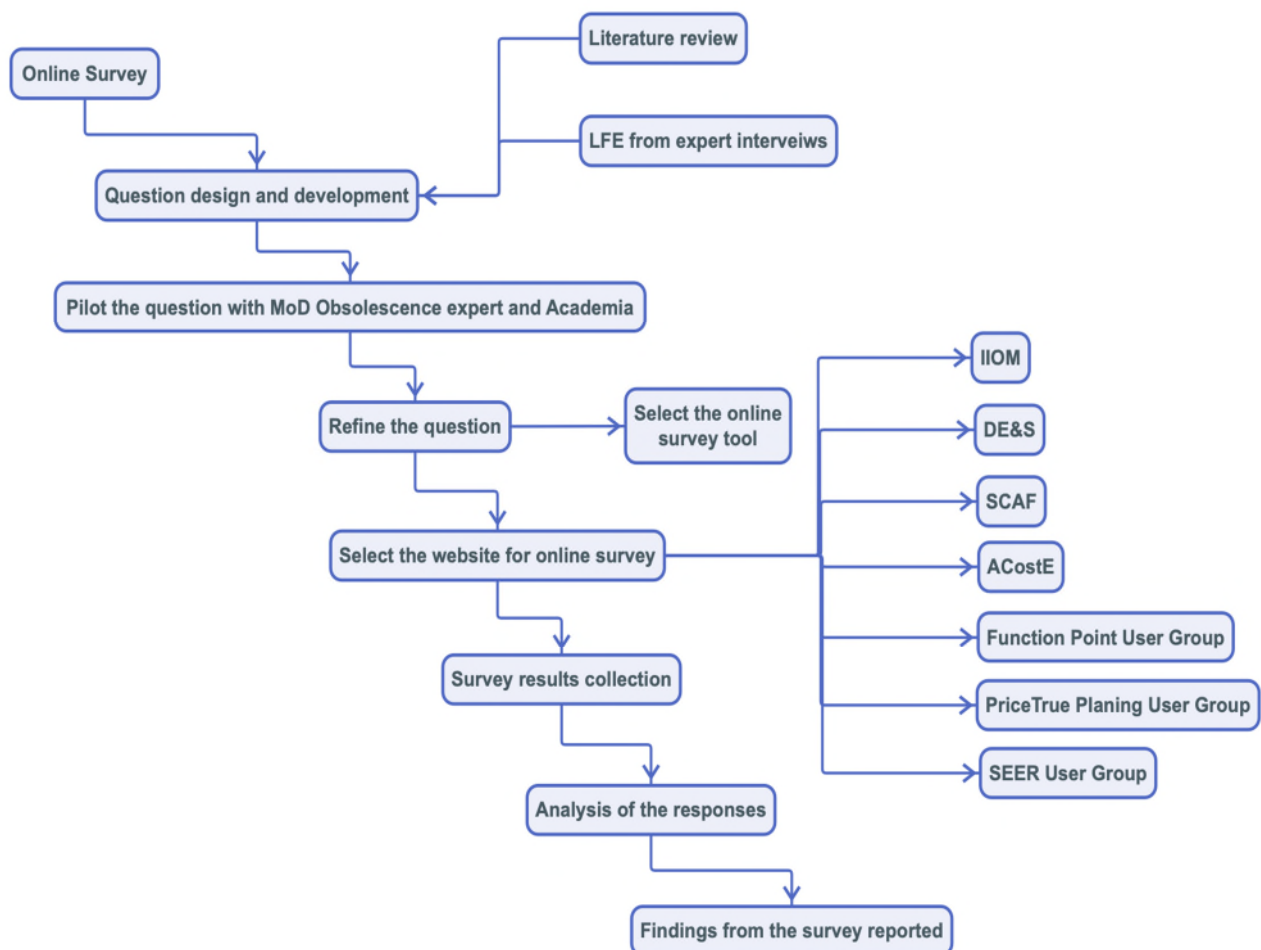


Figure 4-5 Research methodology for online survey

This survey was targeted at anyone who is responsible for software design, software

development, system engineers and others who have interest in software and system obsolescence.

The survey was published for three weeks in early September 2014 (Table 4-5 for the list of websites where the survey was published). A total of 151 responses from Defence, aerospace and automobile industries were recorded.

Table 4-5 List of Websites where the survey was published

Sl. No	Websites
1	International Institute of Obsolescence Management
2	Defence Equipment and Support
3	Society of Cost Analyst and Forecasting
4	AcostE
6	Function Point User Group
7	PriceTrue planning User Group
8	SEER User Group

The survey questions were designed to understand the current process and procedure for monitoring and mitigating software obsolescence. It also tries to find out who is responsible for planning, identification, and mitigating software obsolescence. The latter part was designed to identify the mitigation strategies and to identify the most common mitigation actions for software obsolescence. The summary of the findings is listed below. The analysis of the survey is described below.

Participants Role

Of all the responses, only 12% were software developers and 13% were software engineers (Figure 4-6). Majority of the participants who had shown interests were from various project support functions such as cost management consultant, ILS support functions, Risk manager, commercial manager, and

sales. This suggests that defence and aerospace industry need to identify resources to monitor, manage and mitigate software obsolescence.

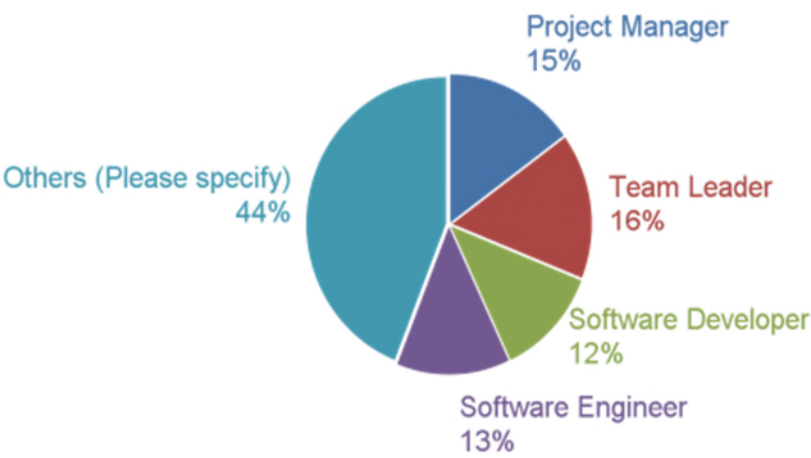


Figure 4-6 Role of participants

About 10% of participants were working in ISS/ISTAR environment, 14% in Land, 20% in Maritime and 24% of participant is from Air platform projects. The other domains recorded in the survey included from medical, communication, financial, banking and transport.

Obsolescence Management Strategy

56% of participants said that they have an Obsolescence Management Strategy (OMS), and they cover various forms of obsolescence issues. Table 4-6 shows the response as to what was covered in their OMS. The respondents were allowed to choose multiple answers.

Table 4-6 Response to element covered in obsolescence management strategy

Answer	Response
Systems	21
Hardware	26
Software	27
Skills	15

Components	14
Material	6
Total	109

If we look closely, it is interesting to note that majority of the respondents cover hardware and software separately in their OMS, although there were 21 respondents who said they cover systems obsolescence. Given for the purpose of this survey the system is classed as a combined hardware and software component, this needs to be investigated further as to understand if the systems covered both hardware and software. It is also worth noting that 14 respondents indicated that they cover components in their OMS, and therefore, it is worth considering if these components included semiconductors and/or electronics; and if so, they should be part of hardware. It is worth noting that the industry is also keen on skills obsolescence, as this is one of the key factors that will make software obsolete.

It is evident from the survey that different companies have different OMS based on their requirements and project. For instance, replacing hardware and software with new approved systems, with the ability to support the system for 25 years is one approach, whereas some organisations identify potential alternative by maintaining adequate Suitably Qualified and Experienced Personnel (SQEP) resources in the OMS. For some obsolescence management plan was integral within the company's governance, functional policies and processes. It has also been pointed out that a good reactive and proactive approach based on IEC 62402:2007 also forms part of OMS.

Software Obsolescence Management Strategy

According to IEC 62402:2007 Software Obsolescence can be managed by a reactive approach such as update, upgrade, patch, bug fixing and emulation as and when the software generates a defect or there are changes in target environment, and via a proactive approach by managing software obsolescence through early engagement to understand and predict software obsolescence, monitoring software obsolescence

and formulating strategy to mitigate software obsolescence. 57% of the total 44 respondents said they would like to engage software obsolescence reactively as and when it happens. (Figure 4-7)

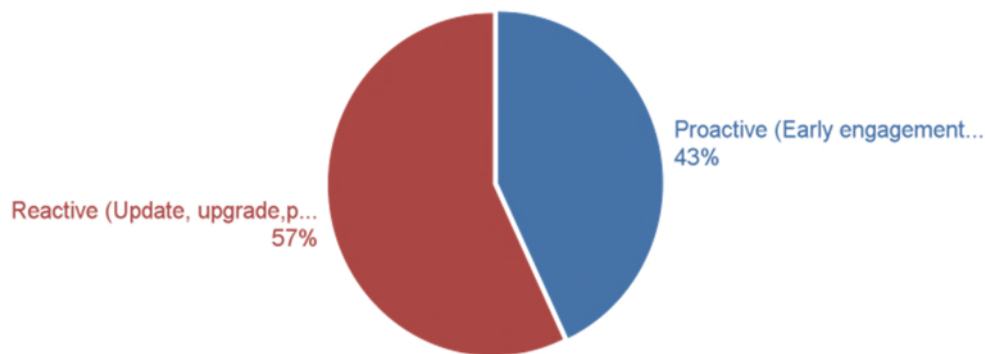


Figure 4-7 Response on Software obsolescence strategy

It was also evident from the online survey that currently project teams are not mandated to use any software obsolescence management standards. 75% of the 43 respondents agreed that they do not follow any software obsolescence management standards. This could also be a reason for 47% of the respondents said they plan for software obsolescence at the in-service phase of the project life cycle, with only a handful of respondents planning for software obsolescence at the Assessment phase of the project life cycle. Table 2-1 below gives details of the response. The respondent was allowed to choose multiple answers.

Table 4-7 Planning for Software Obsolescence

Answer	Response
Concept	6
Assessment	11
Demonstration	1
Manufacturing	5

In-Service	20
Total	43

Software Obsolescence Monitoring

It is very important to monitor software obsolescence, as this will help the project managers to proactively engage, plan and mitigate software obsolescence issues. Only 44% of the participants monitor software obsolescence (Figure 4-8). Of those who monitor, there is a huge inconsistency on the methodology for monitoring software obsolescence. This again points out the fact that there is no consistent approach to monitor software obsolescence. Current methodology used are evidence based and by direct contact with the owner/creator or OEM, relying on OEM obsolescence management plan, following OEM updates and support model, following software releases, patches, and industry standards etc.

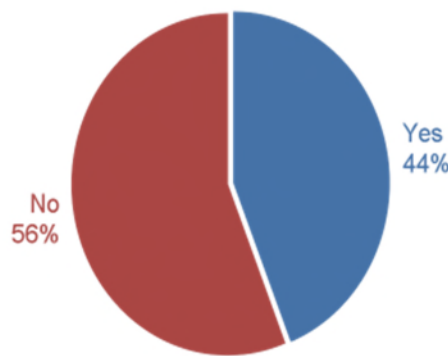


Figure 4-8 Monitoring Software Obsolescence

Software maintenance is the process of managing software regularly by patching, bug fixing, updates and undertaking major upgrades during the productive life cycle. It is evident from the survey that there is inconsistency in defining software obsolescence and software maintenance. It is also evident from the survey that

hardware obsolescence trigger software obsolescence and vice versa, and that hardware is a key driver for software obsolescence.

Software Obsolescence Mitigations

IEC 62402 advises the user to have software obsolescence plan and robust mitigation plan. Different organisations have different views on what should be in software mitigation plan, such as: -

1. ESCROW agreement to maintain the legacy software
2. System dependency mapping
3. Proactive approach
4. Through life support plan

There is a growing tendency for the risk of software obsolescence to be passed on to the contractors or pushed to the supply chain to manage and mitigate software obsolescence. This is due to availability-based contracting where the risk of software obsolescence is placed on the supplier to ensure the capability is maintained. There is also a lack of capacity and capability to undertake these risks and to manage this in-house which historically has been difficult to quantify due to lack of historic data.

It was evident from this survey that there is some commonality in mitigating software obsolescence. The most common mitigation is to pass the risk to the supply chain or the OEM. Several other approaches are also worth noting such as: -

- Long term support planning
- Dealing with this risk on a case-by-case basis
- Preservation of the development and modification environment
- ESCROW-minimising changes to the systems and then performing technology refresh
- Maintenance of obsolescence register

- Tracking and monitoring software obsolescence proactively
- Locking down the system so that there is no technology insertion.
- Spiral development to plan regular software replacement.

Passing the risk to the OEM is also the most popular mitigation strategy and this due to the increase in availability-based contracts followed by reactive and proactive strategies as identified in IEC: 62402.

Some of the above-mentioned software obsolescence mitigation strategies could be hugely expensive. The key cost drivers for these mitigation strategies includes total cost of ownership, legal cost, and support cost. Maintaining a team of highly qualified, skilled, and experienced personnel to provide support is extremely difficult and could be very costly. Testing and requalification of software, especially for custom built or safety critical software carries a huge cost burden.

Sustainment of required skills base in the workforce and sustainment of infrastructure are the key cost drivers for proactively managing software obsolescence. Planning for through life changes and designing an open system upfront might help reducing this cost. Taking time to review, plan and monitor design, development and testing of software will help to reduce the chances of software getting obsolete.

System knowledge and the availability of current skills are essential to manage obsolescence and to proactively manage these risks. These skills will have to be brought in to resolve or assess the risks if not maintained in-house.

Summary of the online survey findings are summarised below.

1. There is lack of dedicated resources to deal with software obsolescence.
2. There is minimum experience within the Defence and aerospace industry in dealing with software obsolescence

3. About 56% said they have obsolescence management strategy but only 25% said they follow any software obsolescence management standards
4. There is clear evidence that reactive approach is the most applied software obsolescence mitigation strategy and majority of them plan for software obsolescence at in-service phase of the project life cycle.
5. There is no standard, process, or mechanism in place to monitor software obsolescence.
6. There is inconsistency in defining software obsolescence.
7. Software obsolescence can trigger hardware obsolescence and vice-versa
8. There is Inconsistency in what needs to be in software obsolescence mitigation plan.
9. There is no evidence to suggest that project team proactively engage to identify software obsolescence.
10. There is no evidence to suggest that the project team undertakes estimating activities to estimate the cost of software obsolescence.

4.3 Summary of Observation

This chapter has presented the current practices in the UK Defence sector for managing software obsolescence and estimating the cost of software obsolescence resolutions. Defence and aerospace are focusing their effort on dealing with component obsolescence, whilst disregarding software obsolescence. This lack of understanding and knowledge in software obsolescence is a key issue identified in this chapter. There is substantial lack of resources including both capacity and capability to manage, plan and mitigate software obsolescence issues let alone the tools to monitor software obsolescence. This is in addition to the lack of availability of historic data to model and predict the cost of software obsolescence resolutions.

This chapter presented the various methods used to capture the current practice in the Defence and Aerospace industry. The industry engagement and SME interviews were carried out with the support of industry partners and the academia to gain an overall understanding on software obsolescence issue. This was then followed by a detailed case study of the Defence projects where the researcher interviewed key personnel from all the six projects that was down selected due to the nature of the project.

The data was analysed, and key findings were identified, however this needed to be validated within the wider industry for generalisability of the findings. The existing research related to obsolescence is largely on hardware and electronic components, with very limited, or no research undertaken on software obsolescence recently. Few researchers such as Sandborn, (2007), Merola (2006) started looking into software obsolescence largely around the software components, and this clearly defines the need for further research in this area. IOM (International Institute of Obsolescence Management) recognised the need for this, and they have published an overview guide on how to manage software obsolescence and further research undertaken by Fran (2009) provided the basic building block for this research.

There is a growing tendency for the risk of software obsolescence to be passed on to the contractors or pushed to the supply chain to manage and mitigate software obsolescence. This is due to availability-based contracting where the risk of software obsolescence is placed on the supplier to ensure the capability is maintained. There is also a lack of capacity and capability to undertake these risks and to manage this in-house, which historically has been difficult to quantify due to lack of historic data.

Passing the risk to the OEM is also the most popular mitigation strategy and this due to the increase in availability-based contracts followed by reactive and proactive strategies as identified in IEC: 62402.

Whilst undertaking the interviews for the case studies, it was soon noted that software obsolescence can be predicted at the development stages of a project life cycle, however, this theory/hypothesis needed to be tested. To test this

theory, in addition to identifying the root cause of software obsolescence and to predict, manage and mitigate software obsolescence at a very early stage, a cognitive case study was undertaken.

The next chapter presents the process, research methodology, experiments, and findings from a Cognitive Case study to determine the root cause of software obsolescence and to identify the cognitive behaviour of the software developer to understand the factors that determines/triggers the obsolescence issue while developing the software.

5 Cognitive Case Study

5.1 Introductions

Software Obsolescence is a major issue for all the custom-built Defence and aerospace software as discussed in Chapters 2, 3 and 4. Software technology is evolving every day and active technological insertion is inevitable. These further lead to constant changes in the requirements for both new and existing software. Due to these reasons, often it is very difficult to predict when and which component of the software will go obsolete and therefore makes it difficult to proactively manage them. There has been very limited research on the mitigation of software obsolescence to date. Defence and Aerospace industries largely investigate component obsolescence and when it comes to software obsolescence issues, they get classed as systems obsolescence (Rajagopal, Erkoyuncu and Roy, 2014). whilst there has been good progress on managing and formulating component/ hardware obsolescence strategy; very little or no effort has been given in managing and formulating software obsolescence management strategy. It is due to these factors listed below (from literature review, case study, SME interviews and online survey) that software obsolescence is treated/managed reactively.

- There is limited or no research on identification, management, and mitigation of software obsolescence
- Defence and aerospace products are unique and bespoke
- Considerable amount of the Defence software products are real-time software, which is required and need to be functional for the whole life of the capability.
- Immense consideration is given to component/platform or capability obsolescence in Defence, due to existing large and more often complex supply chain.
- There is no mandate to follow Obsolescence management standard in UK MoD [JSP 886 Chapter 7]
- There is no standard for Software Obsolescence Management [JSP 886, Chapter 7 and BS 62402]

To model the approach in order to identify the potential software obsolescence issue during the development of the software, the researcher undertakes cognitive case study with software developers from Defence and aerospace industries. This approach was undertaken to understand the reasons, be that external or internal, for selection of various software technologies by the developers. To test this, the researcher developed an experiment by developing two in-house software and tasked each participant to merge this two software together. A detailed user requirement document was developed for the purpose of this experiment. A detailed analysis of the cognitive behaviour was undertaken to understand the reasons for each participant to select the specific technology and its potential impact on the future obsolescence issue. As a result of this analysis, a model was developed, which a software developer can use to understand the impact of current technology selection on the future obsolescence issue, and therefore helping the project manager to proactively manage, mitigate and resolve potential future obsolescence issues saving considerable amount of cost for both customers and suppliers. This cognitive case study is unique as there have been no previous study conducted on the cognitive actions of Software Developers nor have been a relation established between the Software designers to the source of software obsolescence problems.

Majority of software obsolescence research undertaken as described in Chapter 2 is around the mitigation issues and not on where and when the software obsolescence initiated or to identify what triggers obsolescence problems. A Cognitive case study has not been used previously to understand the reasoning for the technological selection by the software developers to mitigate software obsolescence. Following the analysis of the interviews that were conducted on software obsolescence as part of the initial research as described in Chapter 4, the researcher developed a hypothesis that the software developers do tend to perceive when a software will get obsolete. One way to test this hypothesis was to undertake a cognitive case study where the software developers are put to test.

Software Obsolescence is one of the key issues facing Defence and aerospace industries due to the long product life cycle and the need to sustain the capability for much longer than a normal product life cycle. To identify and predict where and what triggers software obsolescence, it is vital that a reverse engineering principle is applied to software development. Case studies to identify software obsolescence made it clear and evident that software developers do identify obsolescence risk related to the technology selections. To prove this hypothesis the researcher decided to apply a cognitive research methodology to identify if the developers identify/ predict the software obsolescence issue during software development stages.

The researcher wanted to identify the process following developers receiving the requirements and design documentation provided as part of this experiment. The plan was to identify the decision-making process related to technology selection during the software development, that will influence the software obsolescence in the future. From the literature reviews as described in Chapter 2, and from case studies and expert interviews as described in Chapter 4, the researcher developed the following key research indicators that will determine the decision-making process by software developers: -

- Standards that they would use and why?
- What is the key decision made through the process?
- Are the company policies a key driver?
- Are they thinking about software obsolescence during software development?
- What influences their decision-making process and how does that map across to the software obsolescence risk?
- Are there any external or internal factors affecting their decision and how does that map across to the software obsolescence risk?
- Testing requirements and standards that they follow and how does that map across to the software obsolescence risk?

These questions formed the basis of the question set that were asked during the experimentation.

5.2 Research Methodology

This study methodology employs the “think aloud” technique to capture the cognitive actions of software developers. This “think aloud” technique has been used since the 1930’s as a viable method to study the thought process of participants and to enable the modelling of their actions and behaviour. This method was only formalised in 1993 by Ericsson and Simon which requires the participant to literally say aloud everything that he/she thinks or does during a controlled experiment. Everything that is said will be recorded (video and audio), transcribed and then described as a “verbal protocol”. Anything that is written down by the participant during the experiment is collected and analysed as a “written protocol”. Once the experiments have been carried out and the various protocols recorded and collected, they are then analysed. A detailed and specific procedure will be implemented to analyse the results.

This research has involved the use of five software developers with various levels of experiences from across the Defence and aerospace industries in United Kingdom. The first was a pilot case undertaken to check the proof of concept and the remaining four were the actual tests undertaken by the experts and practitioners from the Defence and aerospace industry having more than fifty years of total experience. Cross, 2004; suggested that the experts need to achieve a minimum period of proactive and sustained involvement that is required before an individual’s credibility is compared with international peer, and as such this is a reasonable sized sample for a cognitive analysis study which provided a varied sample. A request for participation was sent out to UK Defence and aerospace industry and five participants were down selected based on their understanding of Software Obsolescence issues and risks. Table 5-1 gives the breakdown of the participants and their experience.

Table 5-1 Details of the cognitive case study participants

Participants	Company	Level of Experience	Years of Experience
Participant 1	Company A	Novice	2
Participant 2	Company B	Expert	15
Participants 3	Company C	Expert	20
Participants 4	Company D	Expert	8
Participants 5	Company E	Practitioner	5

5.3 The Cognitive Case Study Experiment

The aim of the cognitive case study is to understand how the expert reason about Software Obsolescence during software design and development. The design of the experiment followed the academic and scientific approach. The approach is stated in the Figure 5-1 below.

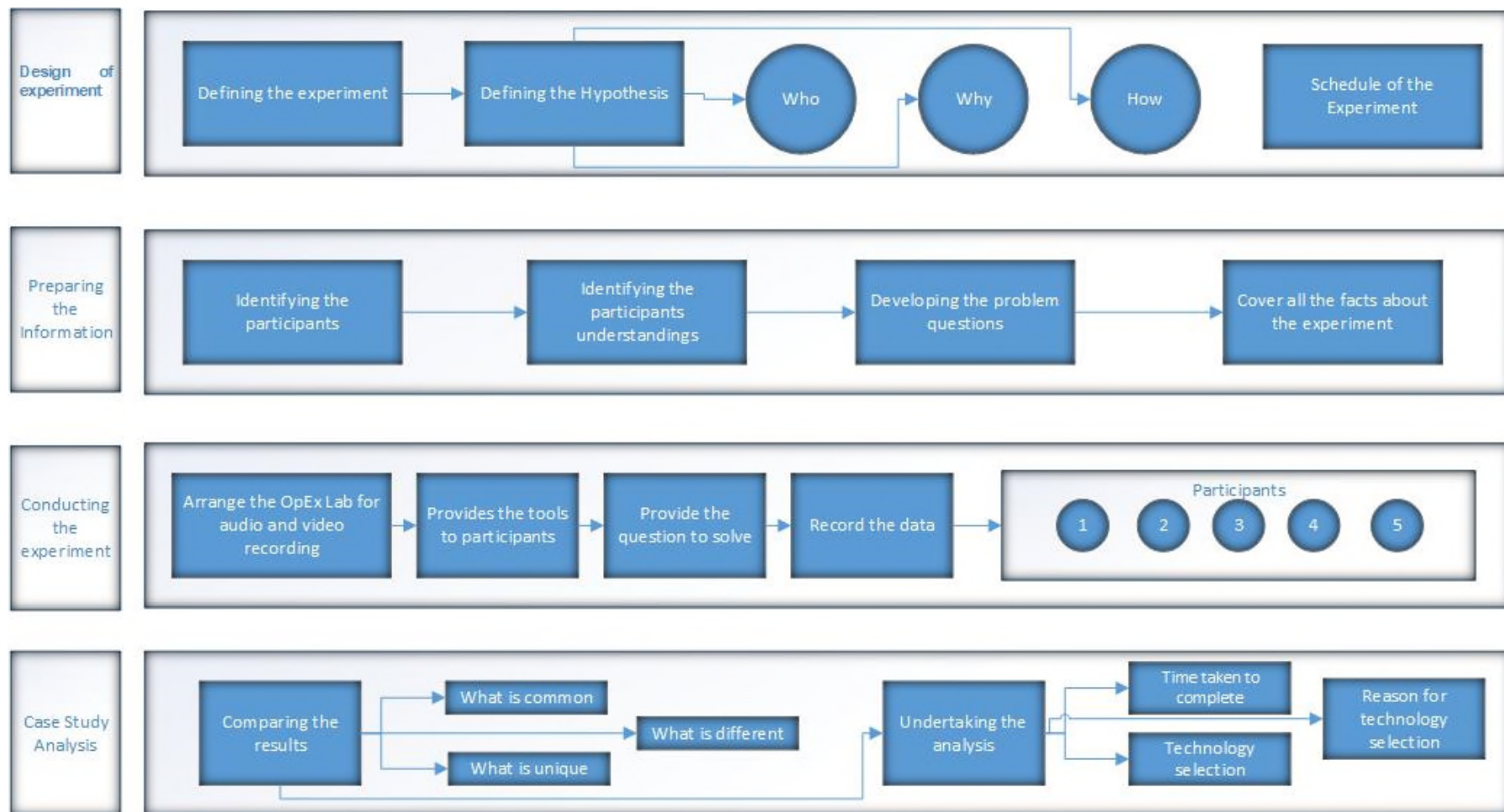


Figure 5-1 Cognitive case study approach

5.3.1 Designing the experiments

The first step of the experiment was to design and define the experiment. This involved understanding the process for the case study that will be conducted in addition to the questions that needs to be tested etc. From the analysis of the expert interview, it was evident that the software developers were able to identify the potential obsolescence risk in the software that they were developing. This, therefore become the exam question for this experiment. A hypothesis was developed to prove these findings and following steps were involved in the designing of the experiments.

5.3.2 Defining the experiment

To prove the hypothesis, researcher developed two sets of software one of which was developed using an open-source software and the other using a latest technology. The experiment question was for the software developers to integrate both the software. The key reason for this approach being the fact that open-source software is not considered for use due to the restrictive nature of the department that the researcher was working and therefore is considered obsolete.

5.3.3 Defining the hypothesis

The hypothesis was developed following the interview analysis which noted that the software developers were able to predict when the software that they are working on would go obsolete. This needed to be proved as this would give an early indication of the root cause of software obsolescence at a very early stage and therefore assisting the proactive management of potential obsolescence issue.

Once the hypothesis was confirmed, the researcher addressed the next step to decide the participants list for the experiment. Since the target audience and the key sponsor of the research was the Defence and aerospace sector, researcher invited software developers from the same industry. The experts were invited to Cranfield University's OpEx Lab to undertake the experiment.

5.3.4 Presenting the information and conducting the experiments

The invite was sent out to Defence and Aerospace industry in the UK through means of LinkedIn, emails, SCAF and ICEAA workshops. Five of the seven invitee who showed interest were down selected. The participants were down selected as shown in Table 5-1 based on their experience, availability, and their understanding of software obsolescence. Literature and information on software obsolescence and the experiment was sent to the participants well in advance. Any subsequent questions were answered via WebEx, skype and/or through by emails.

Participants were invited to Cranfield University audio and video tech lab, where the video recording was prepared and kept ready. The experts were given access to the experiment software and the relevant tools required for the completion of the experiment.

To complete this experiment each participant was given the following: -

- Software Development Database (Front-end and the database developed in LibreBase)
- Software Support and Maintenance Database (Front-end as web application in JavaScript with Tomcat Server connecting to MySQL Database)
- User Requirement Document (URD)
- Compilers such as Eclipse, Dreamweaver
- Software such as LibreBase, MySQL and Tomcat Server
- JDBC Database Connectors

5.3.5 Case Study Analysis

Once the cognitive case study was completed, the researcher compared their results by understanding the common trends exhibited by the experts, looking at any unique responses and also by analysing what the experts did differently. Analysis was also undertaken to understand the reasoning behind the technology selection and its potential to go obsolete.

5.4 The Experiment

The task for the experts was to integrate the two databases that was developed for the purpose of these experiments together to a single software programme. One of these databases was written using an open-source software and the other one was using MySQL. The open-source software is an obsolete software with reference to the MoD, given the restrictive nature of the MoD business. A detailed URD was produced to enable the participant to complete this experiment. Two types of URD were created – Non-Functional Requirement and Functional Requirement which are listed in Table 5-2 and Table 5-3

Table 5-2 Non-functional requirement

ID	REQUIREMENT	IN ORDER TO	OUTPUT DESCRIPTION
1	The SDSD (Software development and support database) application will be provided with technical documentation which describes the system architecture as well as the functional operation of the model	Allow the user to understand how the tool derives the various outputs and other reports	The technical document must be provided in a similar style and format to the CAAS technical documentation
2	The SDSD application will be provided with a user guide in a format agreed by CAAS and DE&S	Allow the user to understand how to use the tool (Database)	The user guide must be provided in both a hard copy and a web-based format compatible with DII and be similar in style and format to the current CAAS user guide.
3	Suitable training will be provided to the user groups identified by CAAS	Train the user group to use the tool to support Software Cost Estimation	A training needs analysis will be undertaken that will inform the development of training material and training delivery programme

4	The application will be hosted in DII(F) Restricted and available through normal DII(F) accounts	Allow ease of access through normal business support infrastructure	The SDSD application will be operable on DII (F) and available through normal DII (F) accounts. Users will only have the level of access directed by the administrator, required by their function.
----------	--	---	---

Table 5-3 Functional Requirements

ID	REQUIREMNT	IN ORDER TO	OUTPUT DISCRIPTION
1	SDSD Application is required to support the current level of usage	Ensure efficient delivery of robust Software Whole life cost estimates	The SDSD Application should support a minimum of 10 users and a maximum of 20 users concurrently. The SDSD Application usage should not be less than 80% at any given time The availability of the SDSD should be at 99%
2	SDSD Application is required to follow MoD Standard protocol	Ensure efficient functionality of the application	The SDSD Application should follow MODAF or TOGAF architectural framework

3	SDSD Applications should function at minimum bandwidth possible	Ensure the efficient running of application during the peak network usage time	SDSD Application should be able to work with minimum of 512Mb bandwidth.
4	SDSD web application should be able to function in IE8 and should be forward and backward compatible	Ensure that all the CAAS offices can use this application	• SDSD Application should be compatible with IE8 • SDSD Application should be backward compatible so that satellite office where the tech refresh has not been completed should be able to use • SDSD Applications should be forward compatible to mitigate the future tech refresh • SDSD Application should be compatible with Google Chrome
5	SDSD Application should be able to update the existing records	Ensure the updated data is captured and used	• SDSD Application should have a button to update the existing records • SDSD Application should have the consistent format to update the records • SDSD Application should have mechanism to validate the data

6	SDSD Application should be able to add new records	Ensure the continuous currency of the application	• SDSD Application should have a button to add new record. • SDSD Application should have the consistent format to add new record • SDSD Application should have mechanism to validate the new record
7	SDSD Application should be able to submit records	Ensure the updated and new data are correctly captured and used	• SDSD Application should have a button to submit records • SDSD Applications should have the consistent format to submit the records • SDSD Application should have mechanism to validate the data
8	SDSD Application should be able to delete the records	Ensure the application is updated and current for users	• SDSD Application should have a button to delete records • SDSD Application should have a warning message before the record is deleted
9	SDSD Application should be able to save records	Ensure the updated and new data are correctly captured and used	• SDSD Application should have a button to save records

			• SDSD Application should have the consistent format to submit the records • SDSD Application should have mechanism to save the data
10	SDSD Application should be able to perform search records	Ensure the user can search the records	• SDSD Application should have a button to search the records • SDSD Application should be able to search using Organisation type, Platform type, Application type and year of the project
11	SDSD Application should be able to report the records	Ensure the user can use the application to produce the reposts	• SDSD Application should have a button to generate reports • SDSD Application should generate report on all the fields in the data tables.

For this exercise both the software programme were developed in house by the researcher. The two databases that was developed were

1. Software Development Data Sheet (this is the database that consist of software development data such as application types, languages etc) and
2. Software Support Data Sheet (this is the database that consist of software maintenance data such as hours spent on each

types of maintenance etc)

The Table 5-4 below summarises the technology selection, requirement, and reasoning for selecting this software.

Table 5-4 Summaries of technology selections used to develop the in-house software

SI No	Software	Technology Selection			Requirement	Reasoning
		Front end	Back End	Connector		
1	Software Development Data Sheet	LibreBase	LibreBase		Is to develop a software which would be seen as an obsolete product for MoD.	LibreBase is an open-source software and due to the restrictive nature of the MoD business, this software will not be able to be supported therefore making it obsolete.
2	Software Support Data Sheet	Web based- JavaScript	MySQL	Tomcat Server	Is to develop a software which would not been seen as an obsolete product	JavaScript and MySQL are supported by the MoD and there are various softwares that uses this technology and are supported. It is also assumed that this will be supported in the future until a new directive is in place.

Software Development Data base was developed with the help of an open-source software whereas the Software Support Data base was developed using MySQL which is the latest technology that is now available. The task was to integrate these two databases into one. The researcher chose to develop this software in two different ways to replicate the current scenario to make it as real as possible in the Defence and aerospace industries. This approach was taken to get a good understanding of the problems that prevail whilst developing the software which then could be mapped across to software obsolescence resolution approaches.

Software Development Database was developed using an open-source software called LibreBase. The researcher chose to use this, as the MoD would not be able to access this software due to the restrictive nature of MoD business and therefore making this software obsolete. This was a deliberate action to identify the problem(s) participants might face and to study the mitigation action they would deploy especially the technology selection process.

Figure 5-2 shows the front end of the software and Figure 5-3 shows the backend of the “Software Development Database” that the researcher developed.

Software Development Data Sheet

ID	1		
Year of Project	1998		
Industry Sector	Service Industry		
Organisation Type	Recreation & Personnel Services;		
Application Group	Business Application		
Application Type	Transaction/Production System;		
Development type	New Development		
Development Platform	MR		
Language Type	4GL	First Record	Last Record
Primary Programming Language	Oracle		Referesh
Count Approach	NESMA		Save
Functional Size	237	Next Record	Previous Record
			New Data

Record 1 of 41 *

Page 1 of 1 Default Style 100%

Figure 5-2 Software Development Database front-end

LibreOffice File Edit View Insert Tools Window Help										
Software Development Data - Software Development Database										
	ID	Year of Project	Industry Sector	Organisation Type	Application Group	Application Type	Development type	Development Platform	Language Type	Primary Programming
	1	1998	Service Industry	Recreation & Personnel Servi	Business Application	Transaction/Producti	New Development	MR	4GL	Oracle
	2	1996	Construction	Construction;	Business Application	Stock control & order	New Development	Multi	4GL	Access
	3	2002	Wholesale & Retail	Billing;	Business Application	Billing;	Enhancement		3GL	COBOL
	4	2004					Enhancement			
	5	2000	Wholesale & Retail	Wholesale & Retail Trade;	Business Application	Management Informa	Enhancement	MF	3GL	COBOL
	6	2000	Insurance	Insurance;	Business Application	Sales contact manag	New Development	MF	3GL	Java
	7	2004	Banking	Banking;	Business Application	Financial transaction	New Development	MF	3GL	COBOL
	8	2000	Banking	Banking;	Business Application	Financial transaction	New Development	Multi	4GL	Visual Basic
	10	1998	Communication	Telecommunication;	Business Application	Transaction/Producti	Enhancement		4GL	ADS/Online
	11	2002	Wholesale & Retail	Ordering;	Business Application	Stock control & order	Enhancement	MF	3GL	C
	12	2002	Professional Servic	Sales & Marketing;	Business Application	Trading;	Enhancement			ColdFusion
	13	1998	Manufacturing	Manufacturing;	Business Application	Management Informa	Enhancement	MR	4GL	Oracle
	14	2002	Communication	Voice Provisioning;		Voice Provisioning;	Enhancement		3GL	C
	15	2006	Service Industry	Government;Community Serv	Business Application	Customer billing/rela	New Development		3GL	C#
	16	2007	Insurance	Insurance;			Enhancement	MF	3GL	PL/I
	17	1999	Manufacturing	Manufacturing;	Business Application	Customer billing/rela	New Development	Multi	4GL	Visual Basic
	18	2006	Banking	Banking;	Business Application	Financial transaction	Enhancement	PC	3GL	Visual Basic
	19	2003	Financial	Financial, Property & Busines			Enhancement		3GL	COBOL
	20	1997					Enhancement			
	21	1994	Financial	Financial, Property & Busines	Business Application	Office Information Sy	New Development	PC	4GL	SQL
	22	2003	Insurance	Insurance;			New Development	MF	3GL	PL/I
	23	2004					Enhancement		4GL	PowerBuilder
	24	2010	Electronics & Comp	Computers & Software;			Enhancement	MR	4GL	ABAP
	25	2004	Financial	Financial, Property & Busines	Business Application	Financial transaction	Enhancement	Multi	4GL	
	26	2002	Insurance	Insurance;			New Development	Multi	ApG	COOL:Gen
	27	2005	Insurance	Insurance;			Enhancement	Multi	4GL	
	28	2004					Enhancement		4GL	Oracle
	29	1999			Business Application	Management Informa	Enhancement	MR		
	30	2010	Manufacturing	Manufacturing;	Business Application	Cars selling;	Enhancement	MR	3GL	Java
	31	1998	Communication	Telecommunication;	Business Application	Transaction/Producti	Enhancement		4GL	Visual Basic
	32	2004			Business Application	Financial application	New Development	Multi	4GL	
	33	1999			Business Application	Transaction/Producti	Enhancement	MF	3GL	COBOL
	34	1998	Government	Public Administration;Commu	Business Application	Electronic Data Inter	New Development	MF	3GL	COBOL
	35	2011	Manufacturing	Manufacturing;	Mathematically Intens	Computing of the the	Enhancement	MR	4GL	MATLAB
	36	1994	Banking	Banking;	Business Application	Transaction/Producti	New Development	MF	4GL	EASYTRIEVE
	37	1999	Insurance	Insurance;	Business Application	Financial transaction	New Development	MF	3GL	COBOL
	38	2000	Banking	Banking;	Business Application	Financial transaction	Enhancement	MF	3GL	PL/I
	39	1994	Manufacturing	Manufacturing;Transport & St	Business Application	Transaction/Producti	New Development	MF	ApG	TELON
	40	2004				MS Billing;	Enhancement		3GL	COBOL
	41	2005	Insurance	Insurance;			Enhancement	MF	3GL	PL/I
	42	2005	Manufacturing	Manufacturing;			Enhancement	Multi	3GL	C++
	43	2004			Business Application	Financial application	Enhancement	Multi	3GL	Java
Record	1	of 87 *								

Figure 5-3 Software Development Database back end

The Software Support and Maintenance database was developed using MySQL Workbench (Figure 5-4), please see Annex B for the MySQL Source Code used for the back end of the database. The front-end is a web-based application, (Figure 5-5) connected to the database using Tomcat server. Based on the interviews conducted with the experts the researcher presumed this as the latest technology in the market at the time of the experiment that would be in service and supported by the Defence and aerospace industries in the future until a new directive comes to effect.

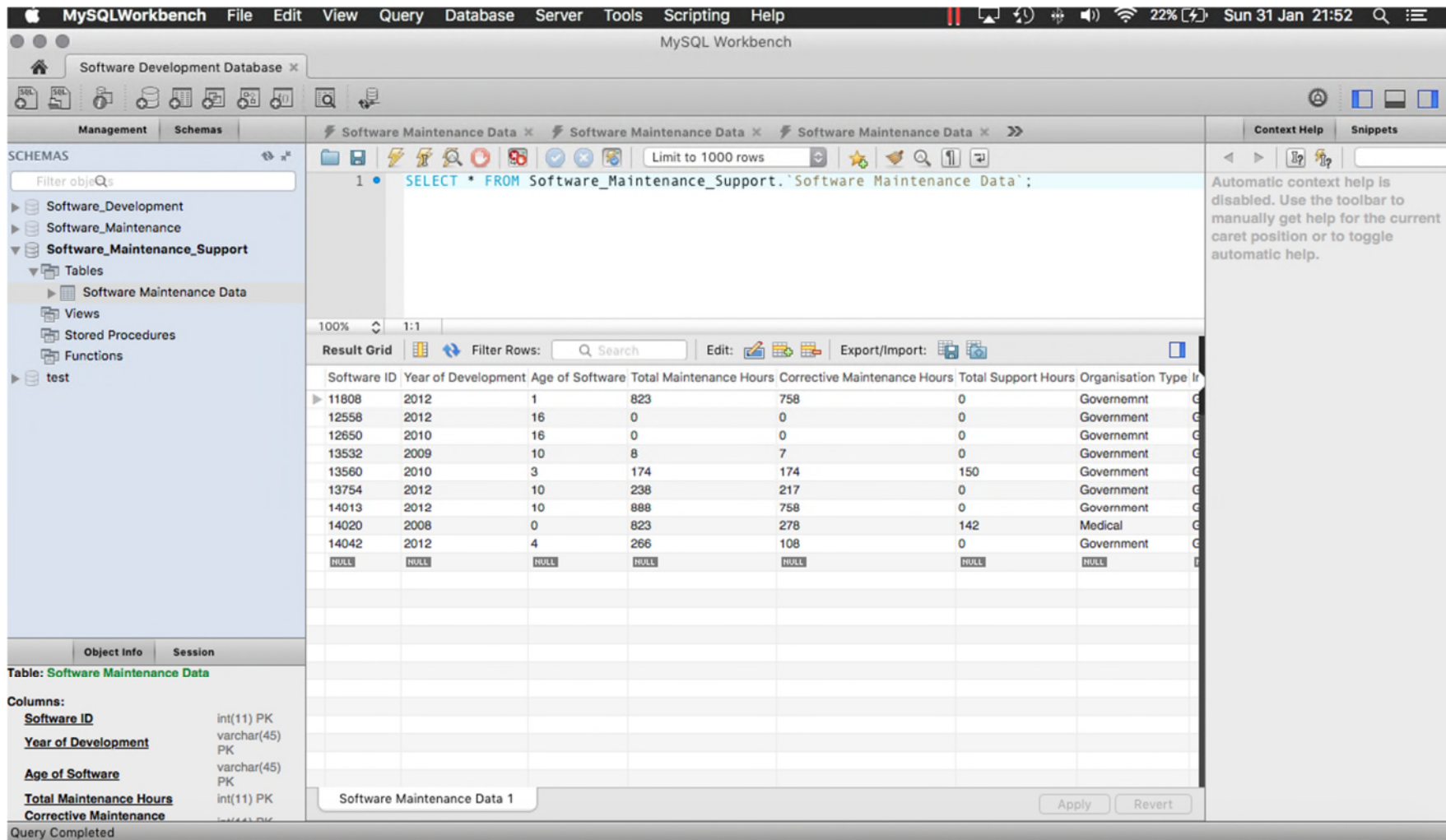


Figure 5-4 Software Support Database back end

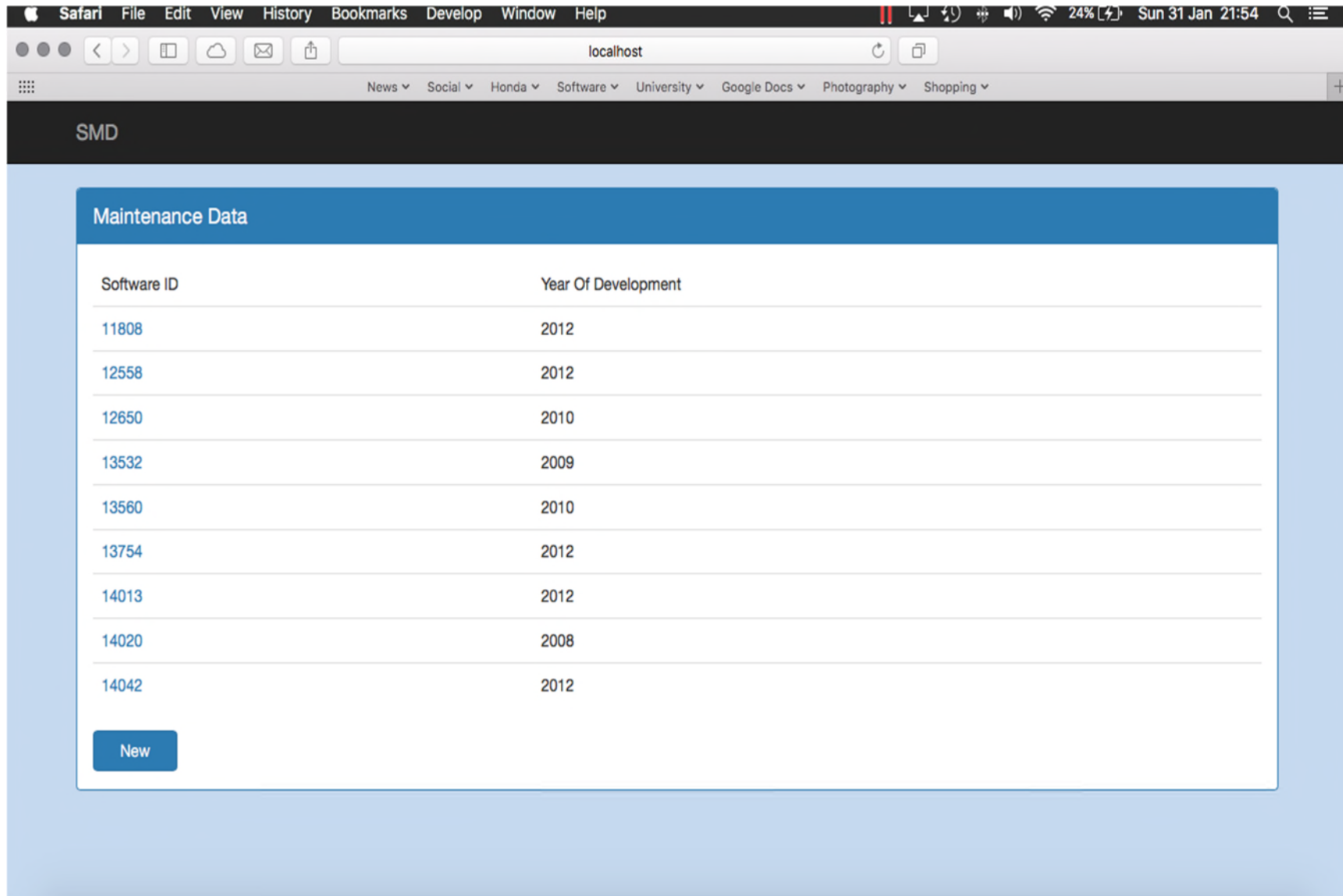


Figure 5-5 Software Support Database front-end

5.5 Cognitive Case Study Coding Scheme

Protocol analysis involves the application of a code to enable the structured analysis of the protocol to take place. This code represents the software developer's cognitive actions. Therefore, before the analysis of protocol could be carried out, a coding scheme was developed. The coding scheme represents how software developer(s) tackle the various components of the software development. The verbal and written protocol, which also includes the written code, will be analysed against Key Performance Indicators (KPI) and Key Performance Measures (KPM).

Historically, cognitive research splits these issues faced by the subject into two areas: how the subject tackles the “problem domain” and the “strategies” that are employed to solve the problem.

Through a review of the literature on cognitive actions of design engineers (Houseman, Coley and Roy, 2008), it is possible to identify several problem-solving techniques utilised by designers. These are frequently divided into two areas known as micro strategies and macro strategies. The researcher has added a third dimension called KPI (key performance indicators) with which the micro and macro strategies will be mapped to identify software obsolescence issues. This is done due to the nature of software design and development, which is considerably different compared to hardware design and development.

5.5.1 Micro Strategies

These micro strategies include the ability of software developers to conscientiously evaluate their design proposal prior to developing them, the use of analogy and the frequent consultation of external information. Software design and development problems are described as being unlike any other problem since the amount of knowledge in both the problem specification and the solving of the problem has no limitations. Due to this, software designers and developers have been observed to allocate substantial period to simply analysing the problem. To deal with a design and development problem software developers utilises several micro strategies to find a resolution more effectively. There is a

heavy reliance upon the individual micro strategies; individual skills and characteristics alter the individual's development process and ultimately the outcome.

Micro Strategy will be analysed against the following competencies (Houseman, Coley and Roy, 2008) -

- Analysing the problem
- Proposing a solution
- Analysing the solutions
- Validation of the solutions

5.5.2 Macro Strategies

Most of the software designers and developers share a common approach such as 'top down' approach for problem solving in addition to other approaches to complement the primary approach. This strategy is the reduction of the overall design problem into smaller issues to be addressed. These strategies provide a strong useful technique.

Macro Strategy will be analysed against the following competencies (Houseman, Coley and Roy, 2008)

- Compilers used
- Decomposing the problem
- The approaches
- Back tracking etc.

5.5.3 Key Performance Indicators

These are the indicators or a minimum standard that the software developers would need to meet with the help of micro and macro strategies. This is used to generate the final analysis and will be a key technique in the identification of origin of software obsolescence in software development phases.

These KPI's were mapped following the literature reviews, interviews and case studies that were undertaken as part of this research. This will become the key

analysis technique to determine the selection of the various technologies which could trigger software obsolescence.

- Low level designing
 - Defining the objects
 - Defining the use case
 - Defining the database and data flows
- Standards
 - Defining what standards to follow
 - Defining documentation and product manuals
- Development life cycle
 - Identifying the appropriate development life cycle
 - Waterfall
 - Agile
 - Test driven development
 - Continuous integration test driven development
 - Prototyping
 - Incremental
 - Iterative
 - Spiral
 - Defining the development approach
- System requirement
 - Defining requirement
 - Key drivers that effect development
 - Identification of internal and external factors
 - Contingency plan
- Validation (Testing)
 - Defining test requirement
 - Defining test cases
- Maintenance (Support)
 - Defining support team
 - Defining support structure
 - Defect rates and error code documentation

The analysis of KPI, KPM and Macro strategy will give the researcher an indication on what happens during a software development and the result can be then mapped across the reasons for software obsolescence and technology selection to mitigate software obsolescence issue.

5.6 Cognitive Case Study Analysis

The task was carried out in order to understand the rational on how the developer's reason about software obsolescence during the design and development and to identify how this could be mapped to SOCAF.

Each experiment had an individual objective to get the best results. Experiment one was a pilot to check if the experiment would work and to capture if the developer could identify all three variables such as Micro, Macro and the key performance indicators in addition to establish if the subject could identify any software obsolescence issues in this constrained environment. The second and third experiment was undertaken to identify a pattern or a model by which the software developers tackle the issue of Software Obsolescence (risk of software obsolescence.

Table 5-5). Experiment four was to validate the working pattern and to rectify any issues and re model; and the last experiment was for final validation and to firm up on the identification of software obsolescence and the technology selections made by the subject because of these issues. This experiment would also help the researcher in the identification of various other reasons such as political,

social, legal, and any other issues that could affect the technology selections to reduce the impact of software obsolescence or increase the risk of software obsolescence.

Table 5-5 Cognitive Case Study tasking

Activities	Key Deliverables
1 st Experiment	Pilot and to understand the Micro, Macro Strategy, KPI and to identify any obsolescence mitigation actions
2 nd and 3 rd Experiment	To understand the Micro, Macro Strategy, KPI and to identify any obsolescence mitigation actions. Also, to develop a working model
4 th Experiment	To validate the model and to re model
5 th Experiment	Final validation of the model.

The participants were given the same task, briefing and materials; however it is worth noting that each participant approached this experiment very differently.

5.6.1 The Analysis

The Technology Selection and Deployment Model

The Analysis investigated the pattern by which the software developer investigated the selection of the technologies while developing software with induced parameters. This was modelled following the second and third experiments. Researcher wanted to validate the consistency of the model and used fourth experiment to validate the approach and the final experiment to finalise the model. Below Figure 5-6 is the model that was used by each subject to make decisions on which technology (ies) to use whilst developing the software.

Each subject whilst making decision, passed through all three stages of the experiment namely micro and macro strategy; and the “looking into the problem” stages. Majority of the technology selection was undertaken at the micro strategy level with the final testing being done at the macro strategy level. The final product development happened while at the problem stages of the experiment. This proved that each subject does consider various options in both micro and macro strategy levels be that political, organisational, legal or availability of technologies before deploying the best possible solution to deal with the problem. The key finding was that each subject was able to deploy and select the required technology based on the external constraints, availability, and the requirement.

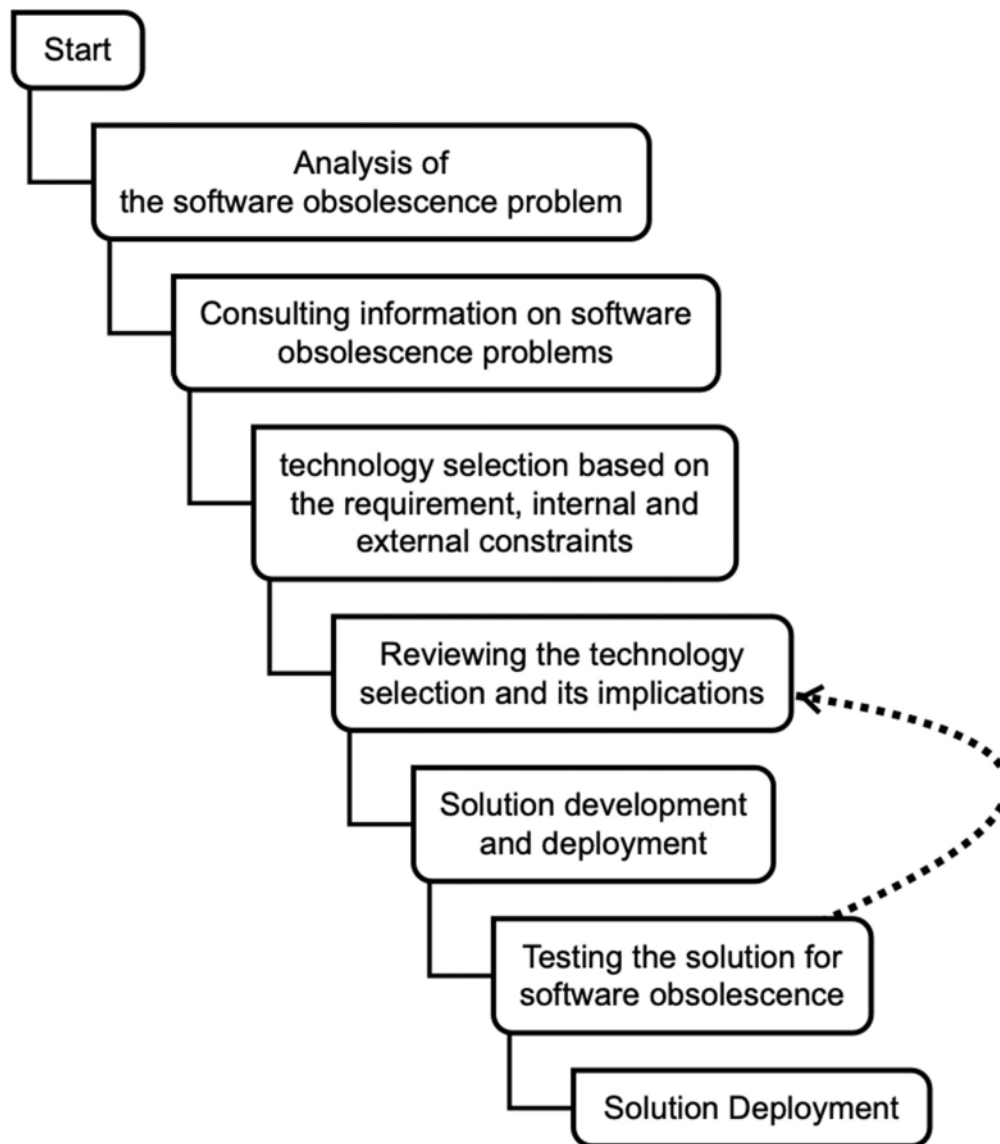


Figure 5-6 Technology selection and deployment

As discussed in Table 5-5, the first experiment was undertaken as pilot with the next two as actual experiments and the final two used to validate the experiments and the process that everyone undertook.

This model was further validated at Software IT CAST workshop in Washington DC where 15 software developers participated to validate this model. Three of the workshop attendees also advised that in a high demanding software development environment it is not always possible to think about software obsolescence unless it's specifically required. The workshop resulted in

validating the model and therefore it was decided no further experiment was required.

5.6.2 Common Approaches and Differences

A Coding scheme 9.4Appendix F) adopted from (Houseman, Coley and Roy, 2008) to undertake the final analysis was developed which combined all three key factors that are discussed above. Each participant had their unique way of dealing with the problem. However, each participant also exhibited a common approach in dealing with specific software obsolescence issue. The common approach was largely around the micro strategies and on how the software has been developed. The key commonality was in identifying the problem in relation to the software obsolescence. The analysis of the problem was different for all the participants, but the process of analysing was very similar. It was the same for solution identification and the validation of the solutions. The Figure 5-7 below explains the commonality and the differences in approaches for Macro and Micro strategies undertaken by the participants. For instance, four out five participants had a similar approach for the validation of the solutions, analysis of the problem and each participant approach to the problems. However, all the participants had a different approach in back tracking the problem and the types of compilers used.

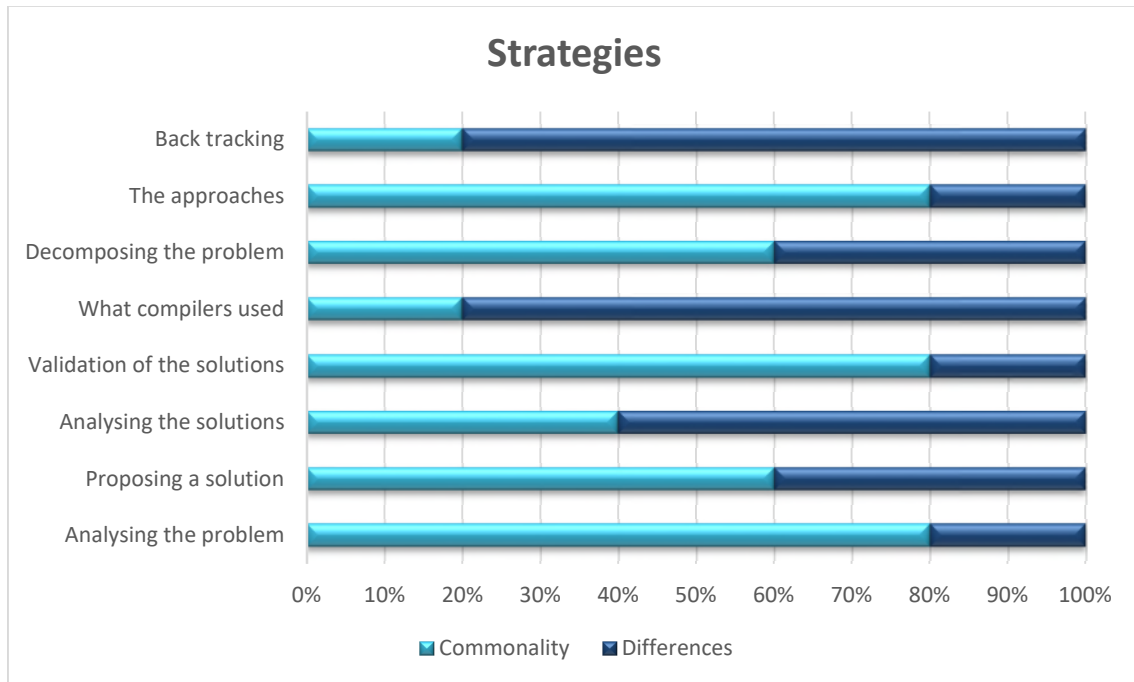


Figure 5-7 Common approaches and differences in micro and macro strategies adopted by participants

The other element of commonality was identified in the way the software was developed. The commonality was observed in the process by which each KPI was addressed, however the actual methodology selected after the process identification was very different. Below Figure 5-8 shows the commonality and the differences in the KPI's. For instance, four of the five participants understood the system requirement. The fifth person although did understand the requirement, however the interpretation was very different to others. Three of the five used different development standards as all of them followed their own organisational standard. The fifth person did not follow any standards as the participants was novice and did not work for any organisation. Three of the five participants used waterfall as their software development life cycle, one used incremental and the fifth person used agile development life cycle. However, there was no commonality on how they validated their development. The key part of this analysis was how they planned their supportability. Three of them followed the same principle as this was directly related to the development life cycle. The approach followed by other two were ad hoc and reactive.

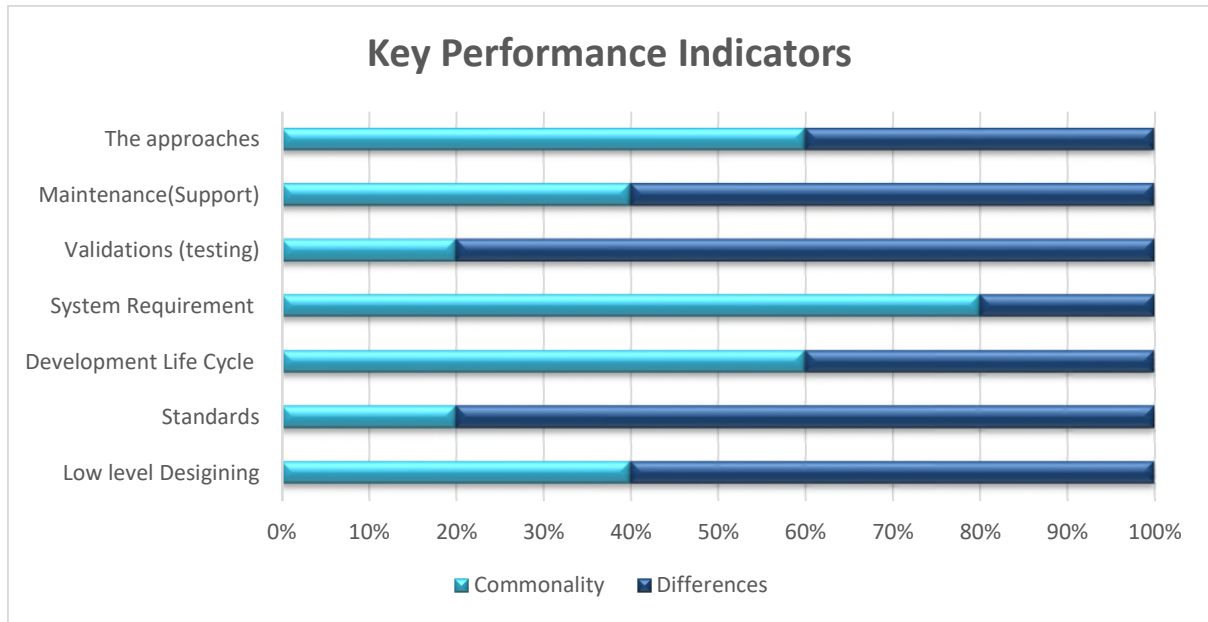


Figure 5-8 Common approaches and differences in the key performance indicators by the participants.

This analysis was important to identify the internal and external factors that affect the technology selection in the software development process to mitigate the software obsolescence.

It was evident that each participant considered options that were best suited, based on what was available for them at the time of the experiment such as their organisational standards, process, procedures, legal requirement and the availability of tools and technology. The technology selection at this stage would help in identifying potential obsolescence issue of the software.

5.6.3 Detailed Analysis

Each participant spent an average of four hours completing the task and went through different phases during the experiment not necessarily in the same order. The whole experiment was structured into three phases. Phase one was all about understanding the problem, proposing solutions, analysing the solutions and validation of the solutions. The second phase was about the approach that they took for decomposing the problem, technology selection and whether they could back track their efforts; and the third and the final phase was about how they

could develop the software around the design, methodology used, understanding requirement, validation, and supportability of the solutions.

On an average, each participant spent 5% to 10% of their time analysing the software obsolescence problem. Participants one, two and four took up to 10% of their total time compared to the rest of them who took about 5% of the total time to analyse the software obsolescence problem. All the participants did spend considerable amount of time looking and consulting information on the software obsolescence problems, so much so that participants four and five spent about 60% of time doing so. This is critical as this shows that software obsolescence is not necessarily considered during the development phases directly; however upon faced with the problem, they tend to consult on the available information to mitigate these issues. Same cannot be said of participant four who took only 30% of his time in evaluating the software obsolescence problem compared to participant one who took around 50% of his time evaluating these issues. Figure 5-9 below shows the graphical representation of the above analysis.

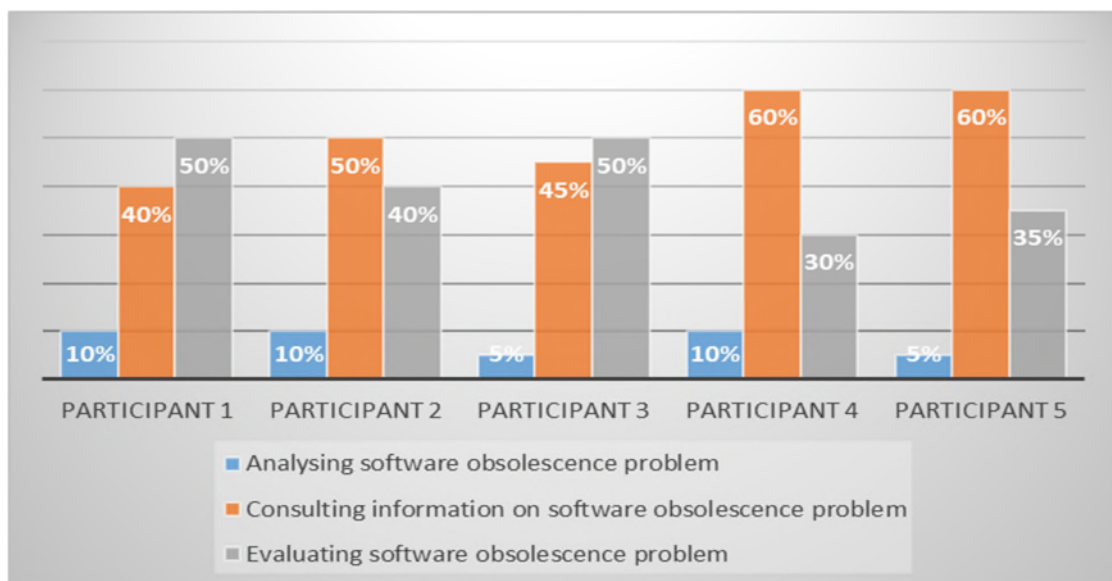


Figure 5-9 Time spent on solving the problem (software obsolescence which was pre-set for this experiment) by each participant and its breakdown.

Almost all the participants spent considerable amount of time referring to the type of technology that could be used showcasing the knowledge that they have in the selection of the technologies to avoid any future obsolescence issue. Same can

be observed in the case of down selecting the technologies or referring to the cause of software obsolescence whilst down selecting the technologies. However, all the delegates spent very little time in understanding or scoping software obsolescence strategy or explaining the effect of technology selection to the obsolescence strategy as shown in Figure 5-10

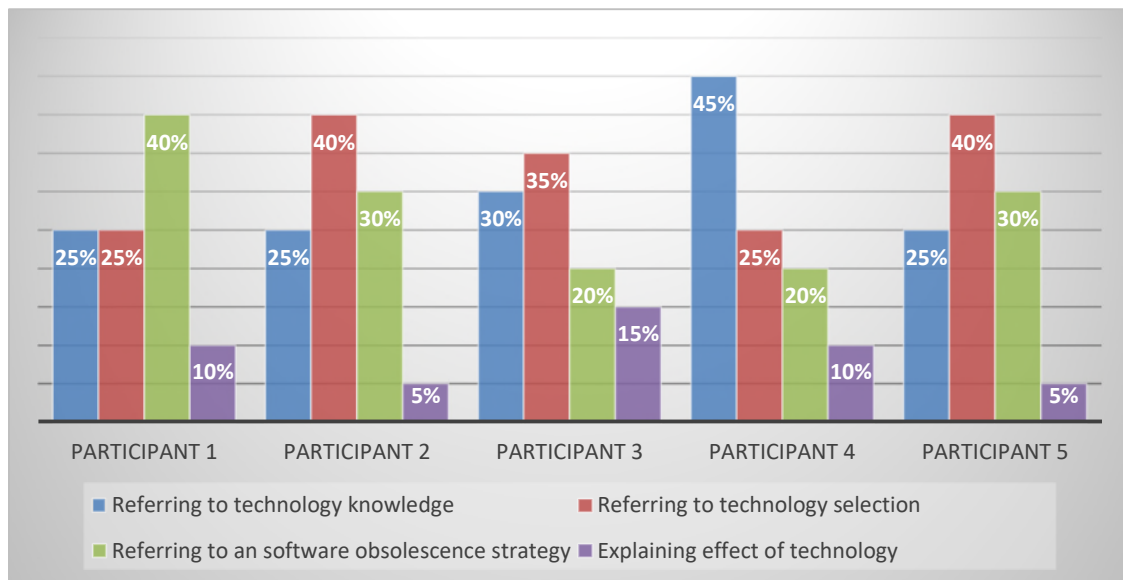


Figure 5-10 Time spent on selecting, refereeing to technology selection, and referring to software obsolescence by each participant

All the participants did spend considerable amount of time in top-down selection of the technologies, however very little time was spent on the confirmation and re-evaluation the technology selections as shown in Figure 5-11. This is a key finding as this shows less time was spent on assessing the impact of the current technology selection on the potential future issues that could arise. This is a key finding as this clearly shows that software designers need to have an early engagement to mitigate future obsolescence issues.

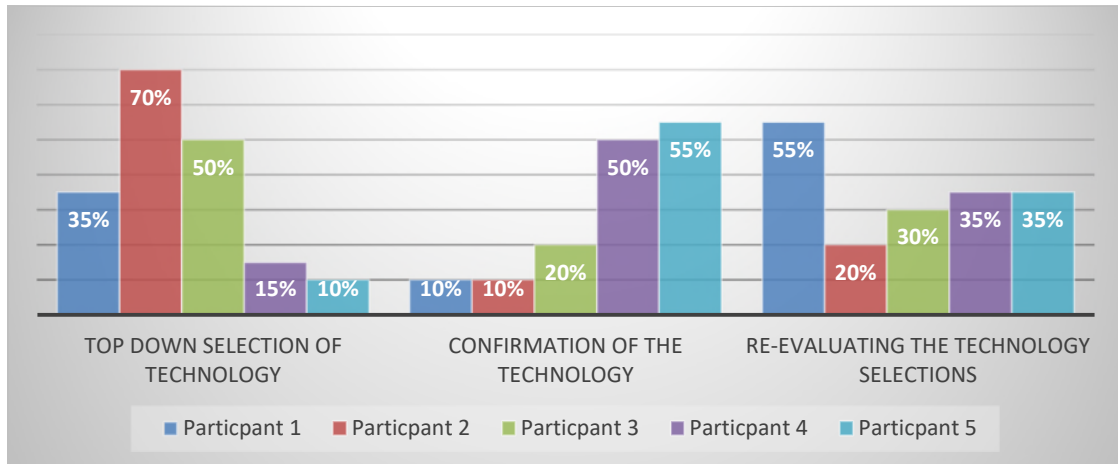


Figure 5-11 Technology selection and confirmation by each participant.

In summary, participants were able to down select the required technology based on the requirement and was able to demonstrate the relationship of down selected technology with the possible software obsolescence in the future, however they spent hardly any time in developing a software obsolescence strategy or any mitigating strategy.

5.7 Summary of Observation

It is important as to understand the cognitive behaviour when dealing with large complex problem, in this instant, to develop a software programme. However, this test concluded that cognitive behaviour of programmers was highly motivated and dictated by the available technologies, internal policies, and legal requirement. It was also evident that participants selected the technologies based on the future availability of the support and maintenance requirement.

Following extracts from the participant's audio/video recording re-enforces this theory.

Participant 1 *"I will use this technology so that it is readily available for the maintainers to use it in future"*

Participant 2 *"I am using this approach (Technology selection) in order to reduce the obsolescence as this technology is independent of changes in hardware"*

Participant 3 *"I am using this technology because I am certain that in next 10 years there will not be a change in the hardware or system in MoD that will make this technology selection an obsolete one"*

This cognitive case study which was focused on the defence and aerospace industry was undertaken to understand whether software developers whilst developing the software know when the software will go obsolete. It is evident from the participants that they do know what kind of technologies that they need to select based on several internal and external factors that will have an influence in the potential software obsolescence issues that could happen in the future. It is evident that there is a direct link between the technological selection by the software developers and the resolution approach that should be taken to mitigate the software obsolescence.

One of the key findings of this cognitive case study was that the researcher was able to establish that the future software obsolescence issues related to the software build could be predicted at early stages of the development. This can be achieved by analysing what technology the developers used, the current internal and external factors such as process, standards, and legal requirements. This finding is reflected in the technology resolution approaches that are described in the next chapter.

The findings from the Chapter 4 helped in the development of the software obsolescence management level and the software obsolescence complexity matrix which formed various sections of the software obsolescence cost analysis framework which is described in detail in the next chapter. The findings from the cognitive case study were used in the development of software obsolescence resolution approach and the key cost drivers which is described in detail in the following chapter.

6 Software Obsolescence Cost Analysis Framework

6.1 Introduction

The literature review in chapter two, has confirmed the lack of understanding of software obsolescence and estimating the cost of software obsolescence resolution for support contracts. Moreover, the capture of current practice in the UK Defence and Aerospace for contracting, as explained in Chapter four and five, has revealed that the responsibility for managing, planning and resolution of software obsolescence issues are being passed out to the prime contractor or OEM and industry collaborations. This has created a new requirement in terms of predicting the cost of software obsolescence resolution at early stages of the project life cycle, so as to include its cost within support contracts. In this chapter, the researcher presents the development of “Software Obsolescence Resolution Cost Analysis Framework” to be used at the bidding stage in order to estimate the software obsolescence resolutions cost.

Literature reviews and expert interviews have laid out the key challenges of estimating the software obsolescence resolution cost as lack of any defined resolution approach along with inadequate historic or current data. Moreover, the approach taken to tackle a particular software obsolescence issue cannot be specified in advance. Due to these reasons, a “software obsolescence resolution approach study” was undertaken in order to determine the probability of using a resolution approach to tackle software obsolescence issue. This is based on the experience of industrial experts on software obsolescence. The key element in the estimation process is a good understanding of software obsolescence cost drivers, as explained in Chapter 2 and Chapter 4, given the lack of knowledge and data around software cost drivers a “Software Obsolescence Key cost driver” study was undertaken, where key cost elements of software obsolescence resolutions were developed and validated.

In this Chapter, the research methodology followed the development of this framework, together with the specific methodology used in the development of “Software Obsolescence Management Level”, “Software Obsolescence

Resolution Approach”, “Software Obsolescence Complexity Level” and Software Obsolescence Key Cost Drivers” studies are detailed. Subsequently, the result of these studies and a description of the structure, methodology and rationale that the “Software Obsolescence Resolution Cost Analysis Framework” followed are presented. Finally, the iterative process followed to develop the framework is also described.

The final part of this chapter also looks at the difference, similarities and the relationship of software obsolescence with software maintenance.

6.2 Research Methodology for the initial study and assumptions

Chapter 3 covers the research methodology followed to develop the framework. Figure 6-1 below shows the steps undertaken to develop this framework which is further discussed in detail later in this chapter. This is adopted from (Javier Romero Rojo, 2011).

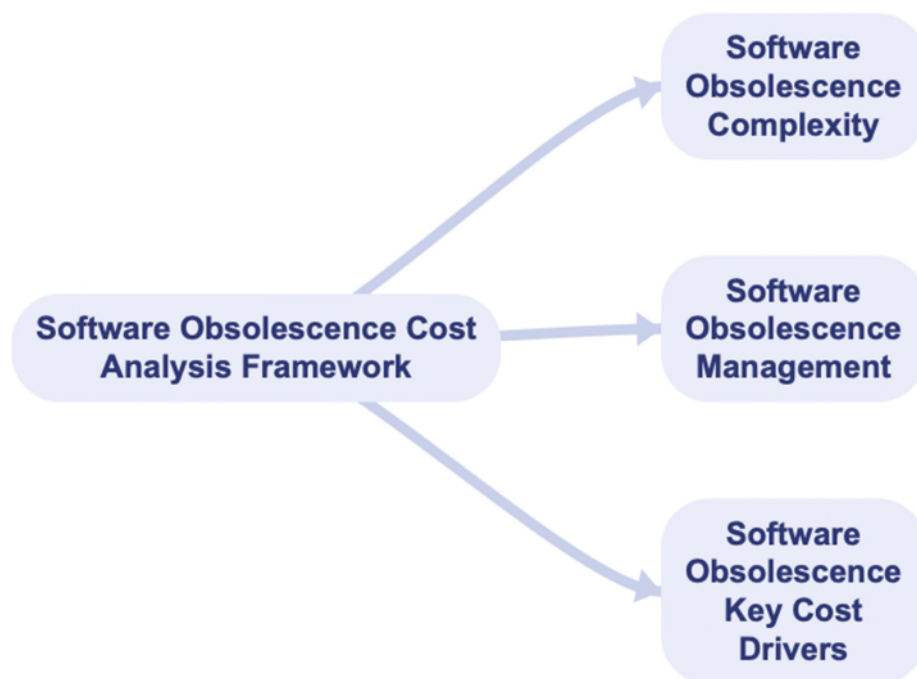


Figure 6-1 steps/study undertaken to develop the Software obsolescence adopted from (Javier Romero Rojo, 2007)

The first step was the development of “Software Obsolescence Complexity Model” to identify and understand the complexity of software obsolescence issue. This is done by identifying both external and internal factors that are related to a software, this is detailed further down in this chapter; given how critical it was to get the correct definitions on Software Obsolescence complexity. The Software Complexity is certainly different to Software Obsolescence Complexity as the initial one is defined as the time required for data to travel from point A to point B and the latter is more dependent on the target and operating environment.

The questions within the case studies, online surveys, and expert interviews were set to define this concept. This methodology allowed the researcher to capture the logic and key parameters that define the complexity of software obsolescence. The outcomes of this session were refined and validated at various IIOM quarterly meetings and at their International Workshops where more than 50 obsolescence experts were in attendance. This was further refined and validated at the ICEAA International Workshops and CAST IT Workshops with a total attendance of more than 120 cost and through-life experts.

The second step was to understand the need for identification and management of Software Obsolescence issues. The software obsolescence management strategies were identified according to the level of pro-activeness deployed by the experts. This classification was based on the information gathered through semi-structured interviews, online surveys, case studies and an exhaustive literature review. Initial findings of how software obsolescence is being managed by experts within industries is shown in Figure 6-2 below.

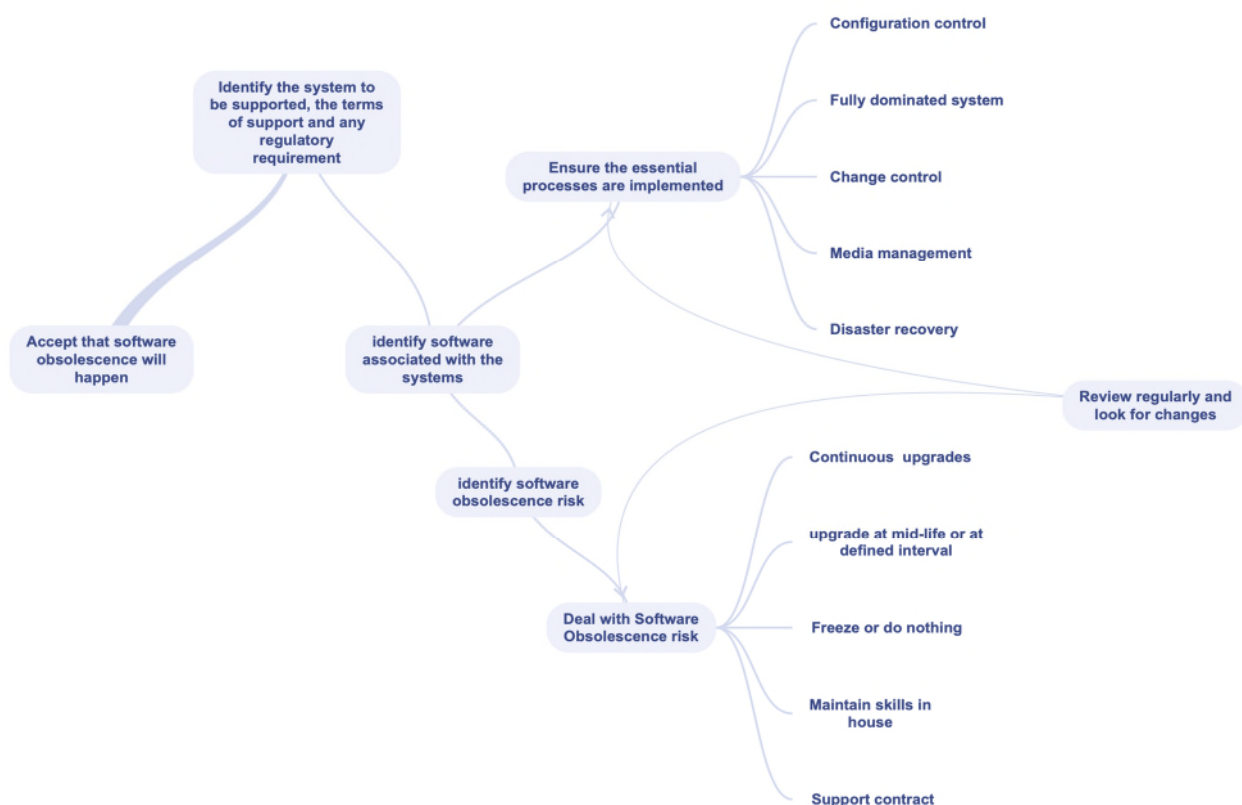


Figure 6-2 How software obsolescence is managed currently in the industry

In the same manner as the complexity matrix, this classification was presented at several IIOM quarterly meetings and international conferences along with ICEAA and IT CAST Conferences with an attendance of more than 120 obsolescence, cost and through life experts. The validation and verification of these are described and explained in detail in Chapter 8.

The third and final step is to gain the understanding of key cost drivers of software obsolescence resolutions. Since the researcher was able to establish a difference between software mitigation and software resolution, identifying the key cost drivers became clearer. The questions within case studies, online survey and expert interviews focused largely on identification of key or potential resolution approaches that are currently undertaken and how they are quantified. The initial findings were validated by the MoD (UK) Obsolescence Manager (15 years of experience). Figure 6-3 below shows the initial findings of key cost drivers which excludes the actual resolution approach which has been identified and explained further in this chapter. The key cost drivers shown in the figure below are the

external and internal factors that play an important factor in understanding the cost. This was developed following extensive literature reviews, online survey, and expert interviews.

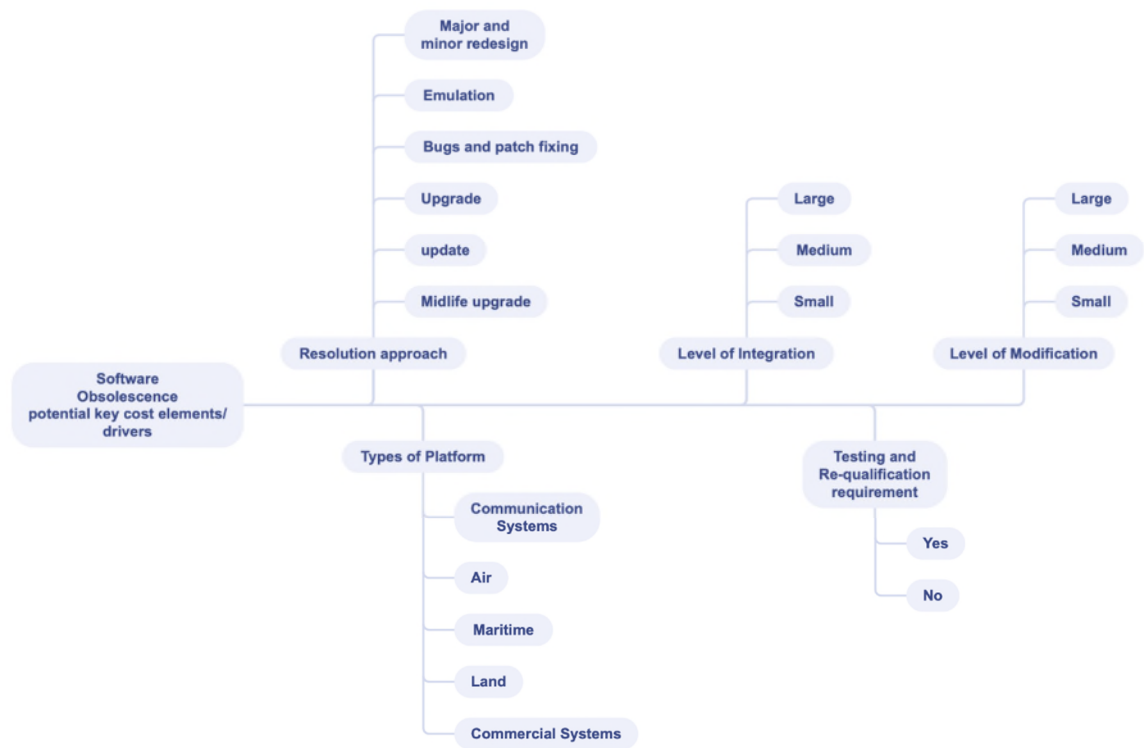


Figure 6-3 Identification of initial key cost drivers

The early research on the best possible ways to identify, manage, mitigate and estimate the cost of resolution approaches is shown in the Figure 6-4 below. This was explored and developed further in this chapter. However, it was clear that there is a certain requirement to develop this into a framework to help the project and programme managers in managing and estimating the cost of software obsolescence at early stages of the project life cycle. This concept was tested with MoD (UK) Obsolescence Manager and further validated through IIOM, ICEAA, SCAF and Software IT CAST Conferences. Detailed validation and verification of this framework is described in detail in chapter 8.

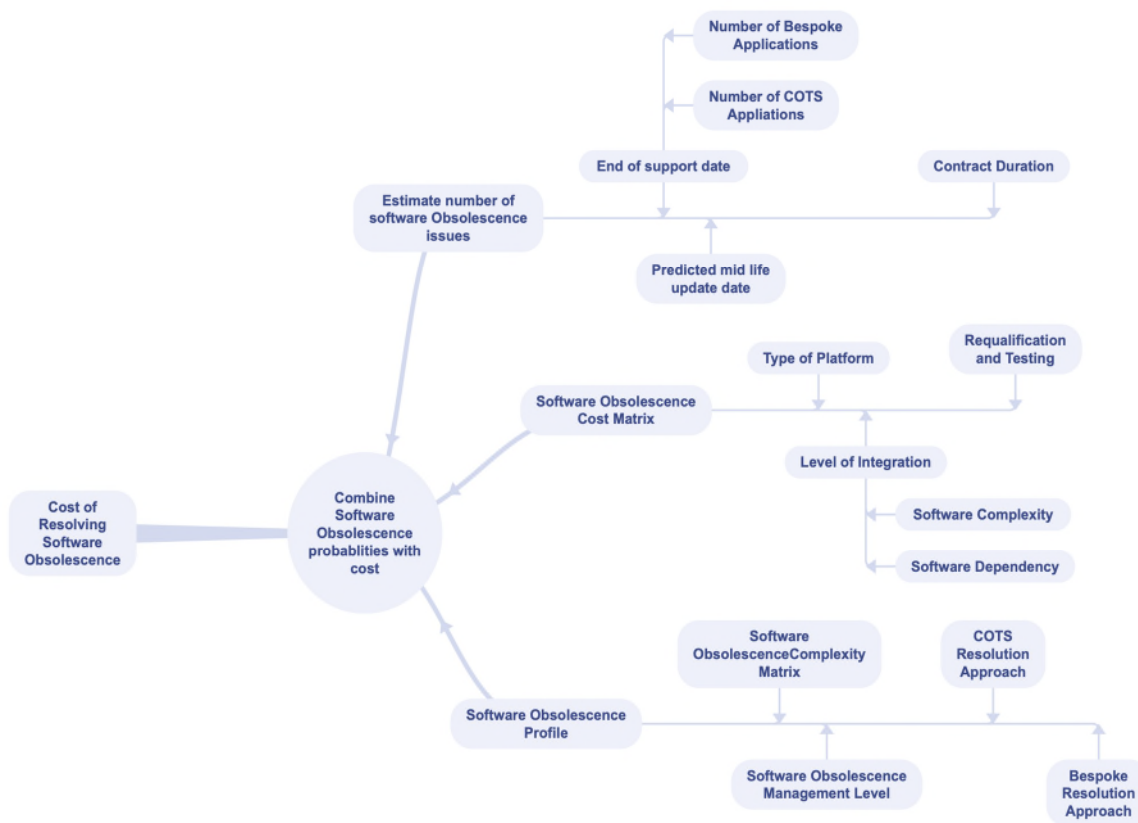


Figure 6-4 Initial understanding and design to estimate the cost of software obsolescence resolutions

6.3 Detailed Research methodology

The research methodology for the Software Obsolescence Cost Analysis Framework followed for the initial requirement is shown in the Figure 3-2. Due to the lack of data and information on software obsolescence it was necessary to follow a scientific method to develop this framework. The first step was to understand the current practices and the accurate representation of methodology used to deal with this issue. Once the requirement was analysed and confirmed, a framework was developed which was then applied to the projects for validation. This framework defined the approach that was required to effectively manage and mitigate the cost of software obsolescence resolution. The framework consisted of process, procedures, techniques and tools which was developed following case studies, literature reviews, online surveys, and cognitive case studies.

The iterative process followed for the development of software obsolescence cost analysis framework is shown in Figure 6-5. Each version was refined further following validation workshops which is further described in Chapter 8.

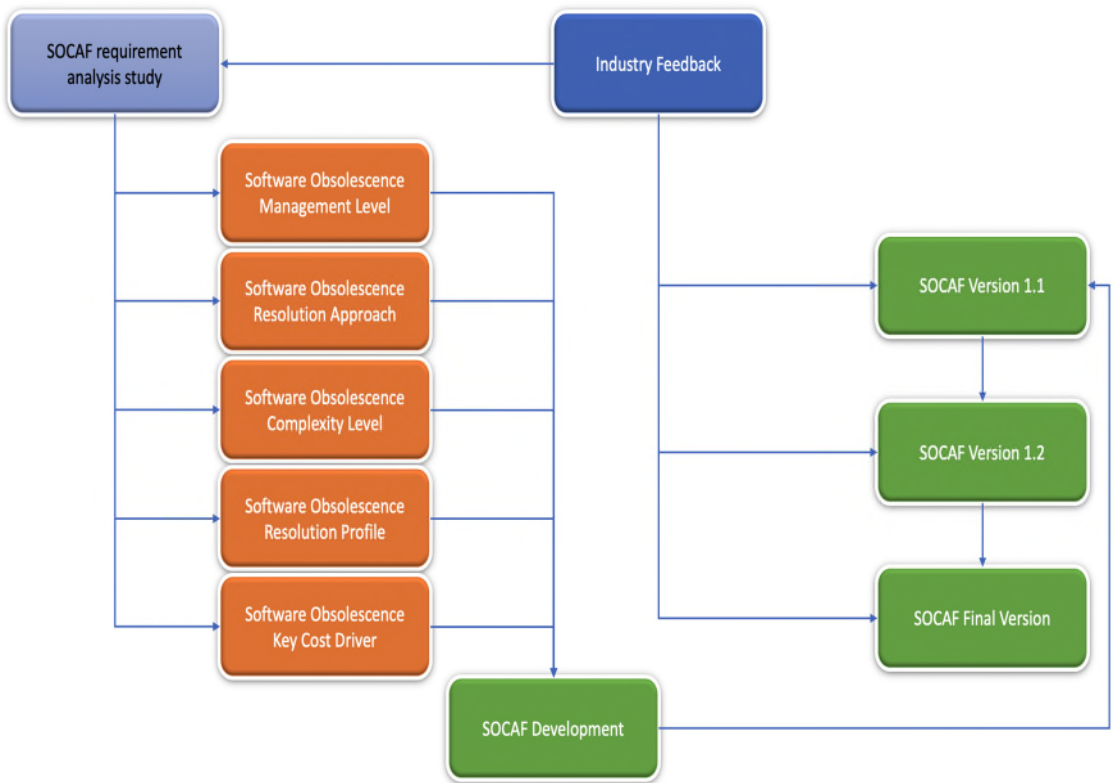


Figure 6-5 Iterative development of Software Obsolescence Cost Analysis Framework

The initial requirement for Software Cost Analysis Framework was captured during interviews and case studies with the experts from Defence and Aerospace industries. Based on these requirements, a draft version of the framework was developed and presented in a workshop with five obsolescence specialists from various organisations within the Defence and aerospace industry. All those experts were obsolescence specialists with 88 years of total experience among them. Table 2-1 below shows the details of the participants.

Table 6-1 Details of the workshop participants

Sl. No	Organisation	Role	Years of Experience	Obsolescence Expert
1	ORG A	Programme Manager	25	Yes
2	ORG A	Obsolescence Manager	15	Yes
3	ORG B	Cost Analyst	13	No
4	ORG C	Systems Engineer	15	Yes
5	ORG D	Software Engineer	20	No

The purpose of this workshop was to refine the initial requirement, identify other requirements and validate the rationale proposed for the framework. This was organised as a half a day workshop during which the draft version of the framework was initially presented and following the discussion, the experts filled in a feedback questionnaire to assess any further enhancement required as shown in Appendix A. Examples of the questions contained in the questionnaire are provided below.

- Would a Software Obsolescence Cost Analysis Framework be beneficial for the Defence and aerospace industries at the bidding stage?
- Assess the relevance of the presented framework to your business
- Assess the benefits of the presented framework to industry
- Indicate improvement required and prioritise them
- Assess the validity of the assumptions made in the framework.

The responses were then analysed by identifying similarities, differences, and any unique responses across the different participants. The outcome from this analysis is summarised further along in this chapter.

6.3.1 Software obsolescence cost analysis framework

The framework has been developed based on the key findings from literature review, cases studies, online surveys, and cognitive case study. This framework

was developed in an iterative way and was reviewed and updated throughout this research. Figure 6-6 shows a final graphical representation of the Software Obsolescence Cost Analysis Framework. The development of this is explained in detail further in this chapter.

Several experts with different backgrounds and from various organisations including BAE Systems, MoD (UK), and QinetiQ etc. participated in the validation workshop to develop the key cost drivers. Their experiences ranged from 4 to 15 years with their roles predominantly covering obsolescence management and through life engineering. This has been covered in detail within Chapter 8.

A brainstorming session in groups, followed by discussion among all the participants, allowed the identification of the key parameters that define the inputs required for the Software Obsolescence Cost Analysis Framework.

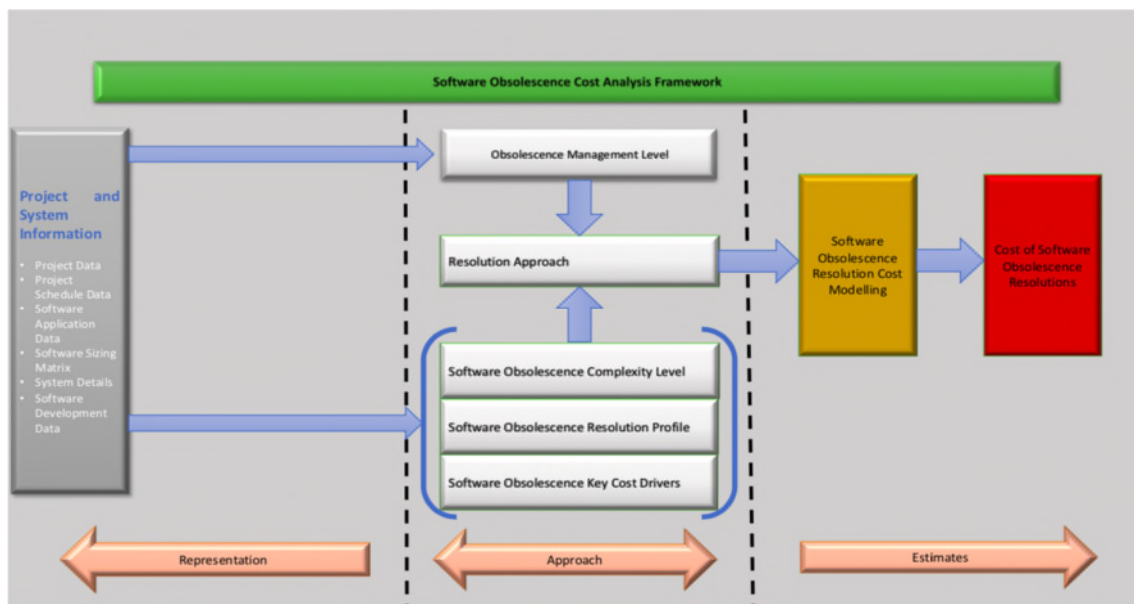


Figure 6-6 Software Obsolescence Cost Analysis Framework

A systematic approach has been followed in the development of this framework. The first phase aimed to gain an understanding of software obsolescence and cost estimation through extensive literature reviews and semi structured interviews. This allowed the identification of key cost drivers of software obsolescence. The approach used to develop the framework is shown in Figure 3-2.

In the second phase, an excel based tool prototype for estimating the cost of resolution of software was iteratively developed and enhanced following validation workshops. This is further explained in Chapter 8. Phase three is where all the different components come together, and the resulting framework was then quantitatively validated using case studies from industry.

Software Obsolescence Framework Requirement

Case studies, interviews, online surveys, and workshops with obsolescence experts enabled the researcher to identify the key factors that play an important role in the software obsolescence and in the identification of the key cost drivers for the resolution of software obsolescence. However, following literature reviews and feedback from expert interviews, it was evident that there is a certain requirement/ criterion that a framework needs to meet, which is explained below.

- **Generalisability**

The framework should be easily adaptable to any type of projects/types of software in the Defence and aerospace industry. The different types of software are listed in section 1.4.2. This is based on the US DoD classification.

- **User- Friendly**

The framework should be easy to use with clear interface along with structured process and procedures to manage and estimate the cost of software obsolescence and appropriate resolution approach.

- **Flexibility**

The framework should be capable to adapt any degree of information available given the fact that all the required information may not be available at early stages of the project life cycle especially within the software development life cycle. The framework should be capable of taking into account any updated information to make adjustments to the estimate so as to form more reliable and credible cost estimates with the available data. It is essential that only one framework is used throughout life cycle to ensure continuity of the estimate.

- **Robust**

The framework should be reliable enough to be used as a basis for contract management and negotiation at the bidding stage of software projects.

- **Distribution based analysis**

Although the framework could generate a single deterministic estimate; however, since the software obsolescence is extremely unpredictable and the resolution of such unpredictable element carries a substantial risk and uncertainty, the framework should be able to undertake a distribution-based analysis to generate a three-point estimate.

During the workshop, the experts highlighted that the framework would provide an effective way to manage and estimate the cost of software obsolescence resolution at the bidding stage. The framework will also enable identification of the key cost drivers for software obsolescence resolutions as well as suitable resolution approach in addition to giving confidence project teams to include software obsolescence in contracts. The experts also indicated that the framework should have the following features.

- Was asked to investigate software supply chain in some detail as it requires the same level of management and monitoring as components.
- Was asked to provide some recommendation on monitoring software supply chain and include that as part of resolution approach.
- Was asked to identify other drivers such as skills, process, and planned obsolescence
- Use Monte-Carlo simulation to take uncertainty into account
- Allow calibration for different projects and domains such as air, land, sea, and commercial systems
- Should be able to work at different levels of the product breakdown structure.

6.3.2 Software Obsolescence Management Level

Requirements

One of the key elements of Software Obsolescence Cost Analysis Framework is to understand how the Defence and aerospace manage their real time custom build software obsolescence issues. It was evident from the case studies, literature searches, online surveys, and cognitive case studies that there was a requirement to develop a Software Obsolescence Management Systems whereby the project manager or someone responsible for Software Obsolescence issues could understand, manage, and mitigate by applying appropriate resolution strategy more effectively.

This requirement led to the development of the Software Obsolescence Management Level which formed a part of the Software Obsolescence Cost Analysis Framework. The other requirements captured during the SME interviews, workshops and literature reviews are listed below.

- There is a requirement to understand where an organisation/ project teams is with respect to managing software obsolescence
- There is a requirement to capture as-is state and provide guidance to improve the management of software obsolescence issue
- Should be able to determine the Software Obsolescence Management Strategy
- Should help the project team/organisation to manage software obsolescence risk
- Should be able to support the requirement of capacity and capability to deal with software obsolescence risk/issues
- Should be able to determine the requirement for deployment of any software obsolescence monitoring systems/tool if any available
- Should be able to help determine the Software Obsolescence resolution approach/strategy
- Should be able to determine the capacity and capability requirement to monitor software supply chain for any potential software obsolescence issues/risks

- Software Obsolescence Management Level should be able to help determine what is required to get to and maintain an optimum level of software obsolescence management, and what they need to undertake to move from being reactive to proactive management of software obsolescence.

Detailed Research Methodology

The Software Obsolescence Management Level study was undertaken in three different phases, which are listed in the Figure 6-7 below



Figure 6-7 Approach undertaken to develop Software Obsolescence Management level

Phase 1 – Data Collection (SME Interviews, online case studies and workshops)

The results from the preliminary study set the appropriate understanding of the requirement to develop the software obsolescence management level. Membership to the organisations such as IIOM, SCAF and ICEAA helped in undertaking the workshop which in turn helped in the iterative development of the Management Level. The research methods used to collect data during the SME interviews and workshops followed a Delphi method. The Delphi method is an “iterative process to collect and refine the anonymous judgement of experts using a series of data collection and analysis methods followed by a good feedback loop” (Skulmoski, Hartman F.T. and Krahn, 2007). Delphi is well suited to exploratory studies such as the development of the software obsolescence management level where there is a lack of appropriate historical data (Linstone and Turoff, 1975; Okoli and Pawlowski, 2004) and expert judgement is the only option (Rowe and Wright, 1999).

A questionnaire was developed to be used in the first round of the data collection with SME interviews, IIOM, and SCAF Workshops. These questions were piloted with MoD Obsolescence experts for validation and verification to make the question clear and unambiguous. The participants were asked about their experience level, combined with the level of software obsolescence management been currently undertaken at organisational level or at the project level and finally the question set covered the likelihood of them using the initially developed software obsolescence management level. (Ralph L. and von Winterfeldt, 1988) supported the use of probability to quantify expert judgement in validating a complex technical and engineering problems. A total of 15 obsolescence experts with various degrees of software obsolescence management experience (please note none of them were software obsolescence experts but had to deal with emerging issues with software obsolescence by either passing the risk to OEM or reactively dealing with software obsolescence issues) participated in the first round of the data collection, and their responses were subsequently analysed.

Table 6-2 shows the years of experience of participants who took part in this workshop.

Table 6-2 Number of participants and years of experience

Years of Experience	Number of Participants
Less than 5 years	1
6 years to 10 years	3
11 years to 20 years	11

The outcome of this analysis was presented at a second round of workshop to 10 obsolescence experts with various levels of software obsolescence management interest. 50% of the participants were from the first round of workshops. They took the opportunity to discuss about the results either by corroborating the results or amending them. A final analysis of these responses were allowed to refine the outcome of the first round, providing as a result, a set of Software Obsolescence Management Levels. The results of the first and second round converged and therefore no further iterations on the Delphi was needed.

Phase 2 –Software Obsolescence Management Levels Refinement

The second phase of this study investigated the refinement of Software Obsolescence Management Levels. In this phase a draft version of the Management Levels was developed which was then used in the refinement workshop with 15 experts from the Defence and aerospace industry. The workshop was undertaken at SCAF, IIOM and ICEAA Workshops. An updated question set was developed to undertake the refinement exercises. This workshop played an important role in defining a common understanding of different aspects of the management for e.g., having a software obsolescence monitoring tool etc.; along with understanding what a good proactive management should cover. The responses were collected and analysed with eventually the mean was applied for the refinement of the software obsolescence management levels from the Delphi study.

Phase 3 – Software Obsolescence Management Level Validation

The objective of this phase was to validate the contents of the Software Obsolescence Management Levels by checking their plausibility (Robson, 2002) and ensuring that the contents of each management level was clear and logical. The experts discussed each content of the management level and concluded whether it complemented their anticipated trend based on their experiences and justified them.

Two workshops to validate the Software Obsolescence Management Level and one further workshop to validate to confirm Software Obsolescence Management Level will help in identifying the key cost driver was undertaken. The first two validation workshops was attended by 15 Obsolescence Experts from IIOM, and BAE System, with the key cost driver validation workshop being attended by 5 cost experts from ICEAA and SCAF. The experts agreed that the Software Obsolescence Management Level was being pitched at the right level. They also agreed that this will help understand where the organisation or individual project or programme are with respect to managing software obsolescence and did provide recommendation as to what level they need to be at to deal with software obsolescence issues proactively. The feedback included the need to have a process/guidance on what the teams need to undertake or improve upon to move from Level one to Level five of the management levels. This is partially addressed further down in these sections.

Final iteration of the Software Obsolescence Management Level is shown in the Figure 6-8 below. Software Obsolescence Management Level will help the organisation/project team understand what they need to do to get to an optimum level of Software Obsolescence Management. The organisation/project team at the lowest level deals with software obsolescence reactively while the organisation/project team at the highest-level deals with the software obsolescence proactively.

However, the teams at higher levels may have a higher overhead which potentially could only be suitable for larger organisation but could likely be expensive to small/medium size organizations. Due to these reasons,

organisation should undertake a sequential trade off to get optimum benefits out of the Software Obsolescence management level. In this classification, Level 1 represents a purely reactive strategy whereas level 5 represents the most proactive measures, where all the following proactive strategies are taken.

- Employ software obsolescence Managers
- Deploy and use software obsolescence monitoring tools
- Manage software obsolescence issue proactively
- Monitoring technological insertion
- Escrow methodology and
- Considering software obsolescence at the design and development stages.

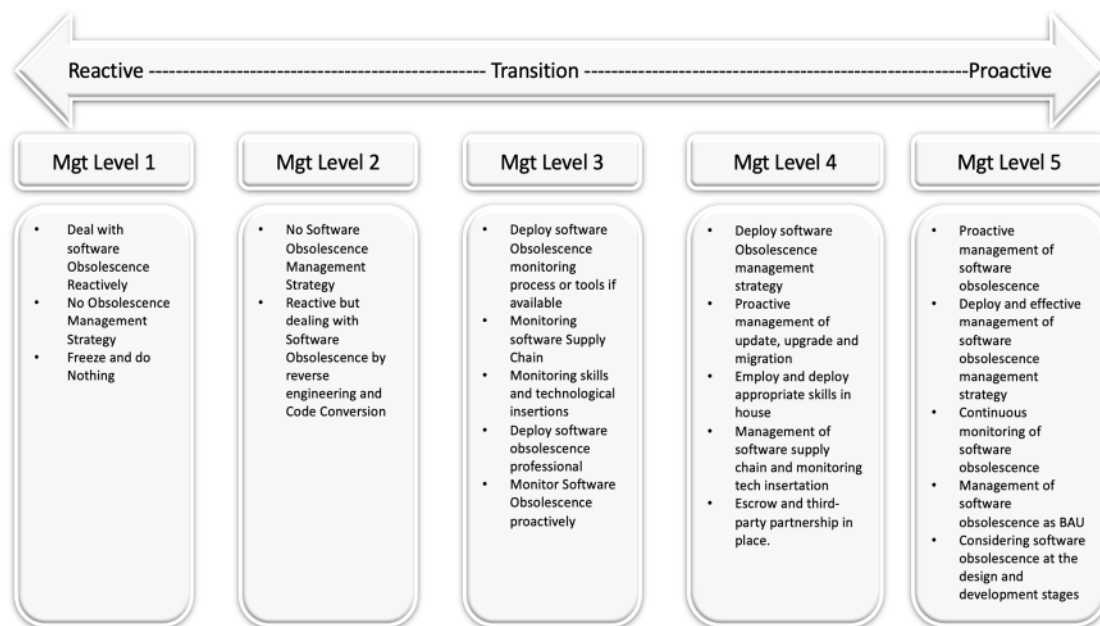


Figure 6-8 Software Obsolescence Management Level

6.3.3 Software Obsolescence Resolution Approach

Requirements

One of the key elements of Software Obsolescence Cost Analysis Framework is to understand how the Defence and aerospace industry mitigate the software obsolescence issue for their real time custom build software. To establish an appropriate mitigation strategy, it is important to establish the appropriate

resolution approach. It was evident from the case studies, literature searches, online surveys, and cognitive case studies that there was a requirement to develop a new and innovative Software Obsolescence Resolution approach whereby the project manager or someone responsible for Software Obsolescence issues can understand, manage, and mitigate the issues by applying appropriate resolution strategy more effectively.

This requirement led to the development of the Software Obsolescence Resolution Approach which formed a part of the Software Obsolescence Cost Analysis Framework. There are three kinds of software resolutions identified through the literature reviews, (Sandborn p, 2009) and case studies.

- **Technological Resolution-** This resolution approach involves the identification of the technology used such as compilers, software languages, whether it's been purchased or licenced etc. Once identified, then the appropriate resolution strategy should be applied.
- **Functional Resolution-** This resolution approach involves the identification of any issues related to hardware and software compatibility and integration, such as software to software integration.
- **Logistical Resolution-** This approach involves the identification of any infrastructure or network issues and mitigating them.

The other requirements captured during the SME interviews are listed below.

- There is a requirement to understand what an organisation/ project teams need to undertake to mitigate the software obsolescence issue.
- There is a requirement to capture as-is state and provide guidance to improve the management of software obsolescence issue
- Should be able to determine the Software Obsolescence Management Strategy
- Should help the project team/organisation to manage software obsolescence risk
- Should be able to determine the requirement for deployment of any software obsolescence monitoring systems/tool if any available

- Should be able to help determine the Software Obsolescence resolution approach/strategy
- Should be able to determine the capacity and capability requirement to monitor software supply chain for any potential software obsolescence issues/risks

Detailed Research Methodology

The Software Obsolescence Resolution Approach study was undertaken in three different phases, which are listed in the Figure 6-9 below.

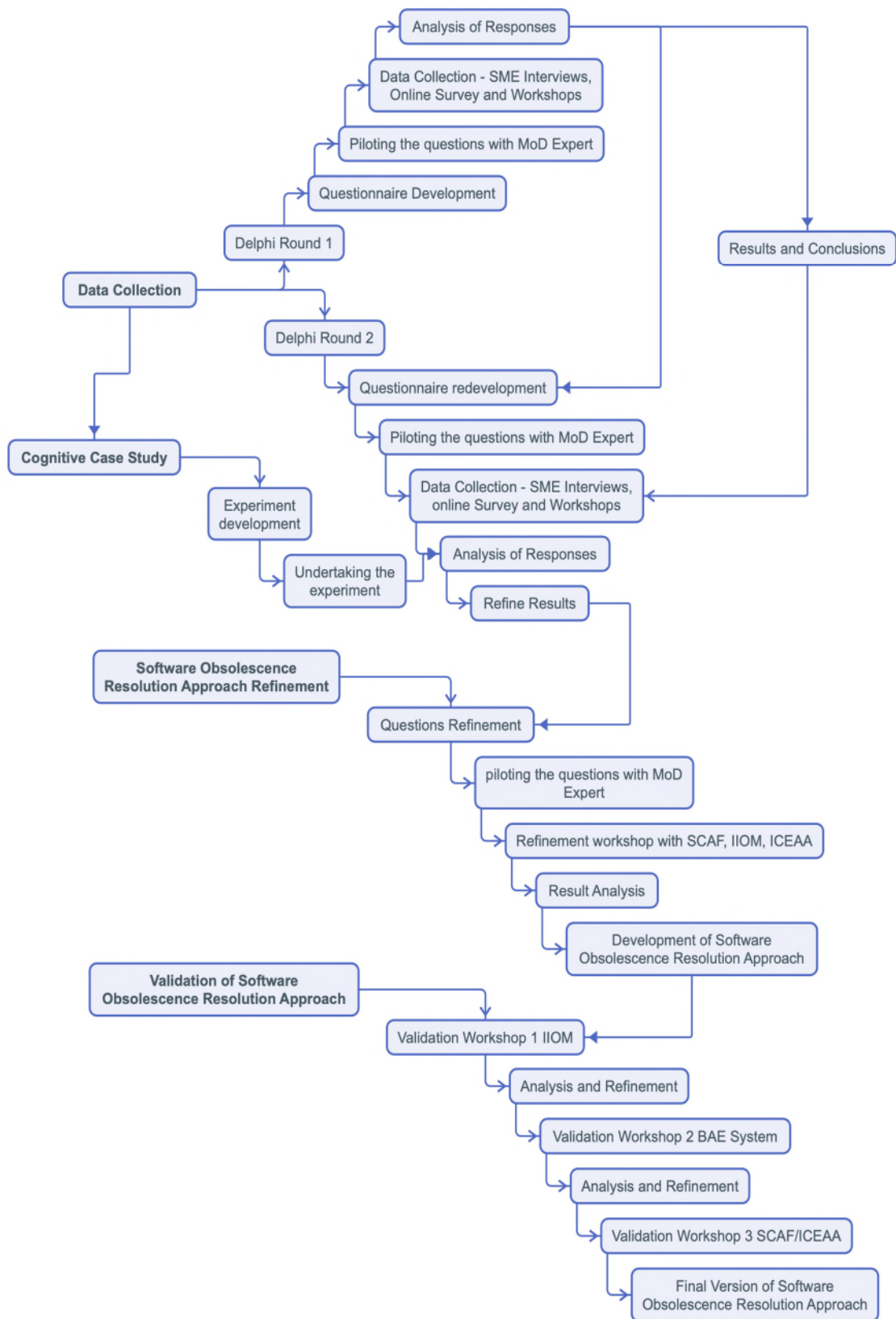


Figure 6-9 Approach undertaken to develop Software Obsolescence Resolution approach

Phase 1 – Data Collection (SME Interviews, online case studies, Cognitive Case Study, and workshops)

The results from the preliminary study set the appropriate understanding of the requirement to develop the Software Obsolescence Resolution approach. Membership to the organisations such as IIOM, SCAF and ICEAA helped in undertaking the workshop which in turn helped in the iterative development of the software obsolescence resolution approach. The research methods used to collect data during the SME interviews and workshops followed a Delphi method. The Delphi method is an “iterative process to collect and refine the anonymous judgement of experts using a series of data collection and analysis methods followed by a good feedback loop” (Skulmoski et al., 2007). Delphi is well suited to exploratory studies such as the development of the software obsolescence management level where there is a lack of appropriate historical data (Okoli and Pawlowski, 2004; Linstone and Turoff, 1975) and expert judgement is the only option (Rowe and Wright, 1999).

A questionnaire was developed to be used in the first round of the data collection with SME interviews, IIOM, and SCAF Workshops. These questions were piloted with MoD Obsolescence experts for validation and verification to make the question clear and unambiguous. The participants were asked about their experience level, combined with the level of software obsolescence management being currently undertaken at organisational level or at the project level, and finally the question set covered the likelihood of them using the initially developed software obsolescence resolution approach. Keeney et al (1988) supported the use of probability to quantify expert judgement in validating a complex technical and engineering problems. A total of 15 obsolescence experts with various degrees of software obsolescence management experience (please note none of them were software obsolescence experts but had to deal with emerging issues with software obsolescence by either passing the risk to OEM or reactively dealing with software obsolescence issues) participated in the first round of the data collection, and their responses were subsequently analysed. Table 6-2 shows the years of experience of participants who took part in these workshops.

The outcome of this analysis was presented at a second round of workshop to 10 obsolescence experts with various levels of software obsolescence management interest. 50% of the participants were from the first round of workshop. They took the opportunity to discuss about the results either by corroborating the results or amending them. A final analysis of these responses was allowed to refine the outcome of the first round providing as a result, a set of software obsolescence Resolution Approaches. The results of the first and second round converged and therefore no further iteration on the Delphi was needed.

As discussed in Chapter 5, a cognitive case study was undertaken to understand the reasoning behind the technology selection by the software developers which will have an effect in the software obsolescence issue further down the software life cycle. This is key as it will help in identifying the appropriate resolution approach that needed to take.

Phase 2 – Software Obsolescence Resolution Approach Refinement

The second phase of this study investigates the refinement of software obsolescence resolution approaches. In this phase, a draft version of the resolution approach was developed using software supply chain model by Bjoern Bartels et al., 2012 as the base of this model. This was due to the identification of the importance to know the software supply chain, as was to establish resolution approaches that can be deployed. This is due to the fact that software could potentially go obsolete, even if any one element within the supply chain identified an issue or a problem. A reverse engineering principle was introduced into the supply chain model where the three software resolution strategies identified by Sandborn et al (2009) was mapped to this model.

This model was then tested with 15 experts from within the Defence and aerospace industry, at SCAF, IIOM and ICEAA Workshops. An updated question set was developed to undertake the refinement exercises. The responses were collected and analysed, and the mean was applied for the refinement of the software obsolescence management levels from the Delphi study.

Phase 3 – Software Obsolescence Resolution Approach Validation

The objective of this phase was to validate the contents of the software obsolescence resolution approach by checking the plausibility (Robinson, 2002) and ensuring that the content within each of the obsolescence resolution approach was clear and logical. The experts discussed each content of the resolution approaches and concluded whether it complimented their anticipated trend based on their experiences and justified them.

Two workshops to validate the Software Obsolescence resolution approach and further one to validate the Software Obsolescence resolution approach that would help in identifying the key cost driver was undertaken. The first two validation workshop was attended by 15 Obsolescence Experts from IIOM, and BAE System, with the key cost driver validation workshop being attended by 5 cost experts from ICEAA and SCAF. The experts agreed that the Software Obsolescence resolution was being pitched at the right level. They also agreed that this will help understand where the organisation or individual project or programme are with respect to managing and mitigating software obsolescence; and did provide recommendation on the appropriate resolution action that they needs to be undertaken to deal with software obsolescence issues proactively. Final iteration of the Software Obsolescence Resolution approach is shown in the Figure 6-10 below.

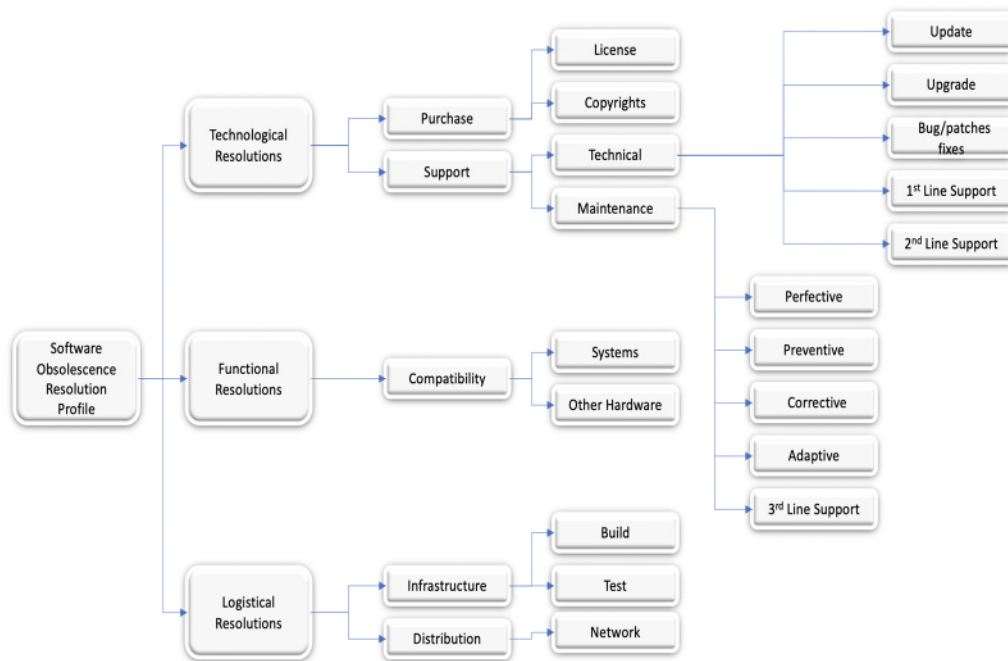


Figure 6-10 Software Obsolescence Resolution Approach

The Obsolescence Resolution approach is divided into three main categories

Technological Resolution

This resolution approach involves the identification of the technology used such as compilers, software languages, whether it's been purchased, licenced etc., once identified, then appropriate resolution strategies should be applied.

This can be further divided into two subsets

- o **Purchase** – this is where the company procures the software outright by means of licences or copyrights.
 - **Licences** – this is the purchase of the software with licences or procuring licences regularly for continual usage of the software. If the licences are not procured based on the contractual agreement, then the updated version of the software releases may not be available for the users and the OEM will cease to support the software which will most likely make the software obsolete.

- **Copyright** – this is when the customer procures the whole of the software including the copyright to the software. This is largely applicable to the custom build software within defence and aerospace industry. This will help the customer in identifying a different supplier or developer and have an Escrow agreement as a mitigation if the OEM faces any issues as suggested by few experts in the online survey.
- o **Support** – this determines the support required to run the software daily as business as usual. The support is largely divided into two types for this research.
 - **Technical** – this type of support involves the following
 - **Update** – this is the process of fine tuning a software product to make it perform better. They are usually small, frequent improvements.
 - **Upgrade**- instead of building on an existing program, a software upgrade is a new version of the existing software with enhancement. Upgrades are generally planned in, and contracts are drawn based on these requirements.
 - **Bug/ patches fixes** – A patch is a set of changes to a software design to fix and improve the software; this includes fixing any security vulnerabilities and bugs.
 - **1st line support** – the first line of support provides the very basic support to more common problems/ issues
 - **2nd line support** - the second line of support provides the support to more complex problems/ issues
 - **Maintenance**- Software maintenance is the modification of a software after delivery to correct faults, to improve performance or other attributes. Maintenance is further classed into (Adapted from ISBSG)

- **Perfective Maintenance** - Perfective Maintenance is the modification after delivery to improve performance or maintainability.
- **Preventive Maintenance** – The modification of a software application after delivery to detect and correct latent faults in the software product before they become effective faults.
- **Corrective Maintenance** - the reactive modification of a software product performed after delivery to correct discovered problems
- **Adaptive Maintenance**- Enhancement necessary to accommodate changes in the environment in which software product must operate.

Functional Resolutions

This resolution approach involves the identification of issues related to hardware and software compatibility and integration such as software to software integration. Functional compatibility is the key part of this resolution approach.

- **Compatibility** - In this resolution approach, the compatibility of the software to systems and/or other hardware is explored. Both software to software integration and software to hardware integration are included.
 - o **Systems**- this investigates the software-to-software compatibility and other systems compatibility.
 - o **Other Hardware**- This investigates the software to hardware compatibility and integration.

Logistical Resolution

This approach involves the identification of any infrastructure or network issues and mitigating them. Logistical resolution can be further broken down to Infrastructure and Distribution resolution approaches.

- **Infrastructure** – This type of resolution approaches investigates the system build and the testing environment
- **Distribution** – this type of resolution approach looks at the network capability that support the systems/ software etc.

Benefits of Software Obsolescence Resolution Approach

The Software Obsolescence Resolution approach developed is part of the Software Obsolescence Cost Analysis Framework and support the development and identification of the key cost drivers for the resolution of software obsolescence issues. It also provides a credible resolution strategy which can be identified and implemented at very early stages of the project life cycle. Other key benefits of the software obsolescence resolution strategy are listed below.

- Software Obsolescence Resolution approach helps to tease out the resolution strategy.
- The resolution approach will help in identification of the best possible resolution techniques.
- Resolution techniques are determined based on the Project and Systems Parameters and Software Management Level Resolution techniques are identified for individual software components rather than software program.
- Software Obsolescence Resolution approach helps the project team to compile an appropriate software obsolescence strategy.
- Software Obsolescence Resolution approach will help to identify the key cost drivers.
- Software Obsolescence Resolution approach will help to identify the key risk/ uncertainties around the selection of appropriate resolution approaches.

6.3.4 Software obsolescence complexity level

Requirements

Another key element of Software Obsolescence Cost Analysis Framework is to understand how complex is the software in terms of managing software obsolescence issue in the Defence and aerospace industry. This is key as this will dictate the appropriate resolution approach that needs to be implemented to resolve the software obsolescence issues. It is evident from the case studies, literature searches, online surveys, and cognitive case studies that there is a requirement to develop a new software obsolescence complexity matrix to help project manager or someone responsible for Software Obsolescence to identify how complex the software obsolescence issue is and make an informed decision on the mitigation strategy that is applicable to the specific software obsolescence issue proactively. The level of complexity will influence the probability of using each resolution approach to resolve the software obsolescence issue. Software obsolescence complexity matrix also helps to determine the cost and resource requirement to resolve an obsolescence issue. For instance, if the project has customs software, then it would require specialist skills and resources to be maintained and supported in-house, whereas if the application is an open source, then the resource requirement to support and maintain is very low. This requirement led to the development of the Software Obsolescence Resolution Approach which formed a part of the Software Obsolescence Cost Analysis Framework.

Detailed Research Methodology

The Software Obsolescence Complexity Level study was undertaken in three different phases, which are listed in the Figure 6-11 below.

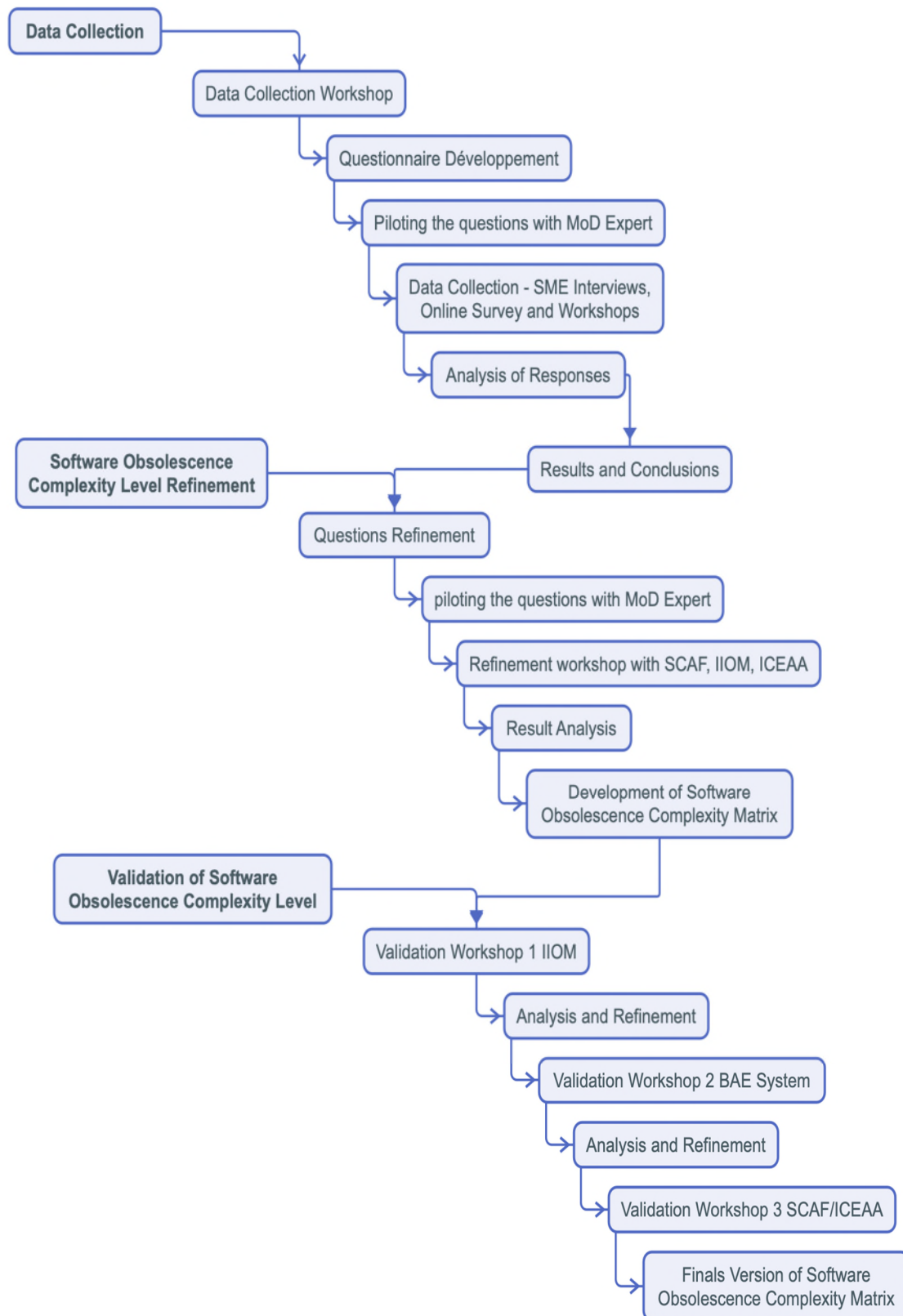


Figure 6-11 Approach undertaken to develop Software Obsolescence Complexity Level

Phase 1 – Data Collection (SME Interviews, online case studies and workshops)

The results from the preliminary study set the appropriate understanding of the requirement to develop the software obsolescence complexity Matrix. Membership to the organisations such as IIOM, SCAF and ICEAA helped in undertaking the workshop which in turn helped in the iterative development of the complexity matrix. The research methods used to collect data during the SME interviews and workshops followed a Delphi method. The Delphi method is an “iterative process to collect and refine the anonymous judgement of experts using a series of data collection and analysis methods followed by a good feedback loop” (Skulmoski et al., 2007). Delphi is well suited to exploratory studies such as the development of the software obsolescence management level where there is a lack of appropriate historical data (Okoli and Pawlowski, 2004; Linstone and Turoff, 1975) and expert judgement is the only option (Rowe and Wright, 1999).

A questionnaire was developed to be used in the first round of the data collection with SME interviews, IIOM and SCAF Workshops. These questions were piloted with MoD Obsolescence experts for validation and verification to make the question clear and unambiguous. The participants were asked about their experience level, combined with the level of software obsolescence management been currently undertaken at organisational level or at the project level and finally the question set covered the likelihood of them using the initially developed software obsolescence complexity matrix. Keeney et al (1988) supported the use of probability to quantify expert judgement in validating a complex technical and engineering problems. A total of 15 obsolescence experts with various degrees of software obsolescence management experience (please note none of them were software obsolescence experts but had to deal with emerging issues with software obsolescence by either passing the risk to OEM or reactively dealing with software obsolescence issues) participated in the first round of the data collection, and their responses were subsequently analysed. Table 6-2 shows the years of experience of participants who took part in these workshops.

The results of the first round were seen as adequate to converged hence no further iteration on the Delphi was considered.

Phase 2 – Software Obsolescence Complexity Matrix Refinement

The second phase of this study investigates the refinement of the software obsolescence complexity matrix. In this phase a draft version of the complexity matrix was developed. This model was then tested with workshop with 15 experts from Defence and aerospace industry at SCAF, IIOM and ICEAA Workshops. An updated question set was developed to undertake the refinement exercises. The responses were collected and analysed, and the mean was applied for the refinement of the software obsolescence management levels from the Delphi study.

Phase 3 – Software Obsolescence Complexity Matrix Validation

The objective of this phase was to validate the contents of the software obsolescence complexity by checking the plausibility (Robinson, 2002) and making sure that the contents within each obsolescence complexity matrix was clear and logical. The expert discussed each content of the complexity matrix and concluded whether it complemented their anticipated trend based on their experiences and justified them.

Two validation workshops to validate the Software Obsolescence complexity matrix and one to validate the whether the complexity matrix will help to identify the key cost driver was undertaken. The first two validation workshops were attended by 15 Obsolescence Experts from IIOM, and BAE System with the key cost driver validation workshop being attended by 5 cost experts from ICEAA and SCAF. The experts agreed that the Software Obsolescence complexity matrix was being pitched at the right level. They also agreed that this helped understand where the organisation or individual project or programme are with respect to managing and mitigating software obsolescence; and did provide recommendation on the appropriate resolution action that they need to be undertaken to deal with software obsolescence issues proactively. Final iteration

of the Software Obsolescence Complexity Matrix is shown in the Figure 6-12 below.

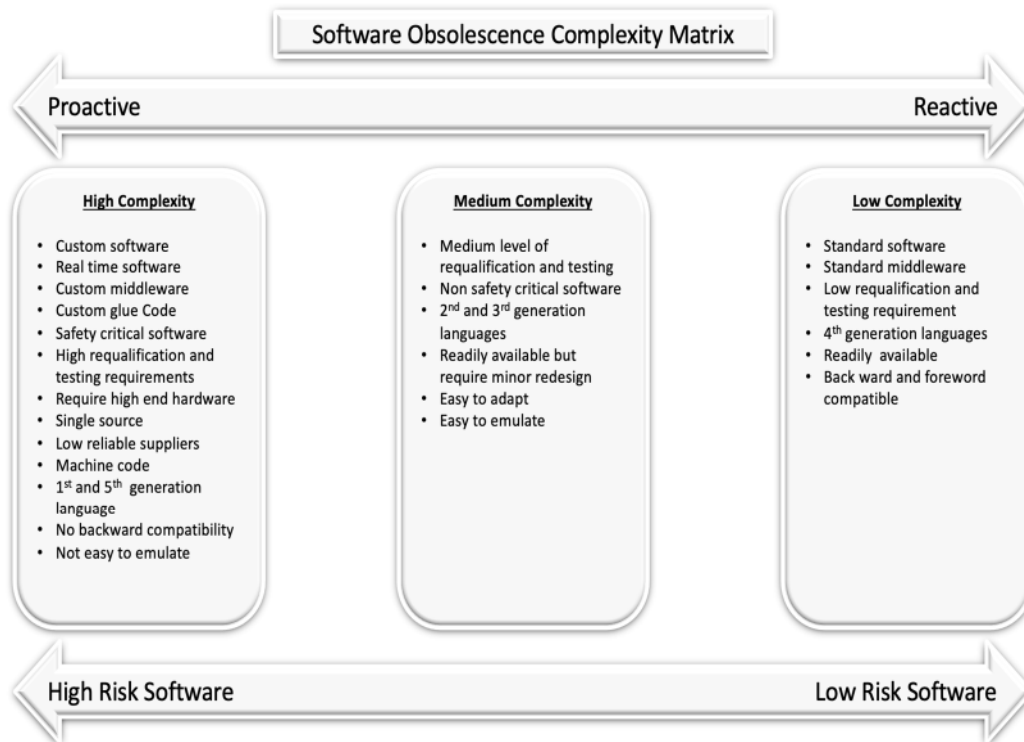


Figure 6-12 Software Obsolescence Complexity Matrix

The level of complexity will influence the probability of using each resolution approach to resolve a software obsolescence issue. Software can be classified into three obsolescence categories based on the complexity level- Low, medium, and high. Each category and the characteristics that define it are shown in the above Figure 6-12 . High complexity components are characterised as bespoke, custom build, cutting edge technology with no backward compatibility. These types of software are largely from single source suppliers which are usually small or with less reliability and are not easy to emulate.

Medium complexity software is usually readily available and easy to adapt. These are generally non-safety critical software and are written largely using 2nd and 3rd generation languages.

Low level complexity software is characterised as standard software which has got low requalification and testing requirement, written in 4th generation

languages or current trending languages, and are easily emulated with backward and forward compatibility.

Benefits of Software Obsolescence Complexity Matrix

It was evident from literature reviews, online surveys, cognitive case studies that there is a requirement to understand the complexity of software obsolescence. Having a complexity level for software obsolescence will help the project manager to identify the complexity of software obsolescence issue in addition to assisting to determine the cost and resource requirement to mitigate the obsolescence issues. For example, if the project has a custom software, then it would require the specialist skills and resources to be maintained and supported in-house, whereas if the application is open source, then the resource requirement to support and maintain would be very low.

Other benefits of software obsolescence complexity matrix are listed below:

- Software obsolescence complexity matrix determine the complexity of resolving a software obsolescence issue at a program level or at an individual component level.
- It will help the project manager in undertaking appropriate resolution approach based on the complexity matrix.
- It supports the project manager in understanding the risk involved in managing software obsolescence for high complex software
- Highly complex software component/program will carry higher risk of managing the software obsolescence
- Highly complex software component/ program requires proactive management of its obsolescence issues.

6.3.5 Software obsolescence key cost drivers

Identifying key cost drivers for the software obsolescence resolution was the key part of the software obsolescence cost analysis framework. An early indication of the key cost drivers was established during the literature research, case studies, and online surveys. During the first half of the workshop, the experts were encouraged to brainstorm in groups, followed by discussion among all the

participants, to identify the key parameters that defines the complexity of a system and influences the obsolescence resolution cost. The discussion held during the workshop resulted in the experts concurring with the following key software obsolescence cost drivers:

- Resolution Approach – these are applied to resolve the software obsolescence issue. This list of resolution approaches is defined in Figure 6-10
- Software/Systems multipliers – these are unique to software programmes and has been identified as a key cost driver. These drivers are
 - o Software complexity – this is defined as the time required for data to pass from point A to point B. This is characterised as High, Low and Medium
 - o Software dependency – this is defined as how dependent the system is on the software. This is characterised as High, Low and Medium.
 - o Software languages – this is the language that the software is written in. Languages range from machine coding and testing to fifth generation languages. 1st 2nd and 5th generation languages tend to be more expensive as there are not enough SQEP available in the market.
 - o Software Development cycles – this is defined as the life cycle that software developers undertake to develop the software. Software development ranges from Waterfall to Agile.
- Type of platform- they can be divided into five categories, which are shown as follows.
 - o Commercial software
 - o Systems and communications
 - o Air domain
 - o Maritime
 - o Land

- The level of integration – both software to software and software to hardware.
- Requalification and testing – this depend upon the level of safety/criticality of the obsolete component, the required level of reliability and if there is a legislative approval that needed to apply.
- Level of modification required- this is classed into high, medium, and low.

Figure 6-13 below shows the detailed view of the software obsolescence key cost drivers.

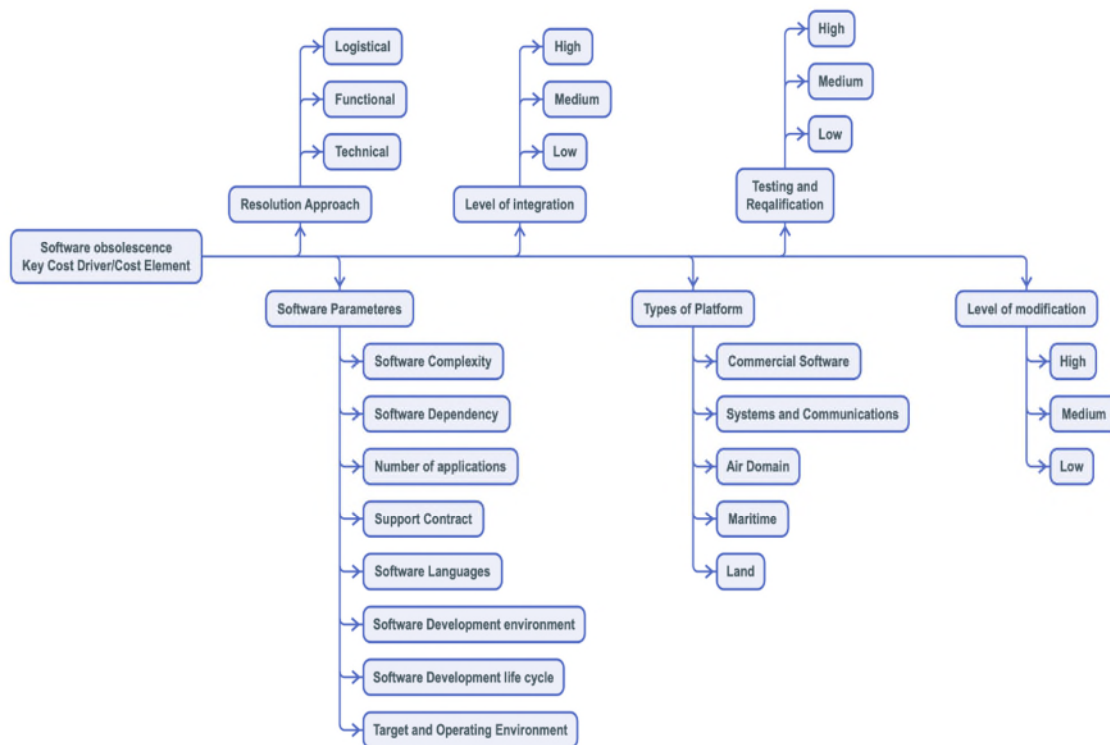


Figure 6-13 Software Obsolescence Key Cost Driver and Cost Elements

Data flow in the Software Obsolescence Cost Analysis Framework

The software obsolescence cost analysis framework is an amalgamation of various processes and tools, all of which are dependent on inputs and outputs to generate a consistent cost output.

Figure 6-14 outlines how the data flows in the framework to generate and estimate the resolution cost of software obsolescence. The main elements are listed below

- Software Obsolescence Resolution Profile
- Software Obsolescence Cost Drivers
- Software Obsolescence Management Level
- Software Obsolescence Complexity Level

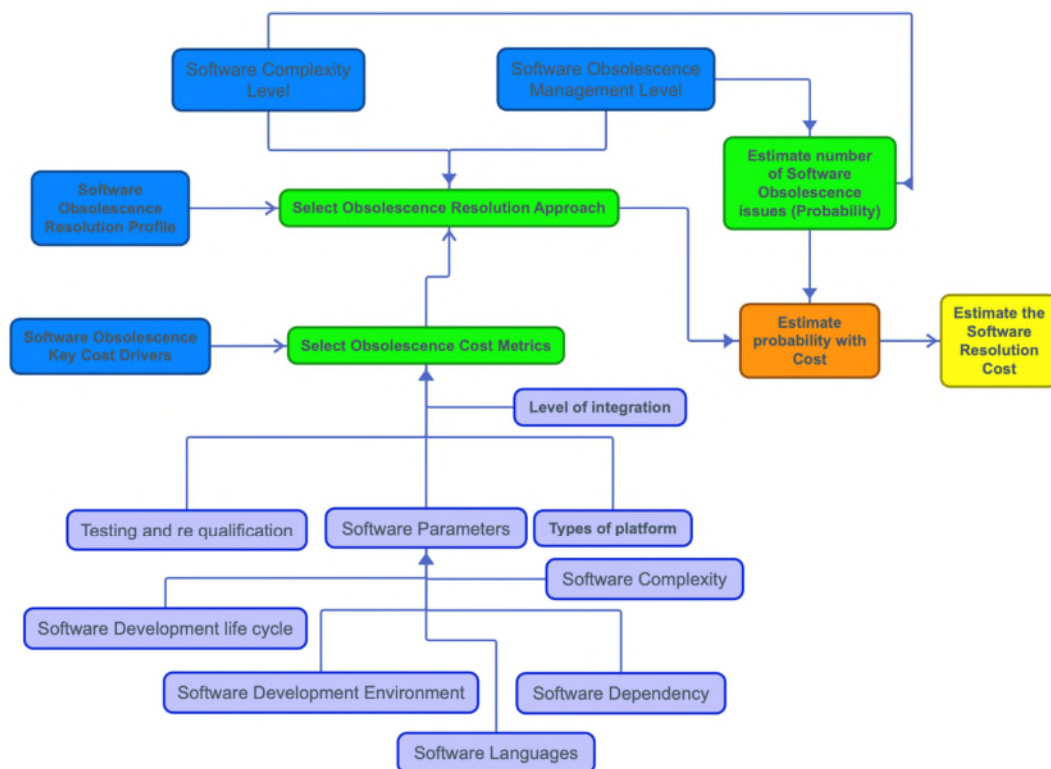


Figure 6-14 Data flow of the software obsolescence cost analysis framework

The number of obsolescence issues and its probability is calculated based on the information available, such as:

- Support contract duration
- Level of software obsolescence management level
- Schedule data
- Complexity of obsolescence in software
- Types of application and number of applications

- Software languages etc.

The collection of processes and tools within the framework are dependent on each other to generate the cost. The steps to estimate the resolution cost is detailed in Figure 6-15 below.

The first step is to identify who is providing the information and when. This is key to provide traceability and accountability of the cost estimate. The second step requires information about the systems that the software sits on. This is key in understanding the external influences on cost. The typical information that is required at this stage based on literature reviews, interviews and cases studies includes but not limited to:

- Project data
- Schedule data
- Type of platform
- Software Development life cycle
- System information
- Software sizing information
- Contract durations and
- Any breakdown of the systems into individual component

The framework is flexible enough to adapt to any level of information available and will still be able to estimate the resolution cost of software obsolescence, where the level of uncertainty is related to the level of information available.

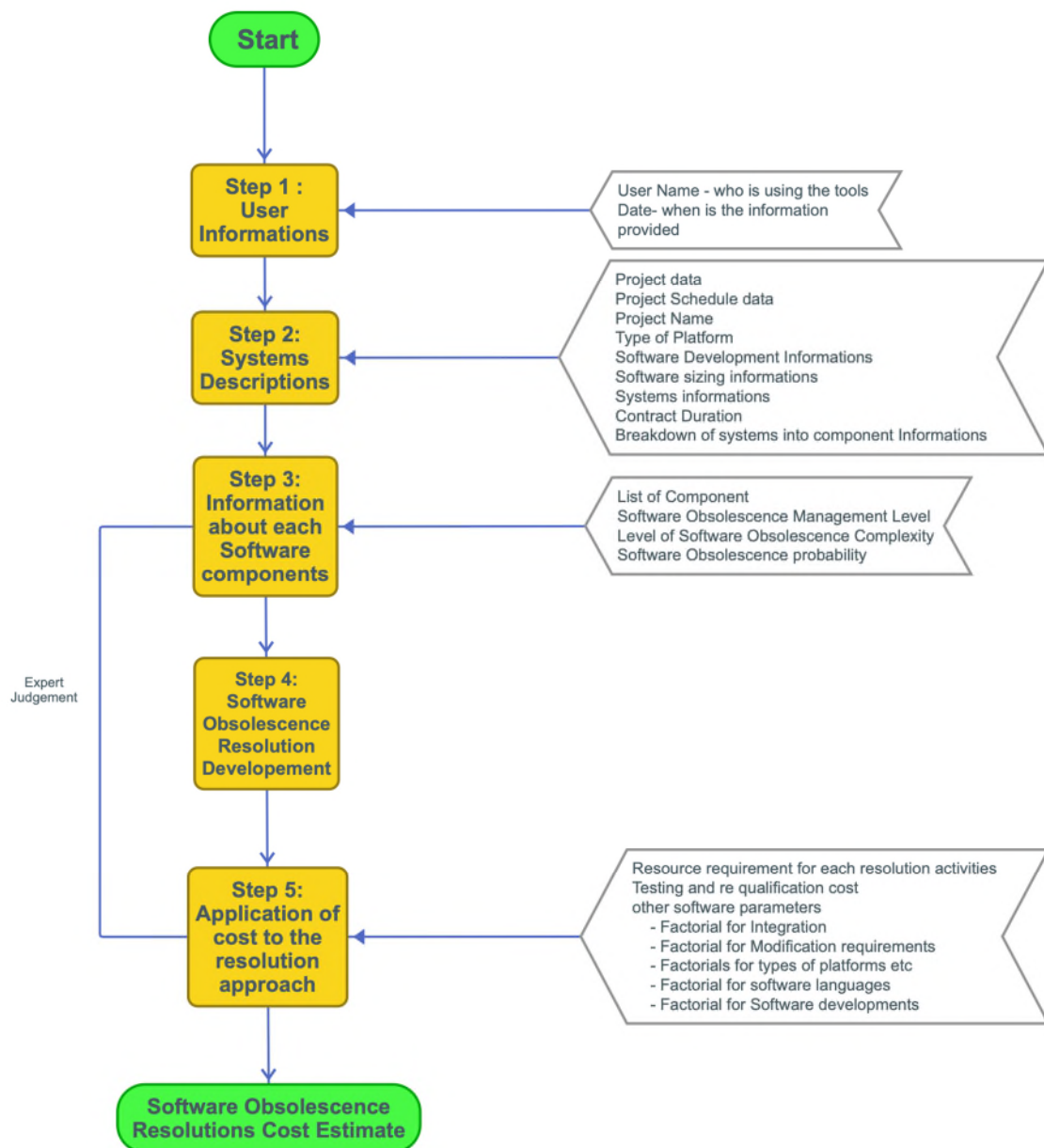


Figure 6-15 Steps followed to estimate the cost of software obsolescence resolutions (Adopted from Javier Romero Rojo et al, 2010)

In step 3, information about each component is dissected in detail, where information on each component is analysed which is then combined with software obsolescence management level. A closer look at the software obsolescence probability is also undertaken. The complexity of software obsolescence is also determined at this stage which will help predict the resolution approach.

In step 4, a resolution approach is predicted and developed for the entire life of the individual software component based on the information provided in steps 1 to 3. This is a key segment of the framework which plays an important part in determining the resolution strategy with the recommended resolution approach.

Step 5, is the last step where the resolution approach that was developed is combined with the factorial (these are discussed in detail in the next chapter) for the external system factors such as:

- Factorial for level of integration
- Factorial for level of modification requirement
- Factorial for types of platforms
- Factorial for software languages
- Factorial for software development life cycle etc.

Expert judgments are used (Rush and Roy, 2001) where there is a lack of sufficient data. The software obsolescence resolution profiles resulting from this study can be customised to reflect the current practices in the project for which this framework can be applied. The framework can be calibrated and presented better with the availability of historical data.

6.4 Summary of observations

In this chapter, the development of the Software Obsolescence Cost Analysis Framework was presented. This framework can be used at the bidding stage of the contract in order to estimate the software obsolescence resolution cost. For the development of this framework, it was necessary to carry out four studies which are listed below.

- Software Obsolescence Management Level
- Software Obsolescence Resolution approach
- Software Obsolescence Complexity Matrix
- Software Obsolescence Key Cost Drivers

In section 6.2, the research methodology followed for the development of the software obsolescence cost analysis framework was presented. It included a detailed description of the methodology followed in each section required for the development of the framework.

In section 6.3, the detailed research methodology was presented, and the author described the requirement set by industrial experts for the development of this framework. In this section the author went in detail about each section of the framework capturing the requirement, research methodology, refinement of the findings and the validation of the findings.

In section 6.5, the researcher described the iterative process followed for the development of the software obsolescence cost analysis framework and indicated the resulting enhancements.

The next chapter presents the development of the Software Obsolescence Cost Analysis Tool that can be used to at the bidding stage of a support contract to estimate the resolution cost of software obsolescence during the contracted period.

7 Software Obsolescence Cost Analysis Model

7.1 Introduction

In the previous chapter, the researcher described the development of the software obsolescence cost analysis framework. The key focus of this framework is on identifying, managing, and mitigating software obsolescence. There exist several models to predict the cost of EEE component and Material obsolescence such as EEE- Force and M- Force (Javier Romero Rojo, 2011), however there is currently no such model available to estimate the cost of software obsolescence resolution.

The literature review in chapter two, has confirmed the lack of understanding around software obsolescence and cost estimation of software obsolescence resolution as part of support contracts. Moreover, the capture of current practice within UK Defence and Aerospace as explained in chapter four and five, has revealed that the responsibility for managing, planning and resolution of software obsolescence issues are being passed to the prime contractor or OEM and industry collaborations. This has created a new requirement to predict the cost of software obsolescence resolution at early stages of the project life cycle so as to include the cost within the support contract. In this chapter the researcher presents the development of the “Software Obsolescence Resolution Cost Analysis Model” that can be used at any stage of a support contract to estimate the software obsolescence resolution cost incurred during the contracted period.

One of the key challenges of estimating the software obsolescence resolution cost in majority of cases, is the lack of a defined resolution approach in addition to a considerable lack of historic or current data making it difficult to tackle a particular software obsolescence issue that cannot be predicted in advance. Due to these reasons, the “software obsolescence resolution approach study” was undertaken to determine the probability of using any given software obsolescence resolution approach to tackle a software obsolescence issue, based on the experience of industrial experts in software obsolescence. The key element in the estimation process is a good understanding of software

obsolescence cost drivers. As it was explained in chapter 2 and chapter 4, there is no existing software obsolescence cost matrix, and therefore became apparent that a valid cost matrix for software obsolescence resolution needed to be developed and validated by industry experts. To address this issue a “Software Obsolescence Key cost driver” study was undertaken.

All the above work streams paved the way for the development of the “Software obsolescence cost analysis model”. This chapter provides an understanding about software obsolescence resolution cost and the methods involved in the prediction/ estimation of the cost. This chapter also looks at the optimisation of the cost based on the best available resolution.

Next section provides a description of the research methodology applied to capture the key cost drivers and its implication within the cost model. In section 7.3 the researcher provides description of the Data Collection Model. By going through the steps taken to build the model in addition to the details on how it's been used within the Resolution Cost Model. In section 7.4 the researcher provides descriptions of the Software Obsolescence Resolution Model, which includes a diagrammatic representation of the resolution approach undertaken. And in the final section of this chapter the researcher provides a description of the cost model that estimates the cost of software obsolescence resolution. A description about the logic developed for the cost estimation approach is also provided.

7.2 Detailed Research Methodology

The research methodology followed for the development of the software obsolescence cost analysis model is shown in Figure 7-1. The first step was to gain a better understanding about software obsolescence, how it is managed currently within the Defence and aerospace industry and the level of information available at the bidding/ early stages of the project life cycle. The next step was to develop a framework consisting of key components such as Software obsolescence Management level, Complexity matrix, key cost drivers and software obsolescence resolution profiles. The Software Obsolescence Cost Analysis Model (SOCAM) was developed following an iterative process which

included experts from industry reviewing it and advising on any enhancements that were deemed necessary and was implemented accordingly. The SOCAM looked at the resolution cost of software obsolescence issue. These key components of the framework provided both qualitative and quantitative data required to complete the cost estimation model.

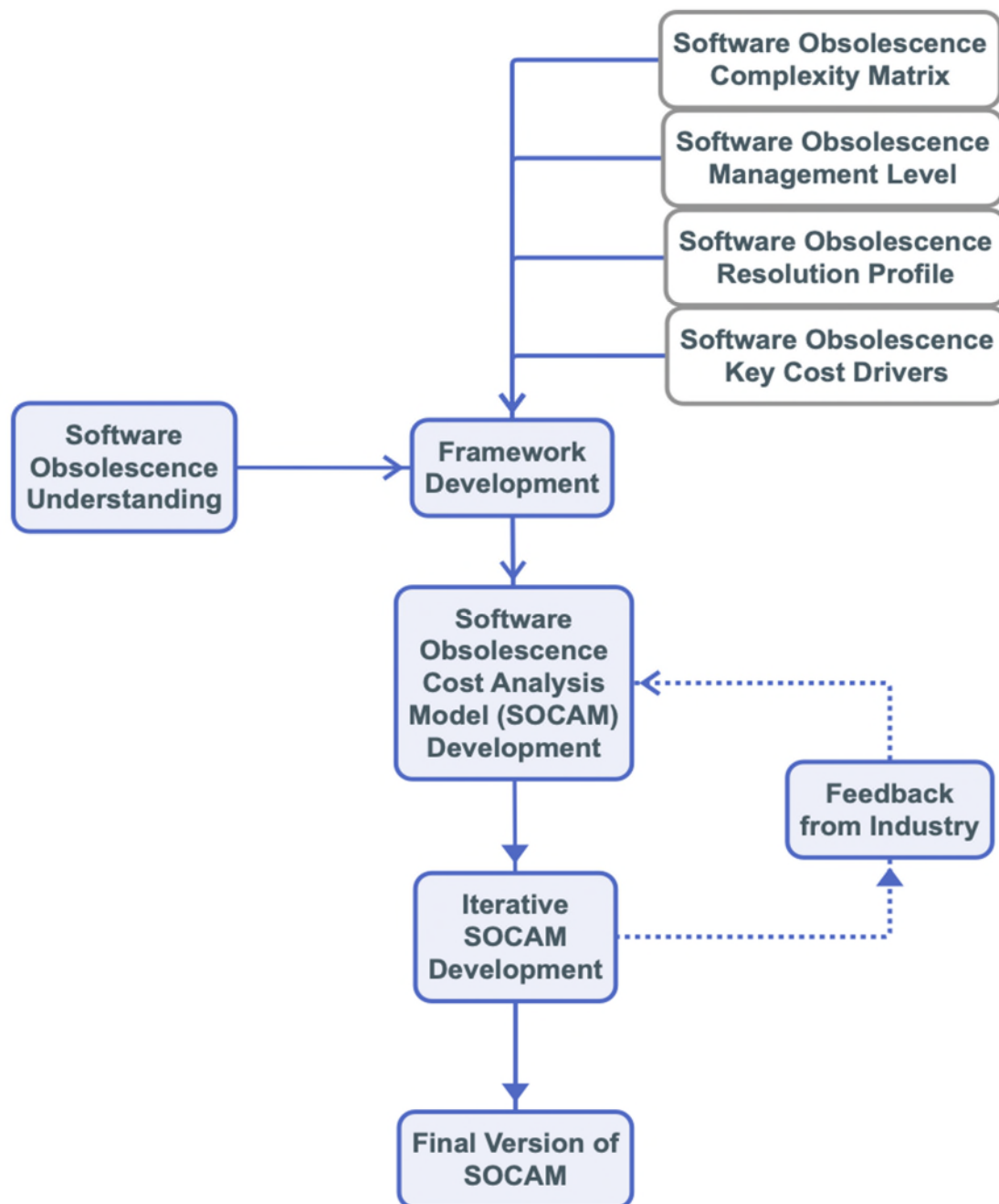


Figure 7-1 Research methodology followed for the development of SOCAM

The initial requirement for Software Cost Analysis framework and the cost model was captured during interviews and case studies with experts from Defence and Aerospace industry. Based on those requirements, a draft version of the framework and the model was developed and presented at a workshop which included five SMEs from several Defence and aerospace organisations. Although all of those experts were obsolescence specialists, however lacked expertise in software obsolescence. Table 7-1 below shows the details of the participants.

The purpose of this workshop was to refine the initial requirement, identify other requirements and validate the rationale proposed for the framework and the cost model. The workshop duration was for half a day during which the draft version of the framework and the cost model was presented and following a discussion from the experts they filled in questionnaire to assess any further enhancement requirement as shown in appendix A (Question set for validation). An example of the questions contained within the questionnaire is provided below.

- Would a Software Obsolescence Cost Analysis Framework and cost model be beneficial for the Defence and aerospace industry at the bidding stage?
- Assess the relevance of the presented framework and cost model to your business
- Assess the benefits of the presented framework and cost model to industry
- Indicate improvement required and prioritise them
- Assess the validity of the assumptions made

The responses were then analysed by identifying similarities, differences, and any unique responses across the different participants. The outcomes from this analysis are summarised further along in this chapter to show the process through which the cost model was developed.

Table 7-1 Details of the participants

	Organisati on	Role	Years of Experi ence	Obsolescence Expert	Software Obsolescence Expert
1	ORG A	Programme Manager	25	Yes	No
2	ORG A	Obsolescence Manager	15	Yes	No
3	ORG B	Cost Analyst	13	No	No
4	ORG C	Systems Engineer	15	Yes	No
5	ORG D	Software Engineer	20	No	No but understands Software Obsolescence

7.3 Data Collection Model

This section describes the process involved in the development of the data collection model. The data collection model forms the key part of this tool set, the development of which was undertaken iteratively with validation and following feedback from experts within Defence and aerospace industry. The development of the data collection model is described in the Figure 7-2 below.

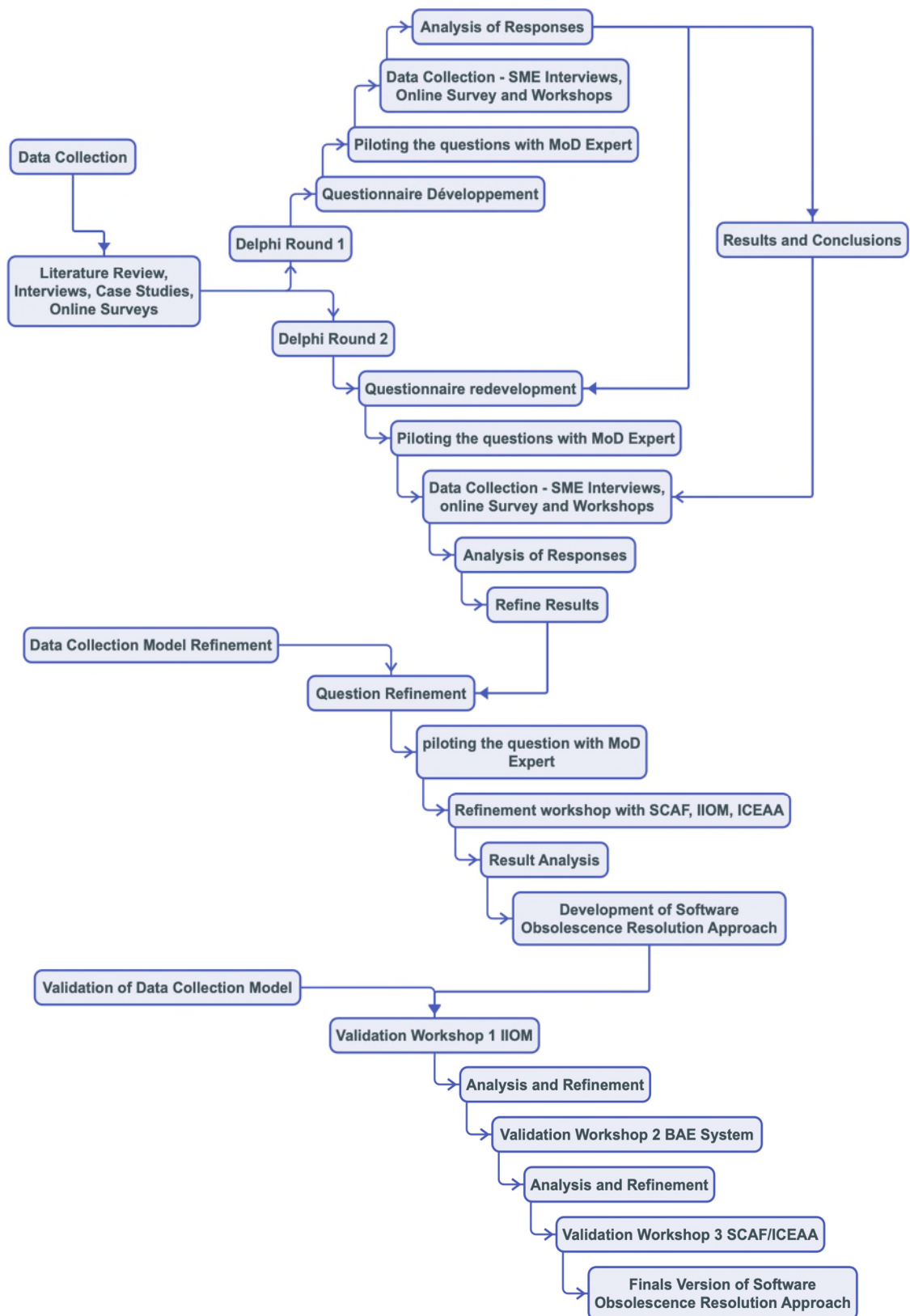


Figure 7-2 Research Methodology followed to develop Software Obsolescence Data Collection Model

7.3.1 Phase 1 Data Collection

The whole process of development of the data management model is divided into three phases. The first phase is the data collection phase, and this involved detailed literature review where initial understanding of types of question that needed to be answered was obtained, which was then complimented by information obtained through case studies, expert interviews, and online surveys. This knowledge was then tested with the experts from Defence and aerospace sectors using the Delphi methodology.

The Delphi method is an “iterative process to collect and refine the anonymous judgement of experts using a series of data collection and analysis methods followed by a good feedback loop” (Skulmoski et al., 2007). Delphi is well suited for exploratory studies such as the development of software obsolescence management level, where there is a lack of appropriate historical data (Okoli and Pawlowski, 2004; Linstone and Turoff, 1975) and expert judgement is the only option (Rowe and Wright, 1999).

A questionnaire was developed to be used in the first round of data collection via SME interviews, IIOM, and SCAF Workshops. These questions were piloted with MoD Obsolescence experts for validation and verification to ensure the questions were clear and unambiguous. The participants were asked about their experience level, combined with the level of software obsolescence management being currently undertaken at organisational level or at the project level and finally the question set covered the likelihood of them using the initially developed software obsolescence management level. Keeney et al (1988) supported the use of probability to quantify expert judgement in validating a complex technical and engineering problems. A total of 15 obsolescence experts with various degrees of software obsolescence management experience (please note none of them are software obsolescence experts but had to deal with emerging issues with software obsolescence by either passing the risk to OEM or reactively dealing with software obsolescence issues) participated in the first round of the data collection, and their responses were subsequently analysed. Table 6-2 shows the years of experience of participants who took part in these workshops.

The outcome of this analysis was presented at a second round of workshop to 10 obsolescence experts with various level of software obsolescence management interest. 50% of the participants were from the first round of workshops. They took the opportunity to discuss about the results either by corroborating the results or amending them. A final analysis of these responses was allowed to refine the outcome of the first round. The results of the first and second round converged hence no further iterations on the Delphi was needed.

7.3.2 Phase 2 –Data Collection Model Refinement

The second phase of this study investigated the refinement of the Data Collection Model. In this phase the draft version of the Data management model was developed which was then used at a refinement workshop with 15 experts from the Defence and aerospace industry via SCAF, IIOM and ICEAA Workshops. An updated question set was developed to undertake the refinement exercise. This workshop played an important role in defining a common understanding of different aspects of the data collection model such as Project data, system data, and software support data. The responses were collected and analysed, and refinement of the data collection model from the Delphi study was applied.

7.3.3 Phase 3 – Data Collection Model Validation

The objective of this phase was to validate the contents of the data collection model by checking their plausibility (Robson, 2002) and making sure that the contents of each section of the model was clear and logical. The expert discussed each content of the Data collection model and concluded whether it complemented their anticipated trend based on their experiences and justifying them.

The first two validation workshop was attended by 15 Obsolescence Experts from IIOM, and BAE System with the third workshop being attended by 5 cost experts from ICEAA and SCAF. The experts agreed that the Data collection model was being pitched at the right level.

The data collection model was sent to the participants few weeks prior to the validation workshop and the researcher was made available to answer any follow up questions. This was a half a day workshop where the data collection model was presented to the attendees and feedback collected.

The software obsolescence data collection activity consisted of five sections which are listed below. This model could be used on its own as a data collection model for collecting software obsolescence data to monitor/manage software obsolescence issue or use this model as an input to the Software Obsolescence Cost Analysis Framework. The framework produced various outputs such as the resolution profile, resolution approach, Obsolescence Management level and or Software Obsolescence Complexity level.

- Project data
- Systems Data
- Software Data
 - o Software sizing data
 - o Software Development data
 - o Software application data
- Schedule Data and
- Software Obsolescence Data
 - o Software Obsolescence Management Data
 - o Software Obsolescence Resolution Data
 - o Software Obsolescence Complexity Data
 - o Software Obsolescence Probability Matrix

The data collection activity was one of the key parts of the framework due to its dual purpose. The first one being the assistance it provides to the cost estimating side of the framework and the second being its usage as a tool to collect data on software obsolescence. Such data collection/ monitoring tool is not currently available, which was established in Chapter 2 through extensive literature reviews, and in Chapter 4 on the current state of the software obsolescence management.

This question set was developed following extensive literature reviews, case studies, online surveys, expert interviews, and cognitive case studies.

Every data that these questions collected was directly related to the output of the cost model, and to the output of the different sections of the Framework. For example, output from this data collection model determine the Software Obsolescence Management Level; it also helps in identifying the appropriate resolution approach to be taken and the complexity of the software obsolescence.

Data Collection Model – Project Data

In this section, the model asks general questions about the project such as the name of the project, project start and end date, the date of initial operating capability and the type of platform this project relates to. Figure 7-3 shows these sections in detail.

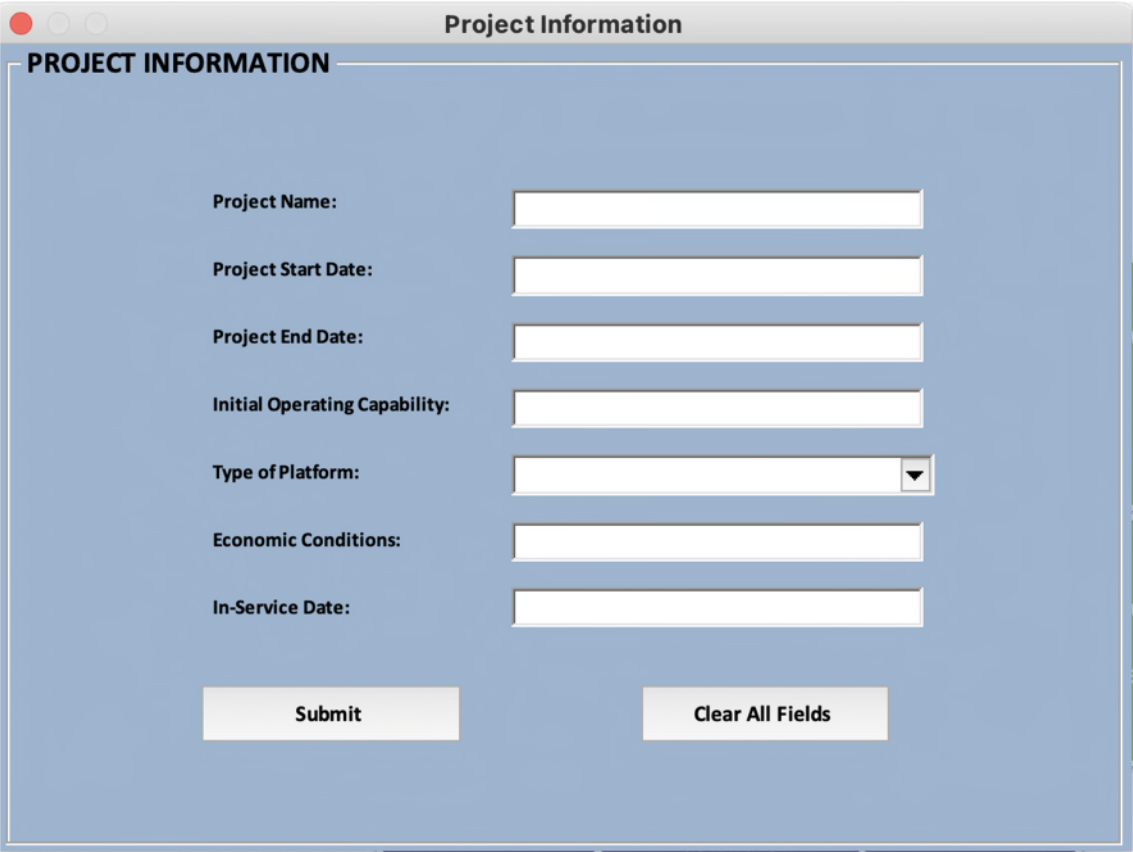
The image shows a software window titled "Project Information". Inside the window, there is a section header "PROJECT INFORMATION". Below this header, there are seven input fields arranged vertically. The first six are text boxes, and the seventh is a dropdown menu. The labels for these fields are: "Project Name:", "Project Start Date:", "Project End Date:", "Initial Operating Capability:", "Type of Platform:", "Economic Conditions:", and "In-Service Date:". At the bottom of the form, there are two buttons: "Submit" on the left and "Clear All Fields" on the right. The entire form is set against a light blue background within the window.

Figure 7-3 Project data Form

Data Collection Model – Systems Information Data

In this section of Data collection model, questions are asked about the system the software is part of, such as project complexity, software languages, software compatibility, software compilers used and the number of applications etc.

This information is useful in determining the software obsolescence complexity matrix and the resolution profile. The detailed question set is listed in the Figure 7-4 below.

Systems Information

SYSTEM INFORMATION

Number(s) of applications per system:

Software language:

What design tools/compilers have been used?

Is the compiler compatible with next hardware and software updates?

Is the software being developed at more than one site?

How many national sites?

How many International sites?

Project complexity:

What is the complexity of Software to Software integrations?

What is the complexity of Software to Hardware integrations?

Figure 7-4 System Information Form

Data Collection Model – Software Data

This is a very important section in this model as it collects information on the software as a whole and its components. This section is key as during the research it has been identified that there is no or very limited data available on

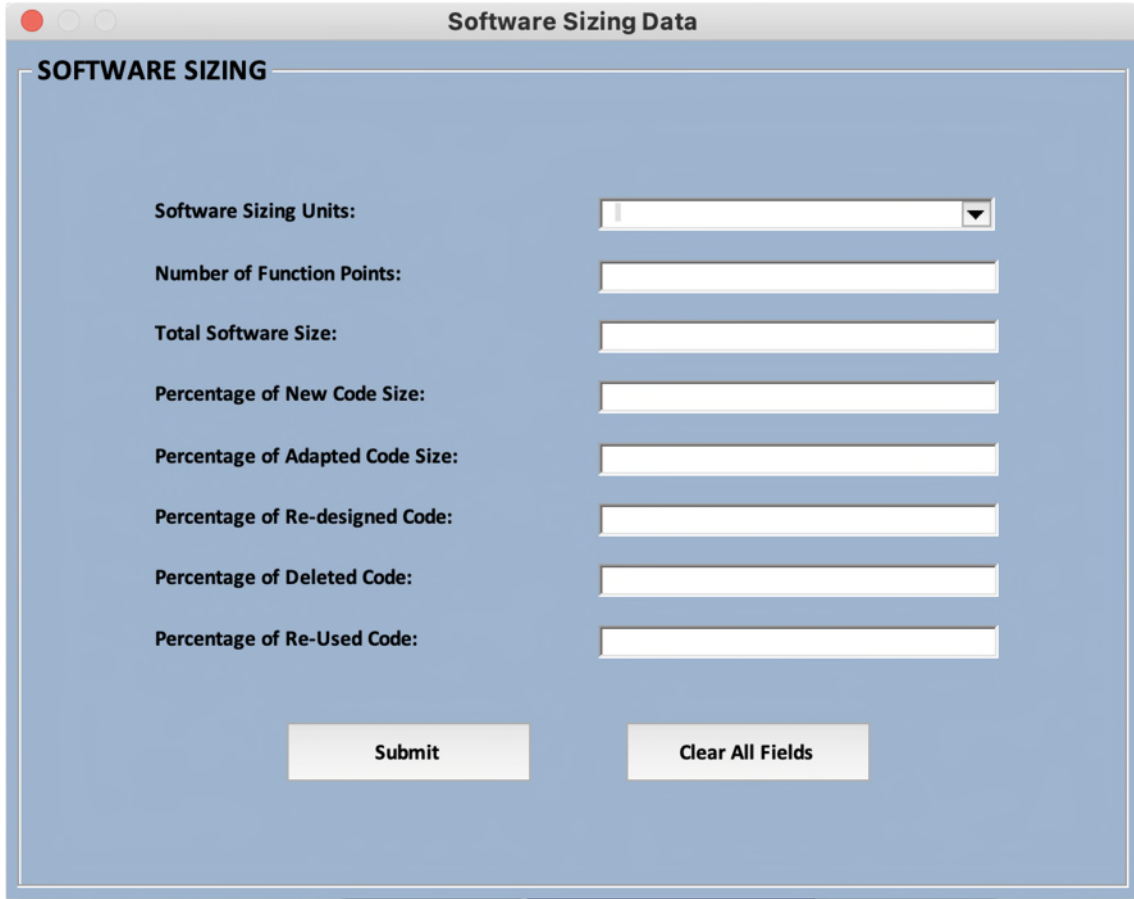
the software obsolescence, this section is key to teeth out potential data that is required to manage/understand software obsolescence. This section of the model is in three parts.

Software sizing Data

This information is vital depending on how mature the project is and when the cost estimates are required. The required information would potentially be readily available if the project is in the in-service phase, however at an early stage of the project, finding this information could be extremely difficult and several assumptions would need to be in place.

In this section, the information collected are largely on the construct of the software. i.e., the size of the software. Software sizing is a key cost driver in the Software Estimating process and it directly co-relates to the effort required to develop/ maintain the software. This is also true in the case of effort requirement to update/upgrade or re-write the software if there is a software obsolescence issue. This would force the project team to choose the appropriate resolution approach as dictated by the Software Obsolescence Framework.

The question set (Figure 7-5 below) is largely centred around- the standard used to measure the size such as SLOC (Source lines of Code) or FP (Function Points), percentage of new code developed, adapted, re-designed, deleted and re-used etc.



The image shows a software window titled "Software Sizing Data". Inside the window, there is a section titled "SOFTWARE SIZING". Below this title, there are eight input fields arranged in two columns. The first column contains labels for each field: "Software Sizing Units:", "Number of Function Points:", "Total Software Size:", "Percentage of New Code Size:", "Percentage of Adapted Code Size:", "Percentage of Re-designed Code:", "Percentage of Deleted Code:", and "Percentage of Re-Used Code:". The second column contains the corresponding input controls: a dropdown menu for "Software Sizing Units:", and seven text input boxes for the other labels. At the bottom of the form, there are two buttons: "Submit" on the left and "Clear All Fields" on the right.

Label	Input Type
Software Sizing Units:	Dropdown menu
Number of Function Points:	Text input box
Total Software Size:	Text input box
Percentage of New Code Size:	Text input box
Percentage of Adapted Code Size:	Text input box
Percentage of Re-designed Code:	Text input box
Percentage of Deleted Code:	Text input box
Percentage of Re-Used Code:	Text input box

Buttons: Submit, Clear All Fields

Figure 7-5 Software sizing form

Software Development Data

The information collected in this section directly correlates to the cognitive case study findings that was discussed in Chapter five, primarily around software developers identifying potential software obsolescence issues that could happen in the future. This was possible due to the internal and external factors that influenced the current software development activities such as the standards that the software was required to develop, the legal requirement, current company policies on technology selections, security level, external and internal integration of the software etc.

This information helped to identify the appropriate resolution approach, software obsolescence complexity and the software obsolescence management level that formed the software obsolescence framework. This in turn helped in the identification of level of resource and effort requirement to mitigate the potential

software obsolescence issue/s. This question set is detailed in the Figure 7-6 below.

Software Development Data

SOFTWARE DEVELOPMENT

Software Development Process:

Number of Spirals or Increments:

External Integration Level:

Internal Integration Level:

Level of Software Reuse:

What is the Software Development Security Level?

What is the Contractors/Developers CMMI Level?

Describe the "Cohesiveness" of the Software Development Team:

Does your hardware impose memory constraints?

Does your hardware impose performance constraints?

Assess the capability of the software analysts and designers assigned to this effort:

Assess the capability of the programmers assigned to this effort:

Assess the capability of the integration team assigned to this effort:

How much experience does the programming team have with the language?

How much experience does the software development team have with this type of product?

How much experience does the software development team have with the development platform?

What is the company's average annual turnover rate for its software development teams?

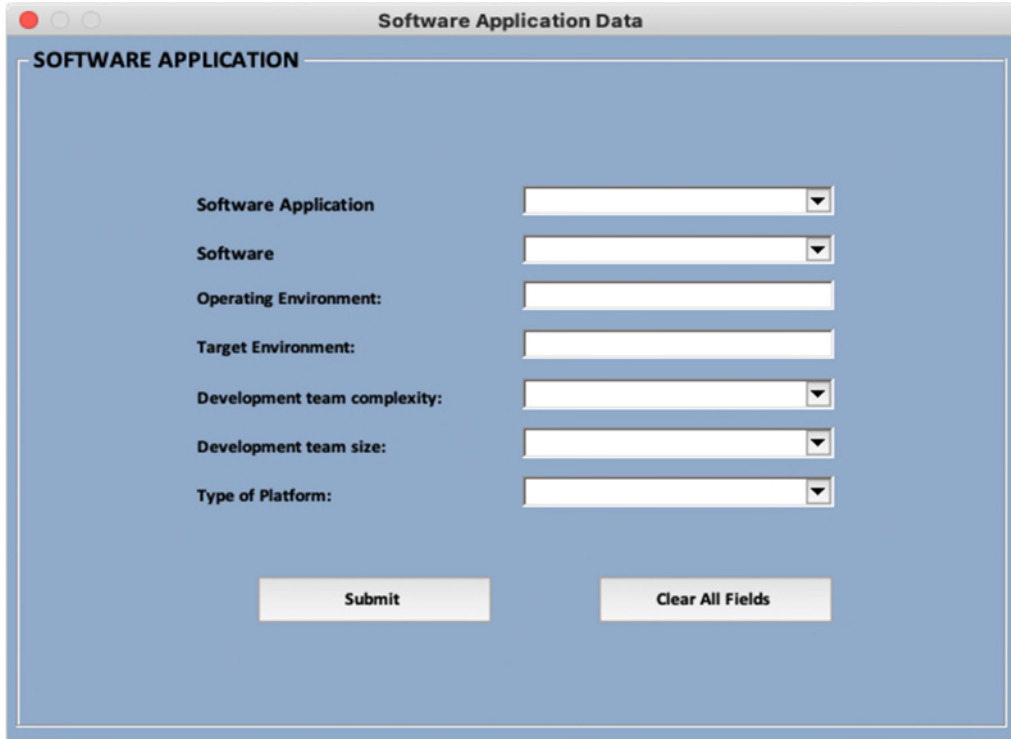
Submit

Clear All Fields

Figure 7-6 software development data form

Software Application Data

In this section, the information collected is related to the target and operating environment in which the software operates. This is a key piece of information as per the definition of software obsolescence. The data therefore collected can be used to define the Software Complexity and the software obsolescence resolution approach. This question set is listed in the Figure 7-7 below.

A screenshot of a web form titled "Software Application Data". The form has a blue background and a white border. It contains seven input fields, each with a label and a dropdown arrow. The labels are: "Software Application", "Software", "Operating Environment:", "Target Environment:", "Development team complexity:", "Development team size:", and "Type of Platform:". Below the input fields are two buttons: "Submit" and "Clear All Fields".

SOFTWARE APPLICATION	
Software Application	
Software	
Operating Environment:	
Target Environment:	
Development team complexity:	
Development team size:	
Type of Platform:	
Submit Clear All Fields	

Figure 7-7 Software Application Data form

Data Collection Model – Project Schedule Data

This section is all about collecting information on the project start date, start of the support contract and any midlife update/upgrade of the Software and hardware.

This information is used to predict the potential software obsolescence issue happening and to help plan software obsolescence strategy, software obsolescence resolution approach and obsolescence profile. The Figure 7-8 below shows the questions that are used in this section.

Project Schedule Data

PROJECT SCHEDULE

Start date of Support Contract:

Termination date of Support Contract:

Date of any planned Software Mid-life Update/Upgrade:

Date of any planned Hardware update/upgrade:

Date of any planned changes to the technical capabilities:

Figure 7-8 Project schedule data form

Data Collection Model – Software Obsolescence Data

This part of the model focusses on the data collection questions which are directly related to Software Obsolescence Cost Analysis Framework/ Model. This section covers the questions that are used to determine the appropriate software obsolescence management level, software obsolescence resolution approach, software obsolescence probability matrix and also helps in the identification of software obsolescence complexity.

Software Obsolescence Management Data

This part of the model tends to ask specific questions related to how well the project team/ or the company deals with software obsolescence. The questions are largely focused on the management strategy such as whether the company/project team have any Obsolescence Management strategy, and/or if they have any software obsolescence management strategy. This section also enquires about how they tend to deal with software obsolescence for example using reactive or proactive management. It also probes into the standard the company uses such as CMMI and provides options to select the appropriate

software obsolescence strategy. The strategy list was compiled as part of the literature search, online survey, and expert interviews. This list is detailed below in Figure 7-9 and the question set in Figure 7-10.

Considering Software Obsolescence at the design and development phases

Continuous Upgrade/Migration

Deal with Software Obsolescence Reactively

Deploy Software Obsolescence Monitoring process or tool if available

Employ Software Obsolescence professional

Escrow Agreement in place or third-party partnership in place

Freeze and do nothing

Maintain appropriate skills in house

Mid-life upgrade and migration

Monitoring Software Supply chain (Skills and technology Insertion)

Monitoring Technology insertion

Notify Software Obsolescence proactively

Reactive but dealing with software obsolescence by reverse engineering and code conversion

Figure 7-9 Questions on software obsolescence management level

SOFTWARE OBSOLESCENCE MANAGEMENT LEVEL

Do you have an Obsolescence Management Strategy?

Do you have a Software Obsolescence Management Strategy?

What is your Software Obsolescence Management Strategy?

What is your CMMI Level?

What is your TOMCAT Score?

Please select all the Software Obsolescence Strategy's that are applicable to your project:

- ☐ Deploy Software Obsolescence Monitoring pr
- ☐ Employ Software Obsolescence professional
- ☐ Escrow Agreement in place or third party pe
- ☐ Freeze and do nothing
- ☐ Maintain appropriate skills in house
- ☒ Mid life upgrade and migration
- ☐ Monitoring Software Supply chain (Skills and
- ☐ Monitoring Technology insertion
- ☐ Notify Software Obsolescence proactively
- ☐ Reactive but dealing with software obsolesc

Submit Clear All Fields

Figure 7-10 Software Obsolescence management level inputs form

Software Obsolescence Resolution Data

In this section, the user is required to refer to the Software Obsolescence Resolution approach from the framework and input the appropriate resolution approach (Functional, Logistical and Technological Resolution) to each software component. User can have more than one resolution approach to a single component over the life cycle of the software project.

The data will help in determining the appropriate resolution approach that is required to mitigate the obsolescence issue and helps in understanding the resolution profile of each component. The Figure 7-11 below shows the data collection form.

Components	0 - 5 years	6 - 10 years	11 - 15 years	16 - 20 years
	Functional Resolution Technological Resolution Logistical Resolution			

Submit Clear All Fields

Figure 7-11 Software Obsolescence resolution form

Software Obsolescence Probability Matrix

In this section of the model, each component of the software is scrutinised to obtain the data required to successfully predict the probability of the software going obsolete. This section is key as it plays an important function in determining the appropriate resolution approach that is needed to resolve the predicted software obsolescence issue/s. These questions were generated from the findings of the cognitive case study as described in chapter 5. Figure 7-12 below shows the data collection form

The questions are largely based on the following

- Version history
- Documentation
- Forward and backward compatibility
- Supportability

Software Obsolescence Probability Matrix Input

SOFTWARE OBSOLESCENCE PROBABILITY

Component 1 | Component 2 | Component 3 | Component 4 | Component 5

What is the current version of the application?

When does this version expire?

When is the next update due?

Have you got the correct documentation for your software?

Do you have the capacity to modify the software?

Do you have the capability to modify the software?

Is the source repository maintained?

Is the development tool

Are there any third party components in the

Is there any third party support available?

Are you able to compile the third party components?

What is the End of Life date for this application?

Is there a planned replacement product?

Is the planned replacement product backward compatible?

Is there a change in the target environment?

Is there a change in the operating environment?

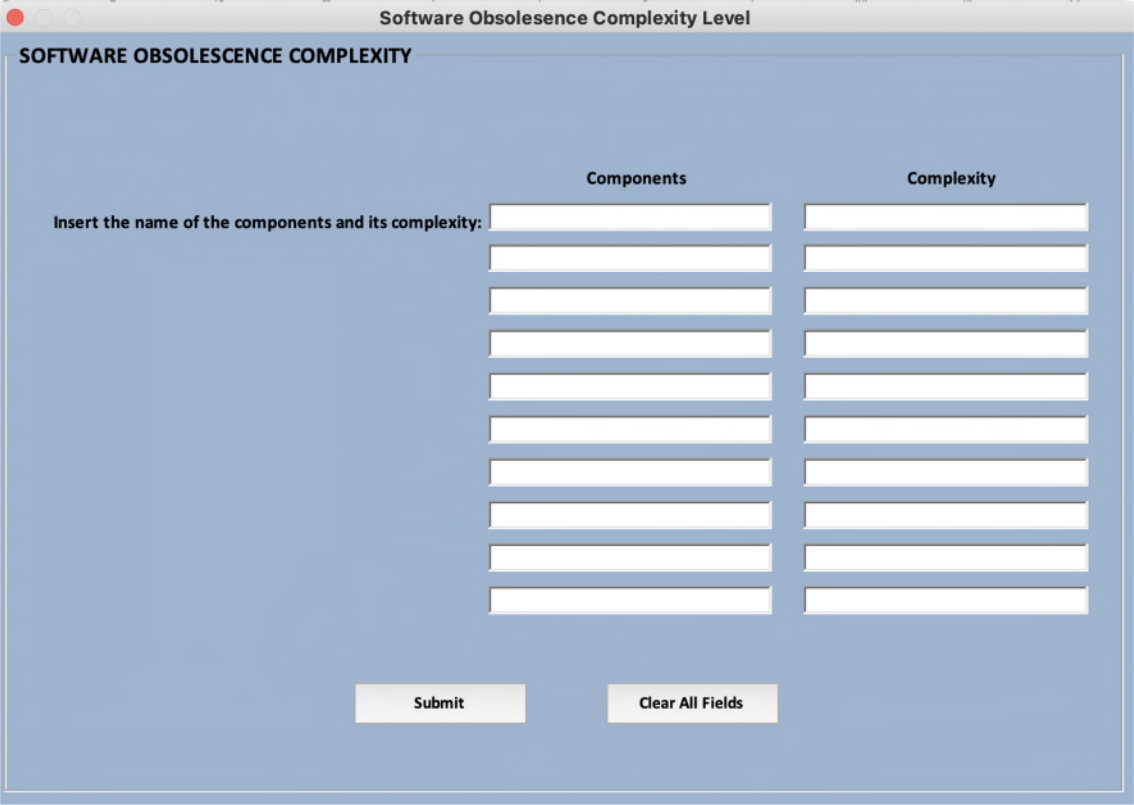
Submit Clear All Fields Clear Component

Figure 7-12 Software Obsolescence probability Matrix form

Software Obsolescence Complexity Data

This section is the last part of the Software Obsolescence Data Collection model. This part of the model is optional as the data that is needed to determine the complexity matrix could be identified through other data forms if the specific data to complete this section was not available. In the section, the user is advised to use the Software Obsolescence Complexity Matrix from the framework as reference to input the data. The user is required to fill in the complexity for each component.

This data helps in understanding the resource and effort requirement to resolve the software obsolescence issue. The data form is shown in Figure 7-13 below.



The image shows a web application window titled "Software Obsolescence Complexity Level". Inside the window, there is a section titled "SOFTWARE OBSOLESCENCE COMPLEXITY". Below this title, there is a label "Insert the name of the components and its complexity:" followed by two columns of input fields. The first column is labeled "Components" and the second column is labeled "Complexity". Each column contains ten empty text input fields. At the bottom of the form, there are two buttons: "Submit" and "Clear All Fields".

Components	Complexity

Figure 7-13 Software obsolescence complexity level form

Each section of the Data Collection Model is directly related to the output that is generated through the software obsolescence cost analysis framework. The relationship between each input sheet and the output is shown in the Figure 7-14 below. The relationship between the inputs and outputs was designed and developed following the case study and expert interview analysis. This was then validated as part of the framework validation which is discussed further in the verification and validation chapter.

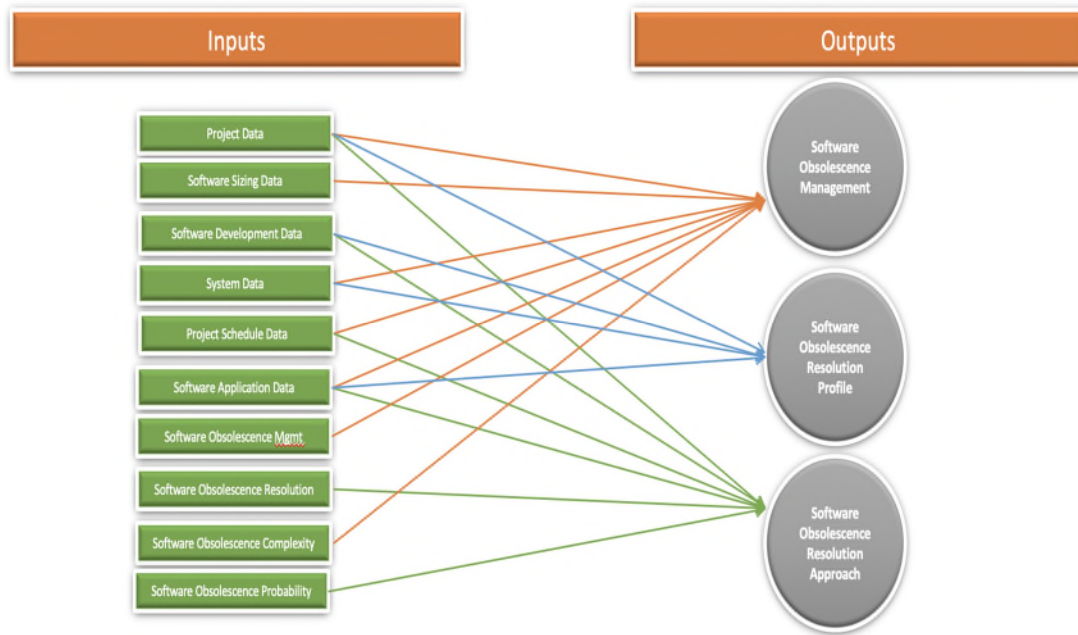


Figure 7-14 Inputs and output relationships

7.4 Software Obsolescence Resolution Model

The researcher, in this section describes the process involved in the development of the Software Obsolescence Resolution Model. This model forms the key part of this tool set. The development of the Software Obsolescence Resolution model was undertaken iteratively following validation and feedback from the experts in the Defence and aerospace industry. The development of this model is described in the Figure 7-15 below.



Figure 7-15 Research methodology used to develop Software Obsolescence Resolution Model

7.4.1 Phase 1 Data Collection

The whole process of development of Software Obsolescence is divided into three phases. The first phase is the data collection phase, and this involves excessive literature review where initial understanding of types of question that needs to be answered was obtained, which was then complimented by the information obtained through case studies, expert interviews, and online surveys. This knowledge was then tested with the experts from Defence and aerospace sectors using a Delphi methodology.

The Delphi method is an “iterative process to collect and refine the anonymous judgement of experts using a series of data collection and analysis methods followed by a good feedback loop” (Skulmoski et al., 2007). Delphi is well suited to exploratory studies such as the development of the software obsolescence management level where there is a lack of appropriate historical data (Okoli and Pawlowski, 2004; Linstone and Turoff, 1975) and expert judgement are the only option (Rowe and Wright, 1999).

A questionnaire was developed to be used in the first round of the data collection with SME interviews, IIOM, and SCAF Workshops. These questions were piloted with MoD Obsolescence expert for validation and verification to make the question clear and unambiguous. The participants were asked about their experience level, combined with the level of software obsolescence management been currently undertaken at organisational level or at the project level and finally the question set covered the likelihood of them using the initially developed software obsolescence management level. Keeney et al (1988) supported the use of probability to quantify expert judgement in validating a complex technical and engineering problems. A total of 15 obsolescence experts with various degrees of software obsolescence management experience (please note none of them were software obsolescence experts but had to deal with emerging issues with software obsolescence by either passing the risk to OEM or reactively dealing with software obsolescence issues) participated in the first round of the data collection, and their responses were subsequently analysed. Table 6-2 shows the years of experience of participants who took part in this workshop. The

outcome of this analysis was presented at a second round of workshop to 10 obsolescence experts with various levels of software obsolescence management interests. 50% of the participants were from the first round of workshops. They took the opportunity to discuss about the results either by corroborating the results or amending them. A final analysis of these responses was allowed to refine the outcome of the first round providing as a result, a model for the resolution approach. The results of the first and second round converged and therefore no further iterations on the Delphi was needed.

7.4.2 Phase 2 –Software Obsolescence Resolution Model Refinement

The second phase of this study investigated the refinement of the Data Collection Model. In this phase a draft version of the Software Obsolescence Resolution Model was developed which was then tested with 15 experts from within Defence and aerospace industry, at the SCAF, IIOM and ICEAA Workshops. An updated question set was developed to undertake the refinement exercises. This workshop played an important role in defining a common understanding of different aspects of the Software Obsolescence Resolution model. The responses were collected and analysed, and the mean was applied for the refinement of the data collection model from the Delphi study.

7.4.3 Phase 3 – Software Obsolescence Resolution Model Validation

The objective of this phase was to validate the contents of this model by checking their plausibility (Robson, 2002) and ensuring that the contents of each section of the model was clear and logical. The experts discussed each content of the Software Obsolescence Resolution and concluded whether it complemented their anticipated trend based on their experiences and justified them.

The first two validation workshops were attended by 15 Obsolescence Experts from IIOM, and BAE System with the third workshop being attended by 5 cost experts from ICEAA and SCAF. The experts agreed that the Software Obsolescence model was being pitched at the right level.

Software Obsolescence Resolution Model is the diagrammatical representation of the resolution approach undertaken by the project team. The input for this model was automatically generated from the Data Collection Model. This model was generated, given the challenges to build a traditional cost model due too many unknowns and missing data points. Therefore, the researcher has used a methodology which in this case breaks down the problem into smaller chunks making it easier to manage. This methodology involved diagrammatical representation of the cost build up following the selection of Software Obsolescence Resolution Approaches as part of the mitigation actions. This method was easy to understand where, how and which resolution approach to be used and subsequent cost giving the user a real time view.

This model represents decomposition of the cost from one output node through successive levels to individual input nodes. The type of nodes which are colour coded are detailed in the Figure 7-16 below. This decomposition is the key focus of the diagram and the techniques. Once the build-up of this diagram is complete, it will be converted into a set of inputs. Each node will have a Min, ML and Max inputs (provided there is sufficient data) to run Monte Carlo Simulation to generate an interval of uncertainty for cost.

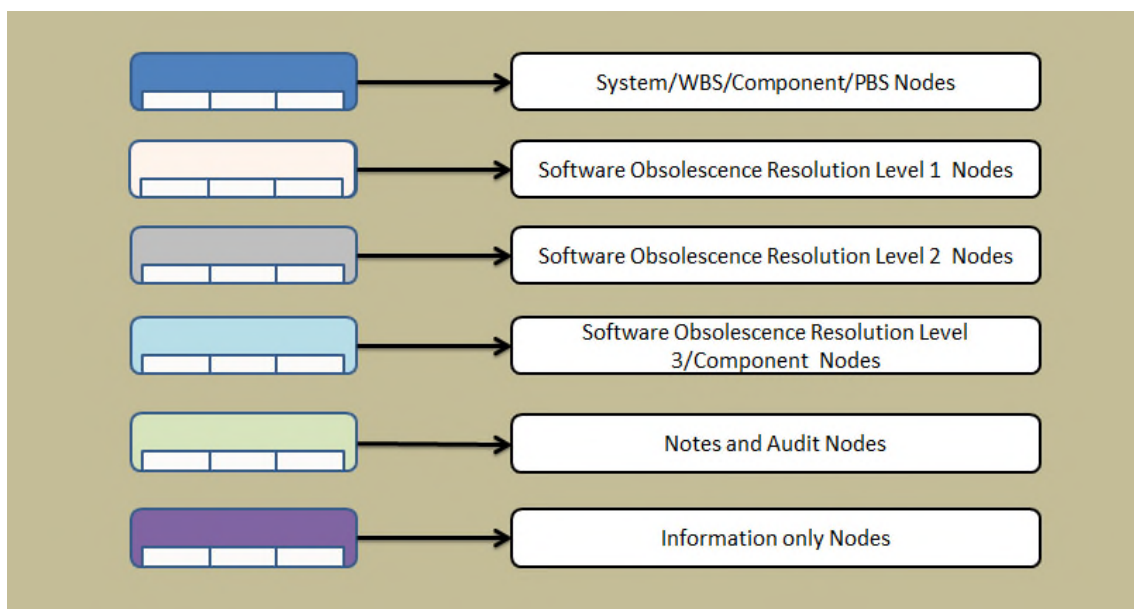


Figure 7-16 Types of nodes used in the resolution model

Each software has several components, and each component undergoes different obsolescence resolution approaches which can be diagrammatically represented using the above methodology. The Figure 7-17 below shows an example of level one decomposition. The diagram represents the decomposition of cost from output nodes (on the left) through successive levels to individual input nodes (on the right) and this decomposition is the key to this diagram and technique.

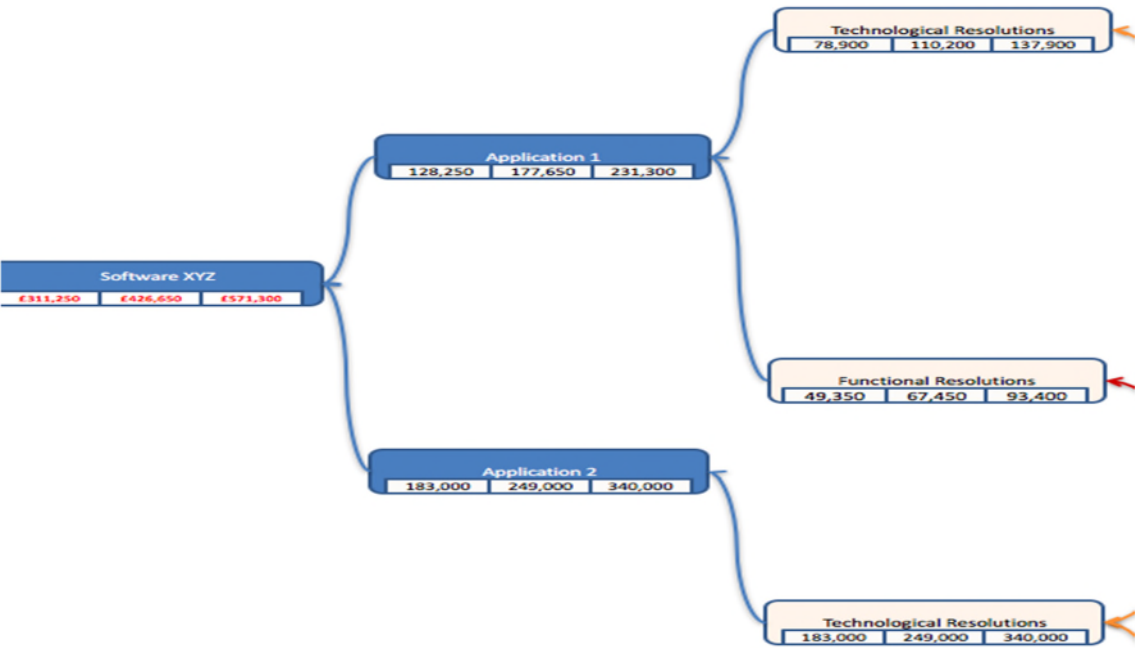


Figure 7-17 Example of level one resolution decomposition

Input values are entered on the right of the diagram and aggregated through the model to give the vignette cost. Each node had a specific attribute to combine inputs, which are defined in the diagram. In this methodology each module node allows repeated use of common elements, such as software engineers rates etc. Figure 7-18 shows further decomposition of the nodes.

This process is repeated until the whole of the software component over the years of support contract are represented diagrammatically to give the whole software obsolescence resolution profile. Figure 7-19 shows an example case of the whole resolution model. The decomposition is complete once the project teams identify

the amount and type of resources that are required to complete individual resolution approaches for each component. Once the decomposition is complete, the values in the diagrams are converted into a set of inputs to conduct the calculations within the cost model.

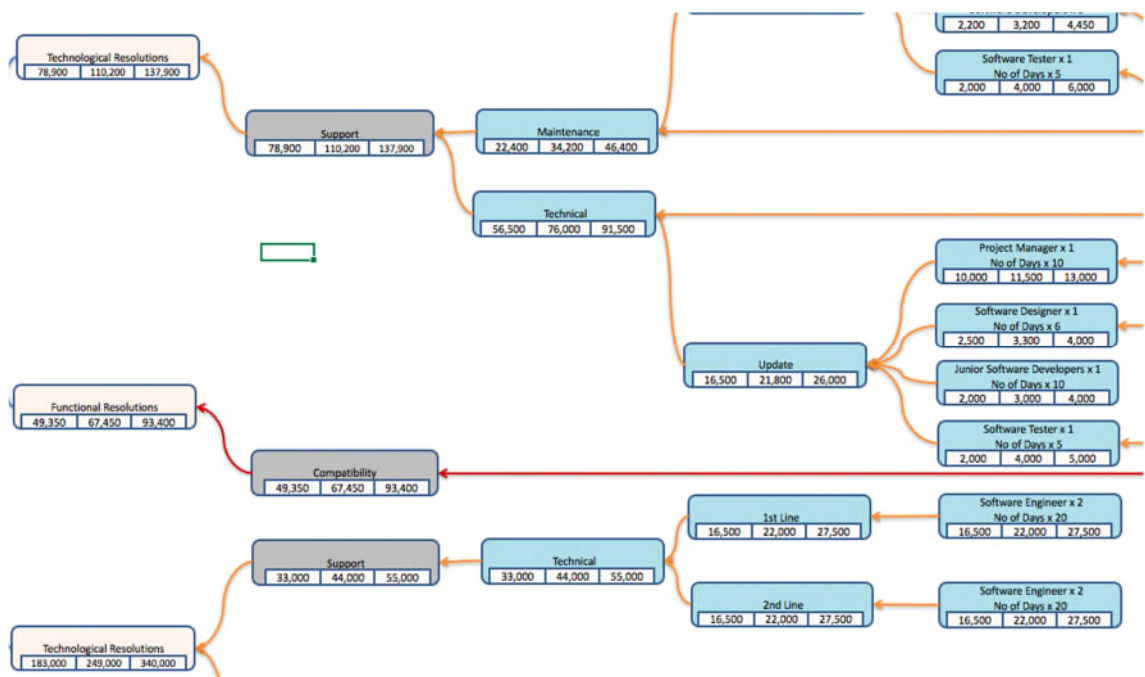


Figure 7-18 Shows further decomposition of the nodes.

Software Obsolescence Cost Model/ Resolution Profile

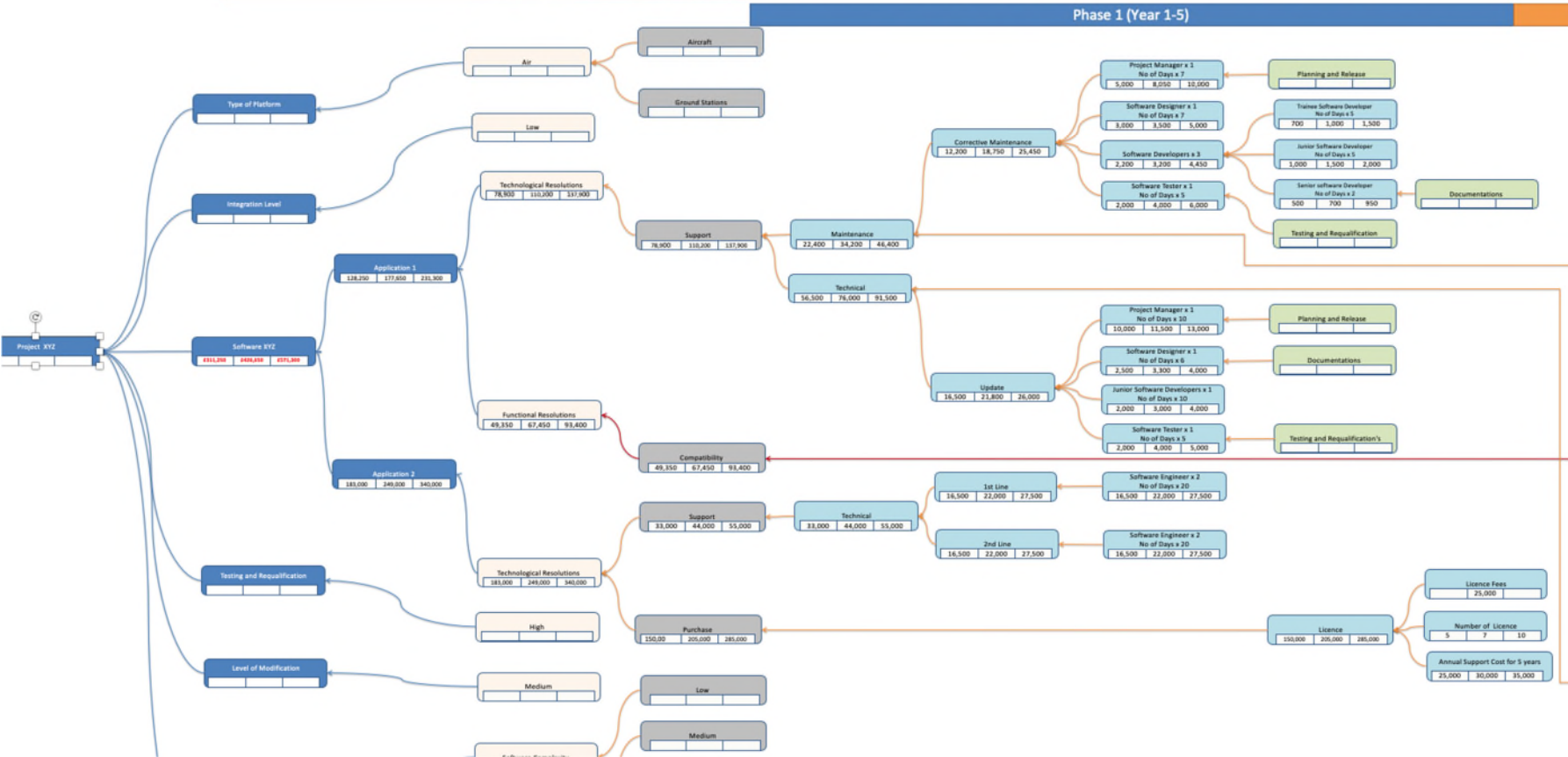


Figure 7-19 Example case for the whole software obsolescence resolution profile

7.5 Software Obsolescence Cost Model

The values from the resolution profiles are then used as inputs into the cost model. This cost model estimates the cost of resource required to resolve the software obsolescence issue. The cost model itself has a global data sheet at the top of the Data Collection Model as described earlier in this chapter. In the global data sheet, the rates of resources are identified, which in turn would depend on the specific company/organisation rates. Workings of the cost model is shown in the Figure 7-20 below.

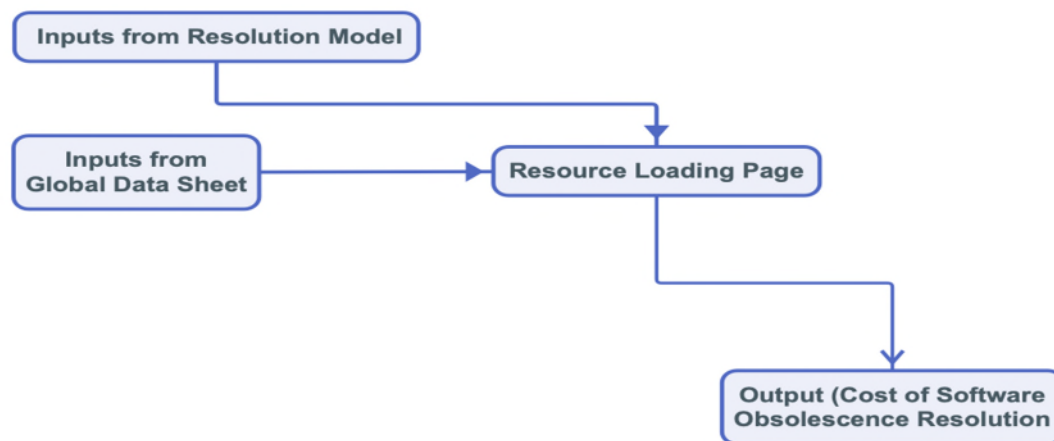


Figure 7-20 Working of the software obsolescence Cost Model

The inputs from the resolution profile are first converted into a “Resource Loading Page” where individual nodes from the Resolution Model is converted into a WBS. level 2 WBS (components) is assigned the relevant resolution approach as described above. And each resolution approach is tasked with the number of resources required to complete the task over a certain period. Therefore, providing a yearly profile. Figure 7-21 shows the Resource loading page of the model.

The output sheet which estimates the cost of the software obsolescence resolution follows the flow of WBS and converts the resource profile into cost. The cost is aggregated at the WBS level one, which also reveals the yearly profile of the cost. Figure 7-22 shows the cost build up and the Figure 7-23 shows the yearly profile.

AutoSave OFF 20190215 Software Obsolescence Cost Analysis Framework UPDATED.xlsm - Read-Only

HomeInsertDrawPage LayoutFormulasDataReviewViewDeveloperTell me

Paste

Cut

Copy

Format

Calibri (Body)11A⁺A⁻

General

Conditional Formatting

Format as Table

Normal 2

Normal

Bad

Good

Neutral

Calculation

Check Cell

Explanatory T...

Insert

Delete

Format

Auto-sum

Fill

Clear

Sort & Filter

Find & Select

Analyse Data

ShareComments

Z13

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Cost Id	WBS Level 1	WBS Level 2	Resolution Level 1	Resolution Level 2	Resolution Level 3	Resolution Level 4	Resolution Level 5	Resolution Level 6	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	
1	Software XYZ	Application 1	Technological Resolutions	Support	Maintenance																			
2																								
3																								
4							Corrective Maintenance																	
5							Project Manager		2	1	1	1	2											
6							Software Designers		2	2	1	1	1											
7							Software Developers																	
8							Trainee Software Developers		3	1	1	0	0											
9							Junior Software Developers		1	1	1	1	1											
10							Senior Software Developers			1	1	1												
11							Software Testers				1	1	3											
12						Perfective Maintenance																		
13							Project Manager								1	1	1	1	1					
14							Software Designers								1	1	1	1	1					
15							Software Developers																	
16							Trainee Software Developers								1	2	1	1						
17							Junior Software Developers								1	2	1	1						
18							Senior Software Developers								1			1						
19							Software Testers											3	2					
20					Technical																			
21					Update																			
22							Project Manager		2	2	2	2	2											
23							Software Designers		2	1	1	1	1											
24							Junior Software Developers		3	3	3	1												
25							Software Testers					3	2											
26					Upgrade																			
27							Software Upgrades																	
28							Software Engineer													1	1	1	1	
29							O/S Upgrade																	
30							Firmware Upgrade																	
31							Project Manager													3	3	3	3	
32							Software Designers													1	1	1	1	
33							Software Developers													2	2	3	3	
34							Software Testers															2	2	
35			Functional Resolutions																					
36				Compatibility																				
37					Systems																			
38						Systems of Sytem Integration																		
39							Systems Engineer								1	2	1	1	1					
40						Hardware to Software Integration																		
41							Systems Engineer								1	2	1	1	1					
42						Software to Software Integrations																		
43							Software Engineer								2	2	2	2	2					
44						Hardware Upgrade																		
45							Server																	
46								Memory																
47								Processors																
48								SSD																
49							Fibre Optics																	
50		Application 2																						
51			Technological Resolutions																					
52				Support																				
53					Technical																			

Software Obs Complexity levelSoftware Obs Prob MatrixSoftware Obs Reslt ApproachUser Guide and DefinitionsGlobal DataInfoResolution InputsSoftware Resolution ProfileSheet1DataInput Resource Loadingoutput+

Figure 7-21 Resource Loading page of the model

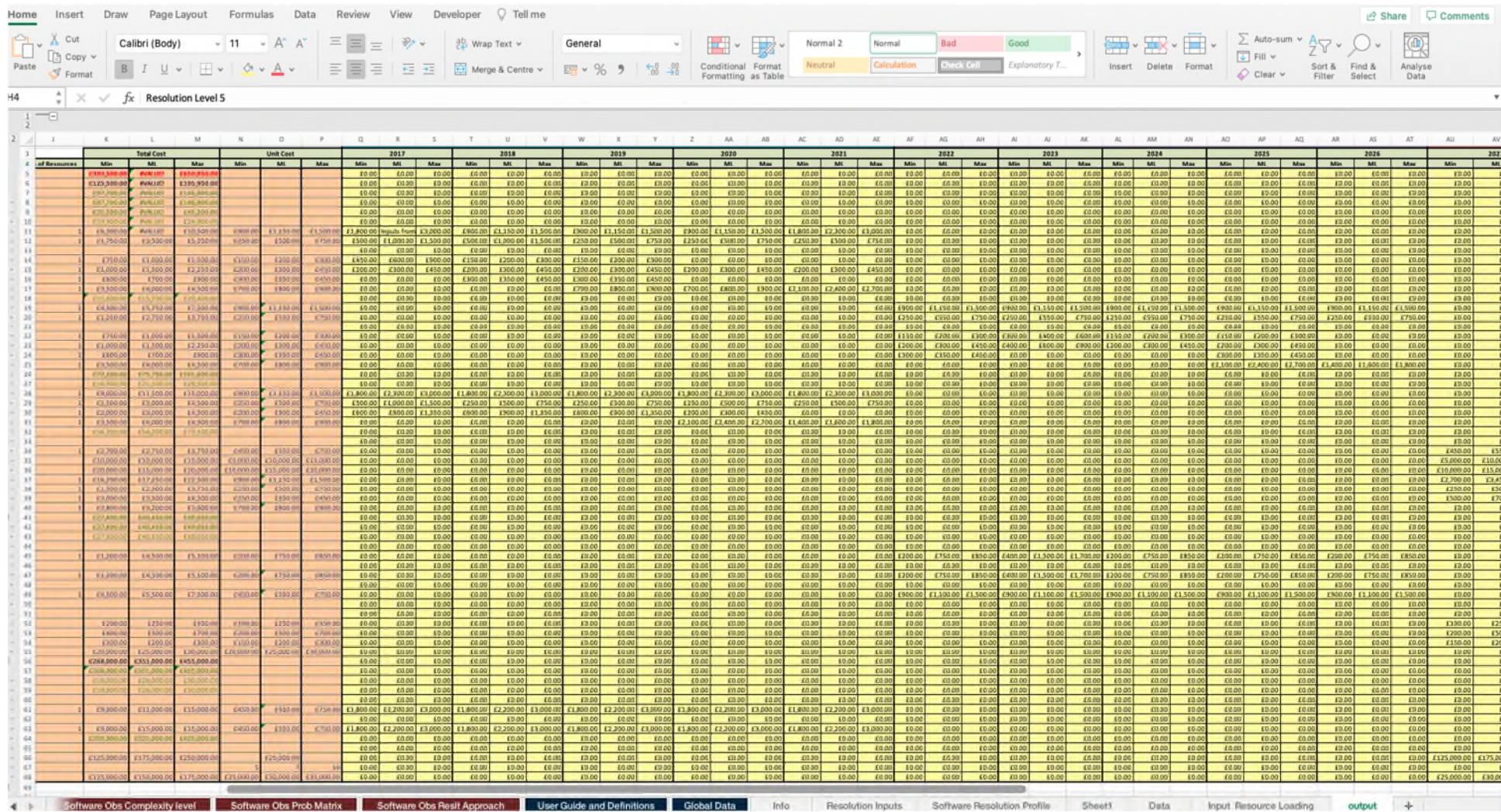


Figure 7-23 Yearly cost profile

7.6 Software Obsolescence Resolution Project Cost

With the above methodology, the researcher was able to estimate the cost of software obsolescence based on the resources. It was now important to know how the other key cost drivers effect the overall project cost.

The cost model discussed in the section five of this chapter relates to just one key cost driver which was the resolution approach, however there are several other key cost drivers that was discussed in Chapter four that drives the whole project cost. To develop a formula that investigates all the key cost drivers, it was necessary to map each cost driver into external variables and constraints. These variables and constraints are identified at the project level rather than component or system level. These variables and constraints are added to the resolution model to show the resolution profile. Figure 7-24 shows how the constraints and variables are added to the resolution profile.

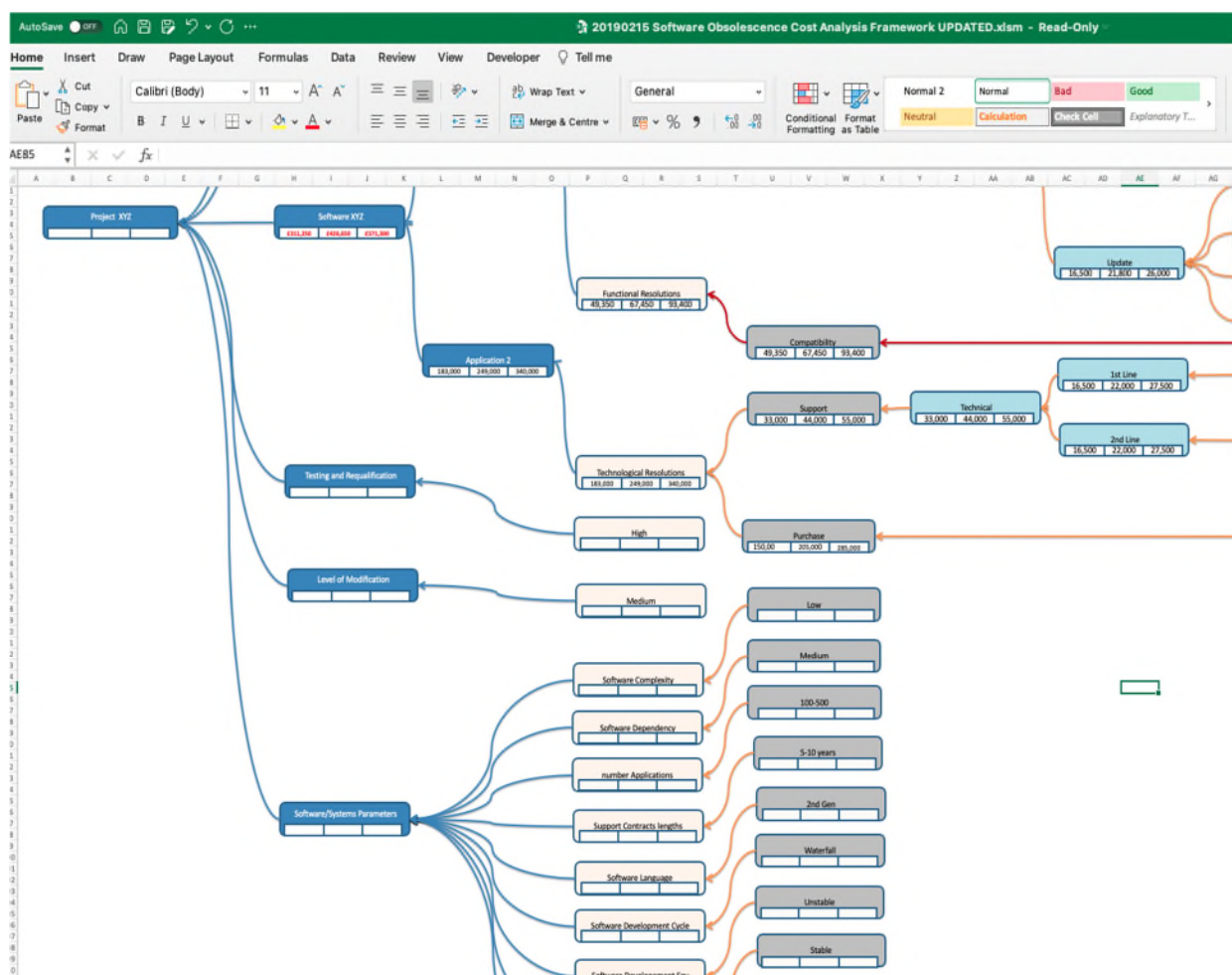


Figure 7-24 Variables and constraints added to the resolution model

Formula for calculating the resolution cost is shown below.

For each resolution identified, the cost of resolution is calculated as

$$RCost = \sum_{i=1}^n Rt_i \times Rr_i \times E_i$$

Where,

$$RCost = Resolution\ Cost(\pounds),$$

$$n = Number\ of\ resources(FTE)$$

$$Rt = Resource\ type$$

$$Rr = Resource\ rate(\pounds)$$

$$E = Effort\ (Months)$$

Equation 7-1 Cost of each resolution

Other key cost drivers and cost elements are added as a constant to the formula to get the total cost of software obsolescence. These constants were postulated as a result of literature reviews, interviews and case studies. These constants need to be matured and calibrated further over time as and when more actual data on software obsolescence is made available.

$$SOBCost = [(m_1Tp \times m_2In) + (m_3Tr \times m_4M)] \times m_5Ap \times RCost$$

Where,

$$SOBCost = Software\ Obsolescence\ Cost\ (\pounds)$$

$$m_n = Constant$$

$$Tp = Type\ of\ platform$$

$$In = Level\ of\ integration$$

$$Tr = Testing\ and\ requalifications$$

$$M = level\ of\ modifications$$

$Ap = \text{Number of applications}$

$R\text{Cost} = \text{Resoultion Cost}$

Equation 7-2 Total cost of Software Obsolescence

Software Systems Parameter covers all the other key cost elements, and which can be calculated as below

The Table 7-2 below gives the mapping of the multipliers and the values.

Table 7-2 Mapping of multipliers and factors

Multiplier Identification number	Descriptions	Values
1	Communication and System	2
	Air	2
	Land	1
	Maritime	1
	Commercial System	1
2,3,4	High	3
	Medium	2
	Low	1
5	10-50 Apps	1
	51-100 Apps	2
	101-500	3
	501 or More	4

An initial validation of the formula, process, multipliers, and variables was undertaken at the ICEAA international workshop in the US in 2018. The formula and the factorial was further

validated and verified as part of the framework validation and the case study validation, results of which are described in Chapter 8. Table 7-3 summaries the key comments from the initial validation exercise.

Table 7-3 summaries the key comments from the initial validation exercise are described

Process	The flow of information needs to be streamlined.
Formula	Asked to revisit the formula as few expressed the grouping to be Inconsistent. E.g., Complexity and Dependencies are grouped together although they said they are independent of each other
Multiplier	Validated the multipliers although advised me to check and rethink the multipliers, as each variable would exhibit different characteristic during Software Development and Maintenance let alone Obsolescence. Advised that the multipliers are an iterative process and should be changed based on the predicted value and the actuals.
Variables	Key feedback was the need to include more variables to undertake a more effective cost optimization.

7.7 Summary of Observation

In this chapter, the development of the Software Obsolescence Cost Analysis Model was presented. The cost model can be used at the bidding stage of the contract in order to estimate the cost of software obsolescence resolution for the contracted period. Whilst the cost model is largely focused and customised for the defence and aerospace industry. It could be modified to be used across other industries.

In section 7.2, the research methodology followed for the development of the software obsolescence resolution cost model was presented. It included a detailed description of the methodology followed to gain a better understanding about the software obsolescence resolution cost.

In section 7.3, the author presented the methodology used to develop Data collection model. This section also covers each aspect of the data collection model. With the questions for each

data collection forms discussed and justified. The section also details the validation activities undertaken with the industry partners to confirm its usage.

In section 7.4, the author presented the methodology used to develop Software Obsolescence Resolution Model and covers every aspect of the resolution model development. This section also covers how the resolution approaches are diagrammatically represented and how the data from the Data collection model is used to develop this model

In the next section, the author describes the development of the Software Obsolescence Cost Model. The author discusses how the data from the Software Obsolescence Resolution Model and inputs from the global data sheet are used to develop a resource loading worksheet which maps into the development of the Software Obsolescence Resolution cost.

In section 7.6, author describes the process involved in the development of the total software obsolescence project cost. Author explains how the key cost drivers that are identified in the chapter four are used to estimate the cost of software obsolescence resolution cost.

In the following chapter, the author describes the implementation of the Software Obsolescence resolution cost analysis framework and the cost model and its subsequent validation and verification by means of six case studies on current projects from Defence industry and qualitative validation with experts from the Defence industry.

8 Demonstration

8.1 Introduction

In chapter 6 and 7, the iterative development of the Software Obsolescence Cost Analysis Framework (SOCAF) and the Software Obsolescence Cost Analysis Model (SOCAM), that can be used at the bidding stages of a contract to estimate the cost of software obsolescence resolution was discussed and presented.

The purpose of this chapter is to describe the implementation of this framework in an excel based tool and its subsequent demonstration by means of six case studies- current projects in Defence industry and the qualitative validation by the experts on the final output.

In section 8.2, the research methodology followed for the demonstration of the SOCAF and SOCAM are presented where the details of the experts who participated in the demonstration are included.

In the next section, the researcher describes the six case studies in detail, along with the study analysis based on the responses that the experts provided to the questionnaire used during the case study sessions.

8.2 Detailed Research Methodology for Software Obsolescence demonstration

The detailed research followed for the demonstration of the Software Obsolescence Cost Analysis Framework and the model are detailed in the Figure 8-1 below. The first phase of the demonstration activity was to undertake a workshop of the framework and the model with ORG B systems. It was agreed with the ORG B Obsolescence lead that they will provide the resources and medium to complete the initial demonstration of the framework and the model. The researcher was given access to their annual obsolescence meeting where the experts from all over the ORG B were present. The response was analysed by identifying similarities, differences, and unique responses across the different participants. The outcome from this analysis is summarised further along in this chapter.

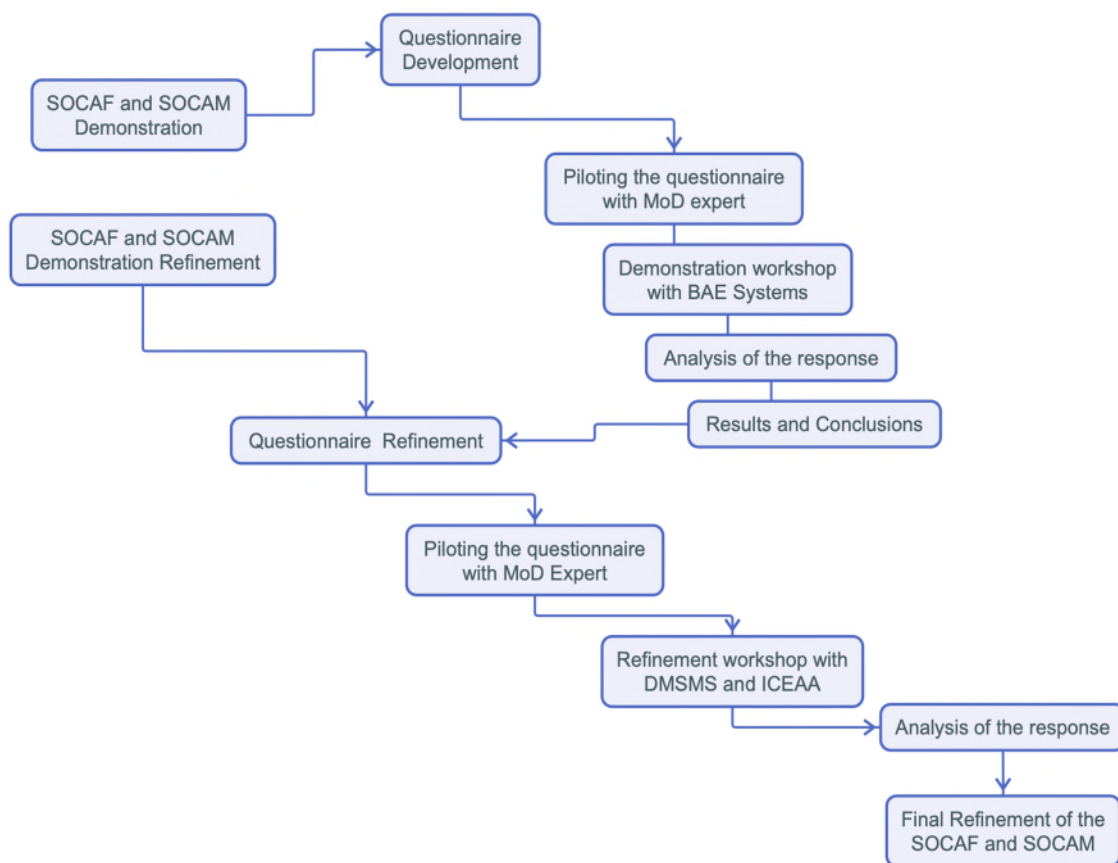


Figure 8-1 Detailed research methodology for demonstration

A total of 15 experts from global ORG B participated in the first phase of the demonstration workshop. Experts attended both physically (experts from UK) and virtually (experts from US and AUS). The detail of the experts is listed in the Table 8-1 below. Whilst the experts attending the workshop had a good understanding and experience of material and component obsolescence, their experience on software obsolescence was limited, although with a good degree of knowledge. The experts concluded that the process of managing software obsolescence is very similar to hardware obsolescence and therefore, will be able to validate the whole framework.

Table 8-1 Detail of experts who participated in the demonstration workshop

Expert No	Organisation	Role	Years of Experience in Obsolescence
1	ORG B	Material Obsolescence Manager	15
2	ORG B	Obsolescence Manager	12
3	ORG B	Assistant Obsolescence Manager	7
4	ORG B (US)	Obsolescence and reliability Manager	13
5	ORG B	Obsolescence Practitioner	5
6	ORG B	Obsolescence Practitioner	5
7	ORG B	Obsolescence Practitioner	4
8	ORG B	Obsolescence Manager	8
9	ORG B	Obsolescence Practitioner	5
10	ORG B	Obsolescence Manager	9
11	ORG B	Obsolescence Manager	8
12	ORG B (AUS)	Obsolescence Manager	11
13	ORG B	Obsolescence Practitioner	3
14	ORG B	Obsolescence Practitioner	2
15	ORG B	Obsolescence Practitioner	3

The approach taken for this demonstration workshop is detailed in the Figure 8-2 below. A phased approach with group discussions, around the different sections of the framework alongside the model was adopted. Feedback questions were provided at the end of each discussion. The copy of the tool and the framework was sent to the participants well in advance for the experts to be familiarised beforehand. Any immediate questions on the framework were clarified using WebEx or skype meeting. On completion of the session feedback was collected and analysed and activities to refine the framework was carried out. Example of the feedback questions are listed below. Please see Annex A for the demonstration question set.

- Do the experts agree with the principle of the framework?
- How confident are they in using the framework?
- Would the experts use the framework to manage their software obsolescence issue?

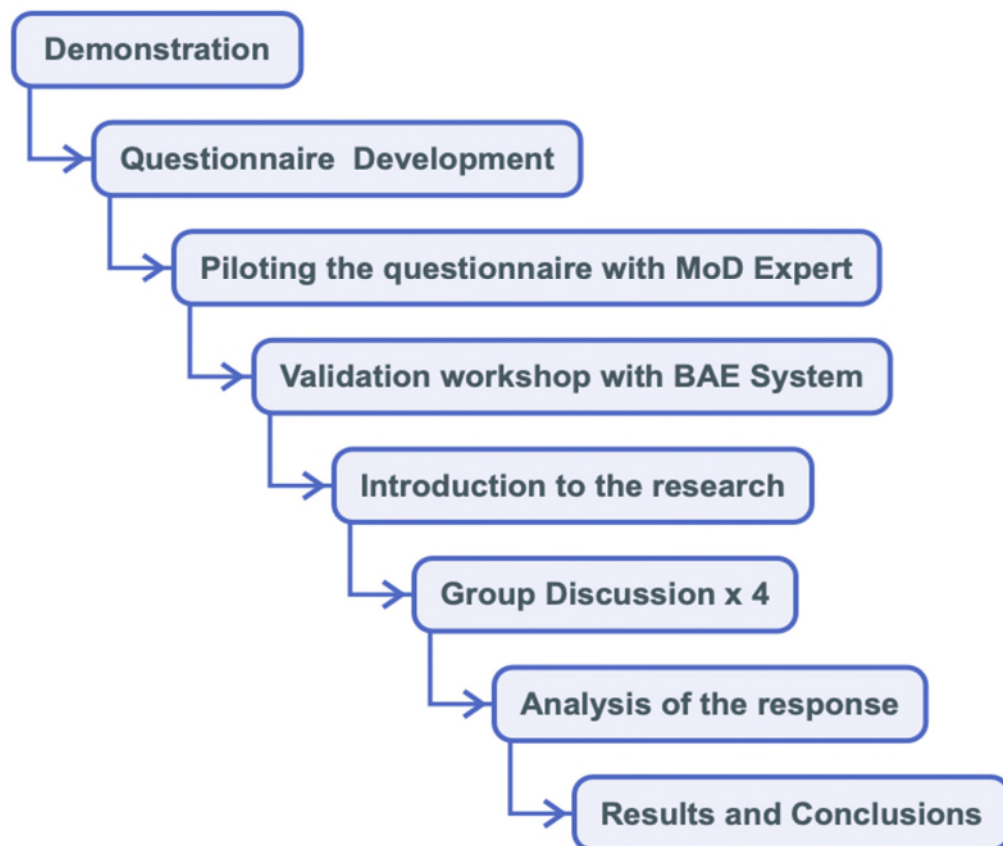


Figure 8-2 Approach taken to complete the demonstration workshop

Demonstration workshop was initiated by the researcher introducing the research and the need for software obsolescence cost analysis framework. This was then followed by introducing the following topics

- Research Methodology
- Research definition
- Research approach
- Gaps in the literature
- Findings from the case studies
- Findings from the expert interviews and online survey
- Findings from the Cognitive case study and finally
- Introduction to the framework

This was followed by the first group discussion where the first iteration of the software obsolescence management levels (Figure 8-4) was introduced. The experts undertook a detailed scrutiny of the management levels. The feedback was collected in the demonstration questionnaire which was analysed and is presented later in this chapter.

This was then followed by the second group discussion where the researcher introduced the Software Obsolescence Resolution Approach. The resolution approach presented at this demonstration exercise was the very first iteration, which was derived from the Bartel et al (2012) of the software supply chain. The group accepted the resolution approach in principle and recommended to have at least three levels of each resolution approach to get to the component level of the software.

In the next step the researcher introduced the Software Obsolescence Key cost drivers. The main aim of this session was to re-enforce the key cost drivers that were identified and concluded through literature review, case studies, expert interviews, and online survey. The researcher introduced the key cost drivers (Figure 8-7) and explained their definitions and their relationship to the resolution approach; in addition to suggesting the experts to contribute to the list of cost drivers which they think the researcher should include based on their experience.

The last group discussion was focused on the software obsolescence complexity matrix (Figure 8-6).

Each element of the software obsolescence complexity matrix was discussed and scrutinised by the experts. In the final part of the demonstration workshop the researcher was able to demonstrate how all the different sections came together to develop the framework (Figure 8-3), and how this framework would enhance the capability of the obsolescence manager in identification and formulation of an appropriate resolution approach.

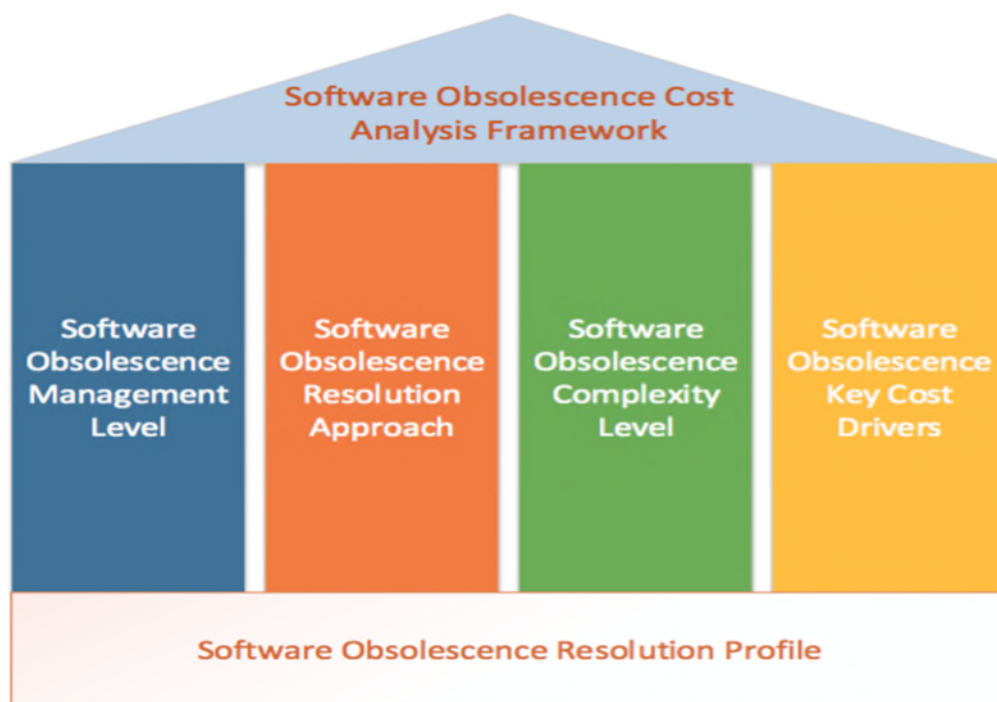


Figure 8-3 First iteration of the software obsolescence cost analysis framework

The workshop concluded with the collection of the feedback questionnaire from the experts. The responses were then collected, analysed by identifying any similarities, differences, and any unique responses across the different participants. The outcomes from this analysis are summarised further in this chapter.

8.2.1 Summary of the feedback from the demonstration workshop

The experts from the ORG B who attended the demonstration workshop agreed that they would be able to validate the process end to end and would provide feedback for the individual sections of the framework. Even though the experts were able to provide constructive feedback against all the sections, experts were more interested with the software obsolescence management levels. ORG B Obsolescence experts were able to relate the management level as a solution to their other obsolescence issue which they found extremely useful and even planned to use the software obsolescence management levels in-house to provide further feedback on the wider usability.

Further feedback was also provided on the software obsolescence complexity level, which was discussed in length. During the session, Participant 2 suggested that the complexity matrix should be in fact a risk impact model, as it gives a clear flexibility in early identification of software obsolescence issue.

The Table 8-2 below summarises the key findings from the first demonstration workshop. Some of the quotes from the workshop are detailed below-

Participant 2- ***“Software Obsolescence Complexity matrix should be used as a Risk Impact Model to get clarity on the level of Obsolescence issue, which is key in identifying these issues very early”***

Participant 4 – ***“There are few other drivers that effects the software obsolescence such as software complexity, dependencies etc., this needs to be part of the key cost drivers”***

Participants 11- ***“Software Obsolescence Management level is key in identifying what is happening now and what we need to do to get to be proactive”***

Table 8-2 Summary of the key findings from the demonstration workshop

	Sections of Framework	Comments/Feedback
1	Software Obsolescence Resolution Approach	1. The workshop delegates agreed that this is a reasonable approach to classify the various resolution approaches. 2. Researcher was asked to investigate the software supply chain in further detail as it was requiring the same level of management and monitoring as the components. 3. Researcher was asked to provide some recommendation on monitoring software supply chain and to include it that as part of resolution approach.
2.	Software Obsolescence Key Cost Drivers	4. Attendees agreed with the approach taken to identify the key cost drivers. 5. Identified other drivers such as skills, process, and planned obsolescence. 6. It was agreed that key cost drivers should be mapped as a horizontal ser rather than the exisiting vertical grouping, given the key cost drivers could potentially also be part of the section of the framework, which was not included originally.
3.	Software Obsolescence Management Level	7. It was recommended to use this as part of Software Obsolescence Standards

		8. Researcher was advised to work with IOM to incorporate this into BS 62402 9. It was advised to include other Software Development standards along with CMMI, as this would provide more visibility to wider software development establishments.
4.	Software Obsolescence Complexity Matrix	10. It was proposed to change Software Obsolescence Complexity matrix to Risk Impact Matrix as this would provide more clarity while estimating cost of software obsolescence resolutions.

Subsequent changes were made to the framework following this feedback. The framework subsequently went through several iteration of the development and the final version was validated using the case studies which is discussed in detail further in this chapter.

The second phase of the demonstration involved the refinement of the initial question set which was then piloted with the MoD obsolescence experts. The refinement workshops were undertaken at the 2018 DMSMS and ICEAA workshops respectively. Five experts from various Defence and aerospace industries attended these workshops. The experiences of experts are detailed in Table 8-3 and, Table 8-4 below.

Table 8-3 Experience of experts who participated in the DMSMS workshop

Expert No	Organisation	Role	Years of Experience in Obsolescence
1	ORG_A	Reliability Expert	5
2	ORG_A	Senior Obsolescence Manager	6

3	ORG_B	Supply Chain Manager	7
4	ORG_C	Obsolescence Manager	10
5	ORG_B	Program Manager	12

Table 8-4 Experience of experts who participated in ICEAA Workshop

Expert No	Organisation	Role	Years of Experience in Obsolescence
1	ORG_D	Risk and Assurance Manager	7
2	ORG_E	Head of Project Control	19
3	ORG_F	Senior Cost Estimator	13
4	ORG_G	Software Cost Estimator	8
5	ORG_F	Senior Risk Manager	9

The same process was undertaken to complete the second phase of the demonstration, with the feedback collected and triangulated. The feedback from the first and the second workshop appeared to be converging closely and therefore it was decided that the framework and the model was now ready to be tested using industry case studies.

Following the feedback, the framework (Figure 6-6), was much more streamlined with the data flow within the framework and the components are highlighted furthermore. Within Software Obsolescence Management Levels, further factors were included, and the type of strategy identified (Figure 8-4). The high-level resolution approach within Software Obsolescence Resolution Approach (Figure 8-5) was further broken down to level 2 and 3 of the resolution approaches. Following the feedback, Software Obsolescence Complexity Matrix (Figure 8-6) was also adjusted to make it look like a risk impact model and finally Software Obsolescence Key Cost Driver (Figure 8-7) was explored further, and other key drivers added to the list.

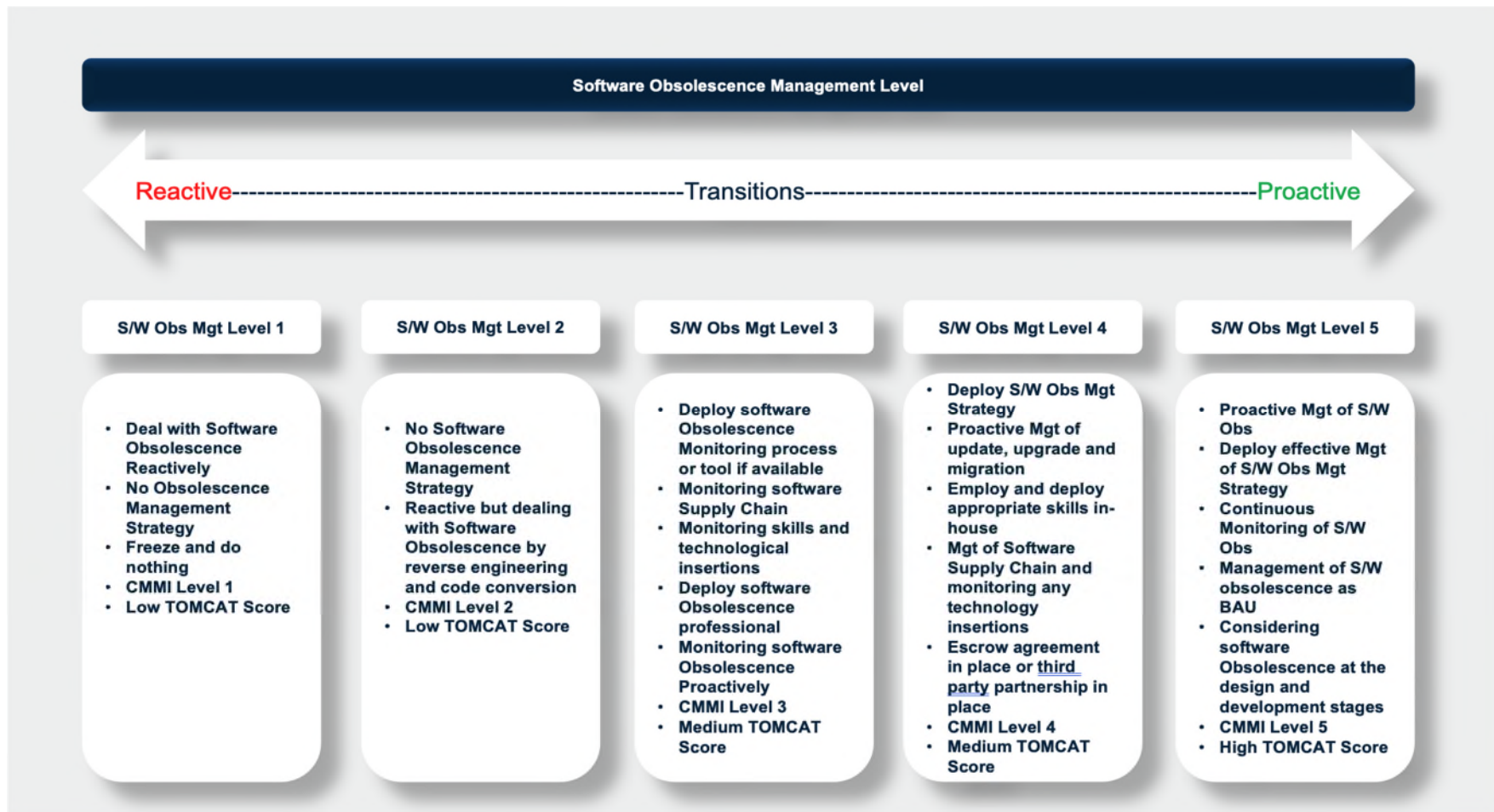
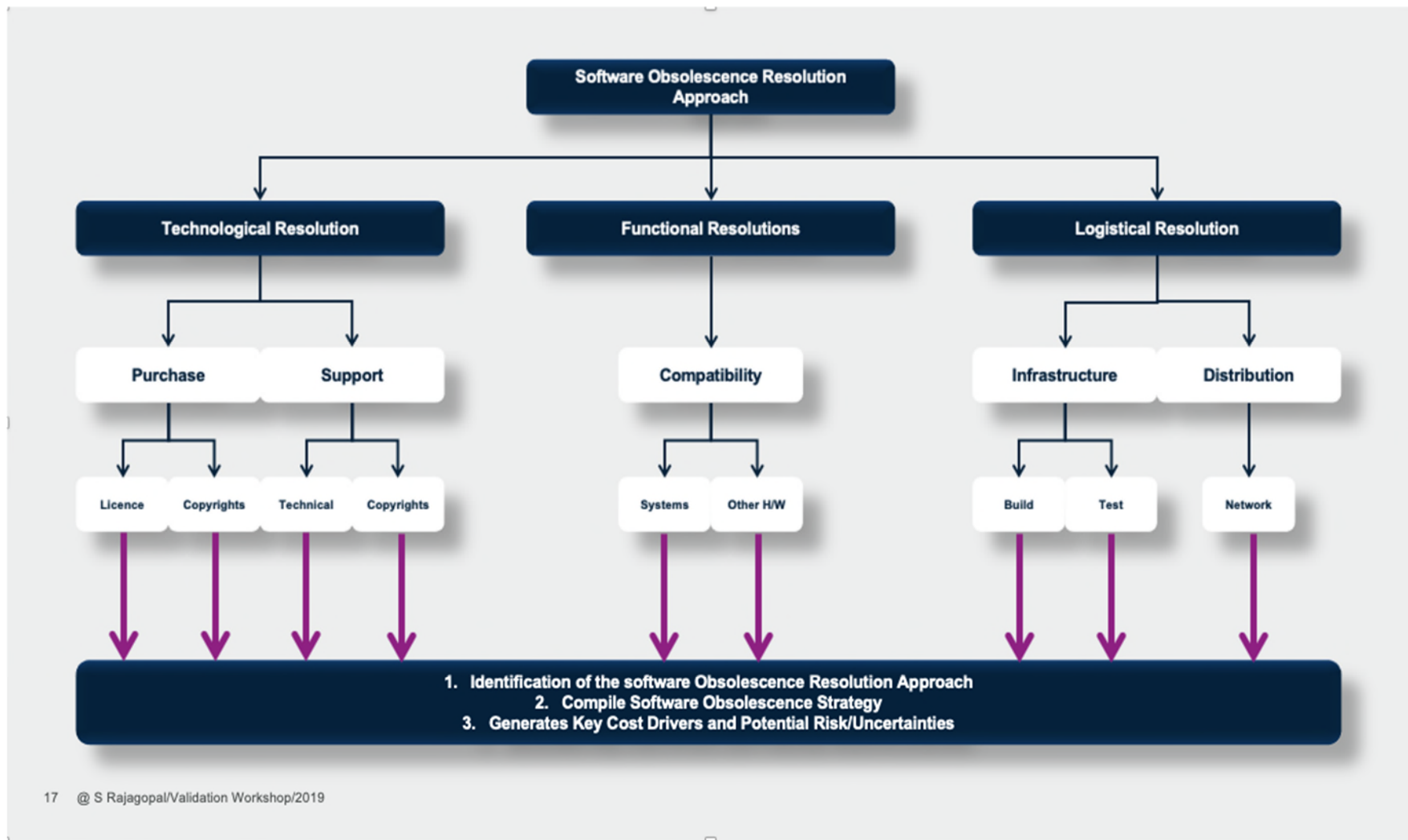


Figure 8-4 Software Obsolence Management Levels



17 @ S Rajagopal/Validation Workshop/2019

Figure 8-5 Software Obsolescence Resolution Approach

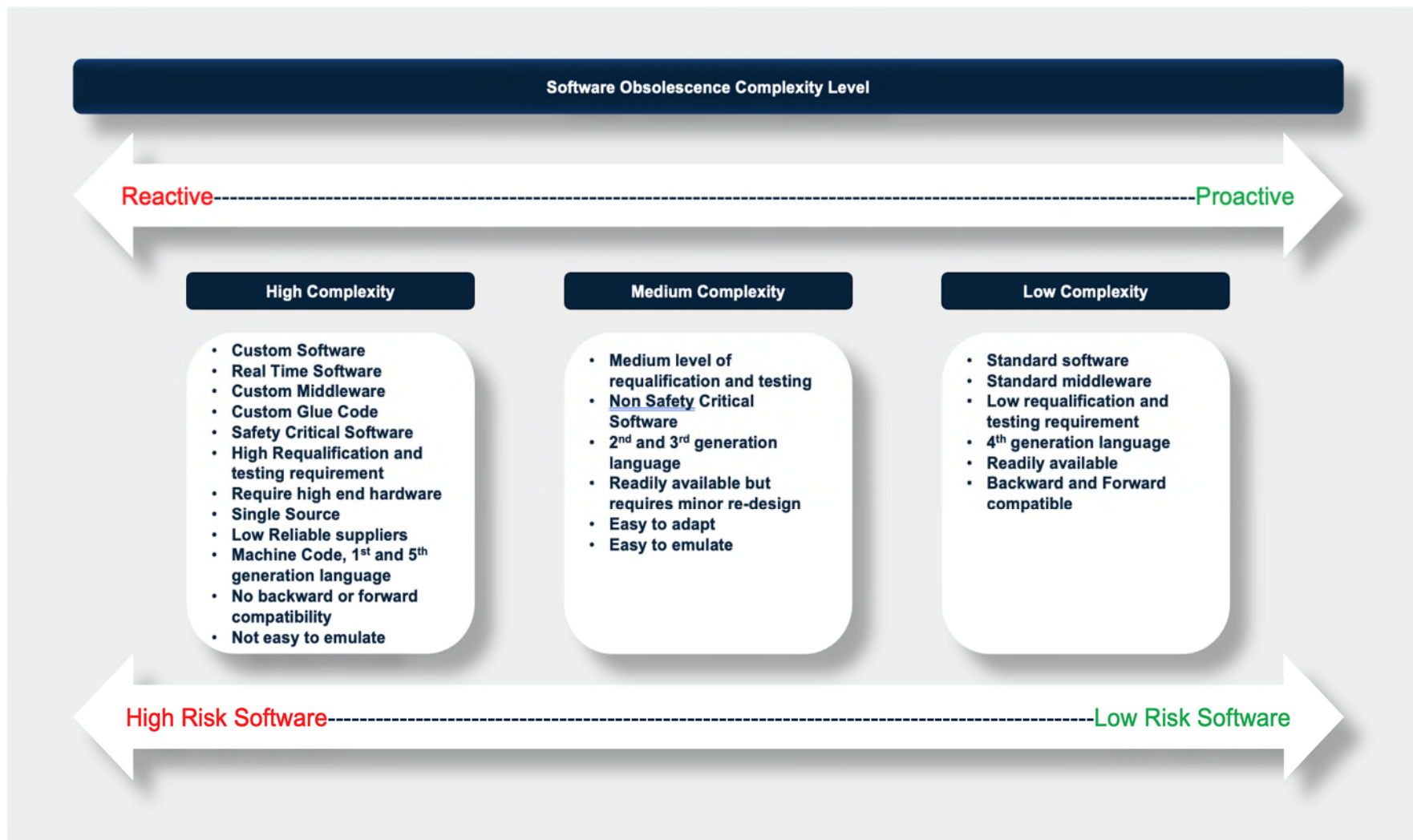


Figure 8-6 Software Obsolescence Complexity Levels

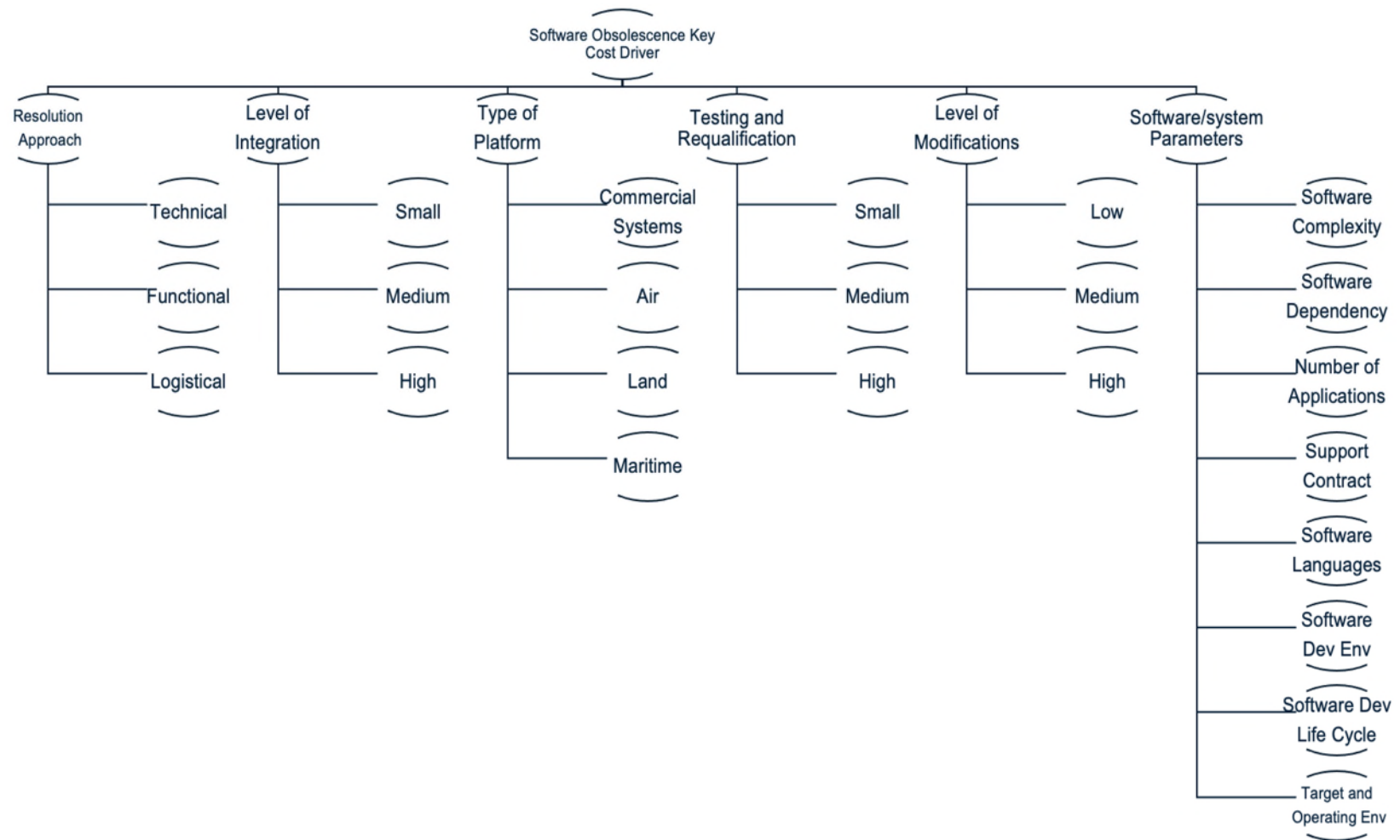


Figure 8-7 Software Obsolescence Key Cost Driver

8.3 Software Obsolescence Cost Analysis Framework Demonstration Case Study.

Software Obsolescence Cost Analysis framework has been validated by applying it to the five case studies across different domains in the Defence industry. Following the initial demonstration workshop and firming up the framework and its sections, the researcher wanted to validate this using a real-time Defence project by means of case studies. The approach undertaken for the case studies demonstration is described in the Figure 8-8 below.

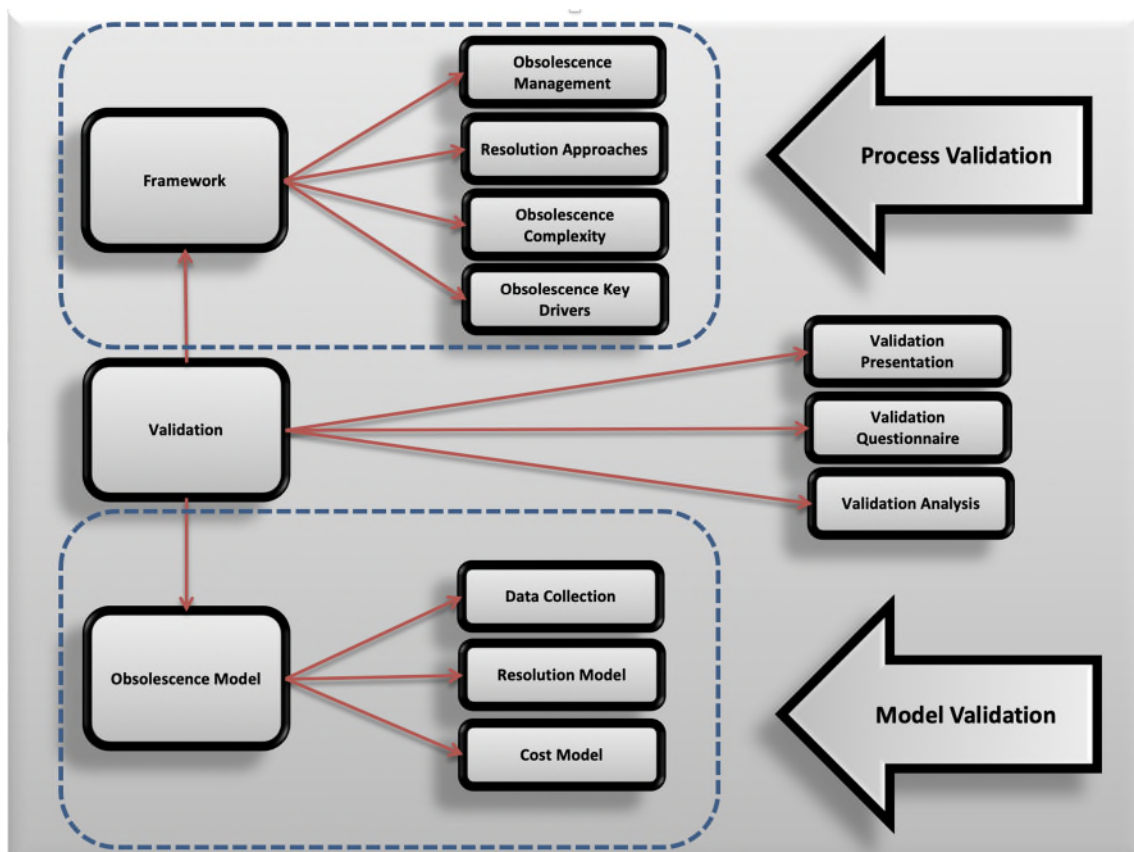


Figure 8-8 Approach undertaken for case study demonstration

The case study demonstration was undertaken in two phases. In the first phase, demonstration of the process was undertaken, whereby each process in the framework was put to test using the case study projects.

In the second phase of the demonstration, the software obsolescence models such as the data collection model, resolution model and the Cost Model was verified and compared against the budgeted cost. It was noted that the budget cost was based on SME judgement due to lack of existence of any data on software obsolescence and was grouped under maintenance cost. The research methodology undertaken to complete the case study is shown in Figure 8-9 below.

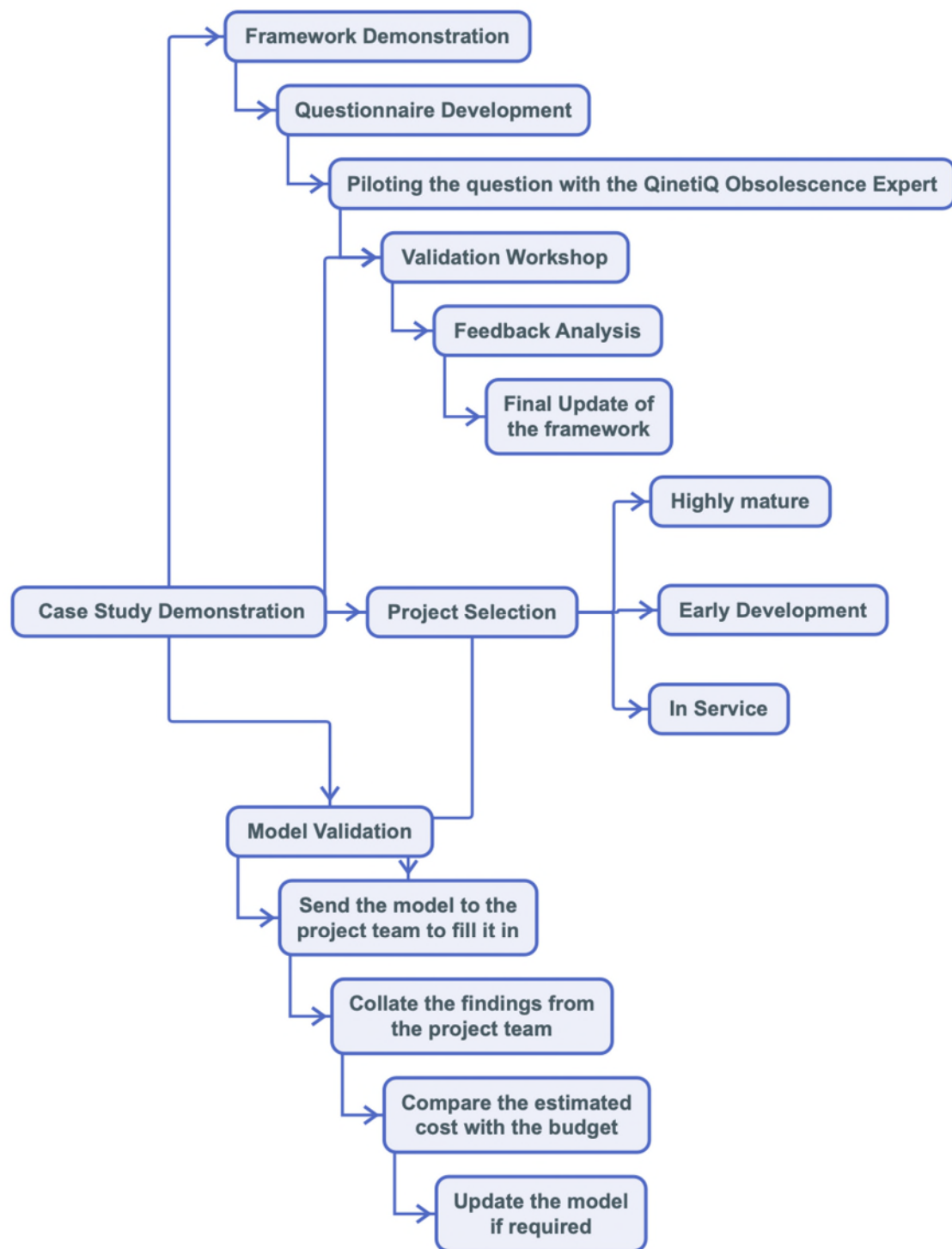


Figure 8-9 Research methodology for Case Study Demonstration

8.3.1 Questionnaire Development

The first phase of the case study demonstration was to validate the framework using actual project in the Defence domain. For this purpose, a questionnaire was

compiled (Annex B), which tested and piloted with QinetiQ (UK) Obsolescence Expert. The question set was divided into

- Logic
- Benefits of using the framework
- Limitation of the framework
- Usability of the framework

Examples of the questions are listed below

- Is the flow of information in the framework logical?
- Is the framework suitable at the bidding stage?
- What are the potential limitation and challenges in using and implementing the framework?
- What are the strongest and weakest feature of the framework?

8.3.2 Case Study –Project selections

Due to the lack of knowledge, data and information about software obsolescence, researcher posted a request for participation for the demonstration activity in the ORG B Portal, which was their internal communication channel.

A total of ten project teams (Table 8-5, shows the projects with its maturity assessment) came forward for the demonstration exercise. Researcher developed a selection criterion for the teams based on the

- Data availability
- Development Maturity

Table 8-5 Software project maturity assessment

<u>Software Projects</u>	<u>Early Development</u>	<u>Highly Mature</u>	<u>In Service</u>
<u>Project A</u>	<u>Y</u>		-
<u>Project B</u>	<u>Y</u>		
<u>Project C</u>	-		<u>Y</u>
<u>Project D</u>		<u>Y</u>	-
<u>Project E</u>			<u>Y</u>
<u>Project F</u>	-	<u>Y</u>	
<u>Project G</u>	-		<u>Y</u>
<u>Project H</u>	-	<u>Y</u>	-
<u>Project I</u>	<u>Y</u>		-
<u>Project J</u>	<u>Y</u>		

Three projects out of the ten were discarded due to lack of data and information required to complete the demonstration. Two projects were kept on the reserve list in case any of the selected projects were unable to complete the demonstration for both the framework and model on time.

The projects that were selected had various levels of data availability with none of the projects having all the required information needed. This will, therefore provide the researcher with the validation of the framework to assess its flexibility on the data availability issue.

8.3.3 Software Obsolescence Cost Analysis Framework Demonstration Case Study

The SOCAF has been validated by applying it to the five case studies across different platforms in the Defence sector. The above analysis was a qualitative demonstration where the flow of information in the framework was tested and validated.

In this section, the researcher applied the framework to a actual project to validate the usability of the framework and the output provided by this framework was compared against the estimated cost by the project team during the bidding phase.

Case Study 1 – Air platform (Project A)

This case study covers the software obsolescence management for the support of the data analysis software, for a radar system in an Air platform for Defence. This software project is at a very early stage of the development and at the bidding stage of the project life cycle. The project will be contracted for an initial 6-year period and then re-contracted for another 4 years. There is no requirement at present for the project team or from the customer to investigate any software obsolescence issue however, the project manager has provided an argument that since this is a new piece of technology and since this is planned to be contracted for a total of 10 years, there are chances that the technology will get obsolete during this contracted period. Due to this, the management has agreed to plan for any future obsolescence eventuality, so that there is an increased level of confidence from MoD customer.

This case study was chosen due to its requirement to understand any potential obsolescence issue. In this case study, the researcher focused solely on the software element. This case study was carried out with the support from the Principal Radar Scientist who had more than 20 years of experience. The summary of the key inputs is detailed in the Table 8-6 below.

Table 8-6 Summary of Case Study A

Descriptions	Inputs
Type of platform	Air
Level of integration	Medium
Testing and requalification	High
Level of modification	Medium
Number of applications	2

The software obsolescence management level part of the framework was not officially completed, given the project team was very new to the idea of software obsolescence and therefore had agreed unanimously that they are at the Software Obsolescence Management Level 1. A recommendation was provided to the project team to transition from Software Obsolescence Management Level 1 to Level 2.

Software Obsolescence Complexity of this software was determined at each individual component level. This software consisted of two applications- the analysis of software obsolescence complexity matrix found that application 1 was highly complex and therefore carried a greater risk of the software being obsolete, whereas application 2 had been analysed to be of medium complexity. Figure 8-10 shows the complexity matrix.

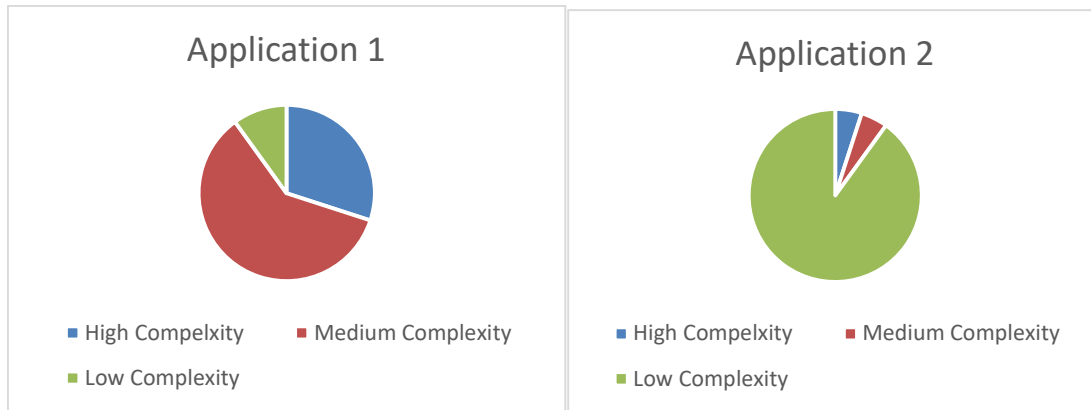


Figure 8-10 Software Obsolescence Complexity Matrix Case Study A

As one of the applications within the software was rated to be of high complexity for software obsolescence, the whole software could be at a high risk of going obsolete. This was the first indication that there was a requirement to develop a software obsolescence resolution approach to these applications individually.

As a result of the above finding, the framework predicted that this software would go obsolete by year two of the development due to a lack of third-party support to one of the applications and in addition predicted eight obsolescence issues throughout its 10-year project life cycle. This framework also helped to identify potential obsolescence issues if the contract were to extend for another five years. Each application would go through several obsolescence issue/s at different stages of its life cycle. Resolution approaches were identified for each potential obsolescence issue as detailed in the Table 8-7 below.

Table 8-7 potential obsolescence issue identified within case study A.

Years	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8	Year 9	Year 10
Resolution Approach										
Functional Resolution										
Technological Resolution										
Logistical Resolution										

To estimate the cost, following (Table 8-8) factorial was applied to the estimating formula.

Table 8-8 Matrix for Case Study A

Descriptions	Inputs	Values
Type of platform	Air	2
Level of integration	Medium	2
Testing and requalification	High	3
Level of modification	Medium	2
Number of applications	2	1

Calculations

This software has got two applications as mentioned above. Both of these applications would undergo different resolution approach throughout the life of the software, as identified in the software obsolescence resolution approach analysis. Resolution approach for the first application is shown below in Figure 8-11

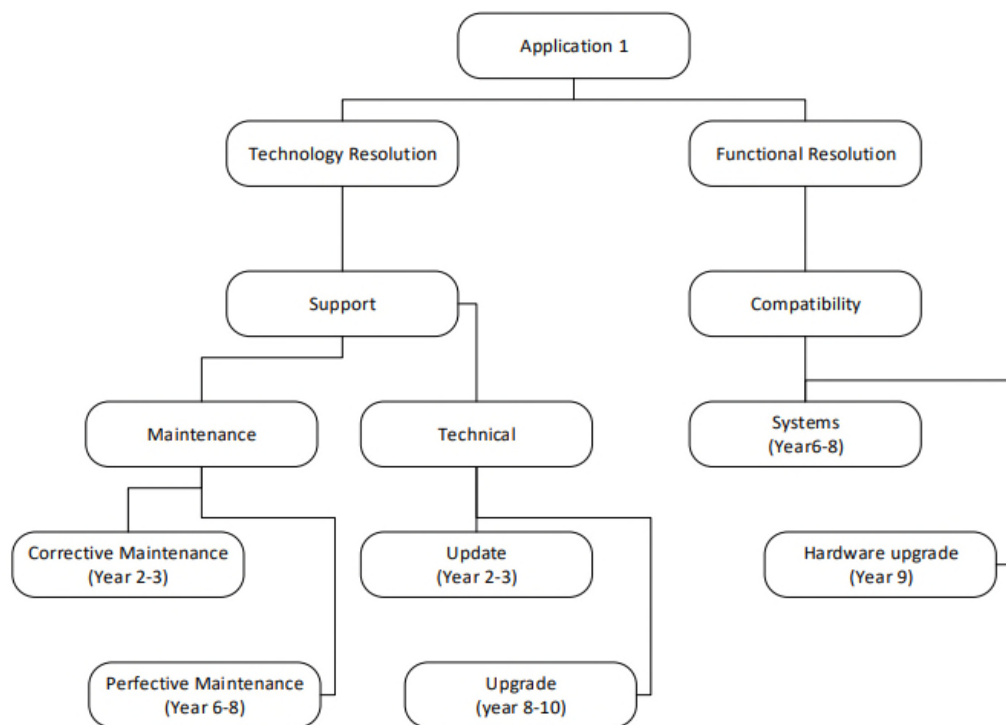


Figure 8-11 Resolution Approach for Application 1

According to the project team the following resources (Monthly rates are based on the organisational agreed rate card for each type of resources. The rates based on 2020 economic conditions and have been escalated @2.5%. (Appendix G The day rate escalation table).

Table 8-9) are required for each resolution approach taken for application 1

Monthly rates are based on the organisational agreed rate card for each type of resources. The rates based on 2020 economic conditions and have been escalated @2.5%. (Appendix G The day rate escalation table).

Table 8-9 Resource type, Rate, Duration and number of resources required for resolution of Application 1

Resolutions approach	Type of Resources	Monthly Rate	Duration Month	Number of resources
Corrective	Project Manager	£12,915	3	1
	Software designer/Engineer	£11,131	2	1
	Senior software Developer	£11,131	3	1
	Junior Software Developer	£10,225	3	1
	Trainee Software Developer	£7,801	3	1
	Software Tester	£11,131	2	1
Perfective	Project Manager	£14,256	3	1
	Software designer/Engineer	£12,287	2	1
	Senior software Developer	£12,287	3	1
	Junior Software Developer	£11,286	2	1
	Trainee Software Developer	£8,611	3	1
	Software Tester	£12,287	2	1
Update	Project Manager	£12,915	3	1
	Software designer/Engineer	£11,131	2	1
	Junior Software Developer	£10,225	1	1
	Software Tester	£11,131	2	1
Upgrade	Project Manager	£15,352	2	1
	Software designer/Developer	£13,231	2	1
	Junior Software Developer	£12,154	1	1

	Software Tester	£13,231	2	1
System	Project Manager	£14,256	2	1
	Systems Engineers	£13,994	1	1
	Software Engineers	£12,857	2	1
	Technicians	£12,857	2	1
Hardware Upgrade	Project Manager	£15,736	1	1
	Systems Engineers	£15,447	1	1
	Software Engineers	£14,192	1	1
	Technicians	£14,192	1	1

The total cost (Constant Cost) of software obsolescence resolution for Application 1 is £690K. The calculation is detailed below in .

Table 8-10 and is based on the Equation 7-1 Cost of each resolution.

Table 8-10 Total Resolution Cost for Application 1

Resolutions approach	Type of Resources	Monthly Rate	Duration Month	Number of resources	Total cost	Total Resolution cost
Corrective	Project Manager	£12,915	3	1	£38,745	£170,740
	Software designer/Engineer	£11,131	2	1	£22,262	
	Senior software Developer	£11,131	3	1	£33,393	
	Junior Software Developer	£10,225	3	1	£30,675	
	Trainee Software Developer	£7,801	3	1	£23,403	
	Software Tester	£11,131	2	1	£22,262	
Perfective	Project Manager	£14,256	3	1	£42,768	£177,182
	Software designer/Engineer	£12,287	2	1	£24,574	
	Senior software Developer	£12,287	3	1	£36,861	
	Junior Software Developer	£11,286	2	1	£22,572	
	Trainee Software Developer	£8,611	3	1	£25,833	
	Software Tester	£12,287	2	1	£24,574	
Update	Project Manager	£12,915	3	1	£38,745	£93,494
	Software designer/Engineer	£11,131	2	1	£22,262	
	Junior Software Developer	£10,225	1	1	£10,225	
	Software Tester	£11,131	2	1	£22,262	
Upgrade	Project Manager	£15,352	2	1	£30,704	£95,782
	Software designer/Developer	£13,231	2	1	£26,462	
	Junior Software Developer	£12,154	1	1	£12,154	
	Software Tester	£13,231	2	1	£26,462	
System	Project Manager	£14,256	2	1	£28,512	£93,934
	Systems Engineers	£13,994	1	1	£13,994	
	Software Engineers	£12,857	2	1	£25,714	
	Technicians	£12,857	2	1	£25,714	
Hardware Upgrade	Project Manager	£15,736	1	1	£15,736	£59,567
	Systems Engineers	£15,447	1	1	£15,447	
	Software Engineers	£14,192	1	1	£14,192	
	Technicians	£14,192	1	1	£14,192	
Total Resolution Cost for application 1						£690,699

The resolution approach for application 2 is shown in Figure 8-12

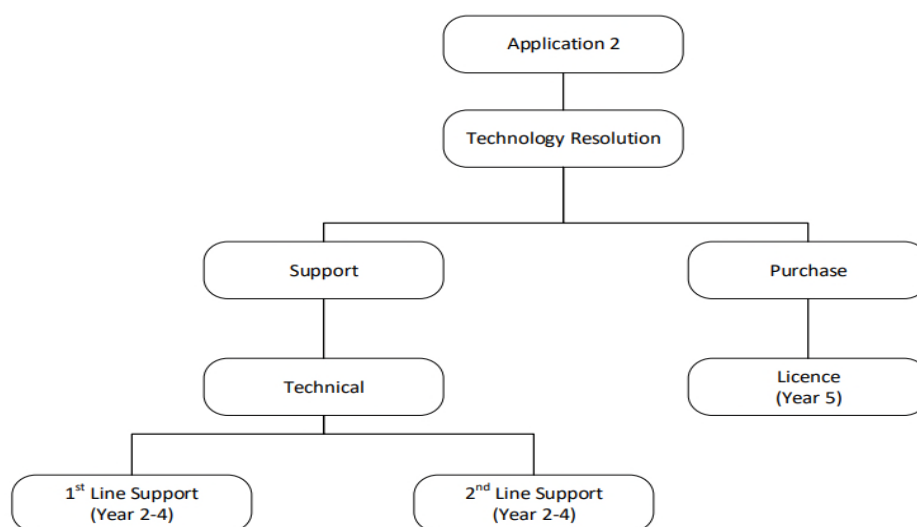


Figure 8-12 Resolution Approach for Application 2

According to the project team the following resources (Table 8-11) are required for each resolution approach for application 2

Monthly rates are based on the organisational agreed rate card for each type of resources. The rates are based on 2020 economic conditions and have been escalated @2.5%. (Appendix G The day rate escalation table).

Table 8-11 Resource type, Rate, Duration, and number of resources required for resolution of Application 2

Resolution Approach	Resources	Duration(Month)	Number of Resources	Monthly Rate
First Line Support	Software Engineer	1.5	2	£11,648
Second Line Support	Software Engineer	1	2	£11,648
Licence			2	£5,000

The total cost (Constant Cost) of software obsolescence resolution for Application 2 is £63K. The calculation is listed below in

Table 8-12 and is based on the Equation 7-1 Cost of each resolution.

Table 8-12 Total Resolution Cost for Application 2

Resolution Approach	Resources	Duration(Month)	Number of Resources	Monthly Rate	Total Cost	Total Cost of Resolution
First Line Support	Software Engineer	1.5	2	£11,648	£34,944	
Second Line Support	Software Engineer	1	2	£11,648	£23,296	£58,241
Licence			2	£5,000	£10,000	£5,000
Total Resolution Cost for application 2						£63,241

The total cost (constant cost) for the software obsolescence resolution of application 1 and 2 is £753K.

Based on the Equation 7-1, the total cost of software obsolescence on this project is calculated as below. The value for the constants is obtained from Table 8-8.

$$\text{Total software obsolescence cost} = [(2 \times 2) + (3 \times 2)] \times 1 \times 753,940$$

$$= \text{£7,539,396}$$

As a result, the Software Obsolescence Cost Analysis Framework predicted the Software Obsolescence resolution cost at £7.5M. This result was compared to the project teams original technical estimate of £6.2M for software obsolescence work packages.

An independent workshop was undertaken to clearly define software and hardware obsolescence cost. Due to lack of historical information, and the rigour with which the framework worked to estimate the cost of resolution, the project team collectively agreed to bid the resolution cost for software obsolescence at £7.5M.

As a learning from this case study, the project team started an internal data collection activity to collect and monitor software obsolescence for this project.

A Financial analysis was undertaken to understand the Net Present Value (NPV). The discount factor used for this calculation was 3% resulting in an NPV of £7.3M for this project.

Case Study 2 -Land Platform (Project B)

This case study covers the software obsolescence management for the support of the GUI for a communication system, in a Land based platform for Defence. This software project is at a very early stage of the design and development, and at an early bidding stage of the project life cycle. The project will be contracted for an initial 5-year period and then re-contracted for further 5 year. This project came into existence due to an earlier obsolescence issue.

This case study was chosen due to its requirement to understand any potential future obsolescence issue, with a key focus on the software element. This case study was carried out with the support from the Technical Leader- Software Development who had 38 years of experience.

The summary of the key inputs is detailed in the Table 8-13 below.

Table 8-13 Summary of the Case Study B

Descriptions	Inputs
Type of platform	Land
Level of integration	Low
Testing and requalification	Low
Level of modification	Medium
Number of applications	6

The software obsolescence management level part of the framework was not officially completed, given the project team was very new to the idea of software obsolescence and therefore had agreed unanimously that they are at the Software Obsolescence Management Level 1. A recommendation was provided to the project team to transition from Software Obsolescence Management Level 1 to Level 2.

Complexity of this software was determined at each individual component level. This software consisted of six applications. The analysis of software obsolescence complexity matrix found that applications no 2 and 6 were highly

complex and therefore carried a greater risk of the software being obsolete. Findings from the Complexity matrix can be found in Figure 8-13 below.



Figure 8-13 Software Obsolescence Complexity Matrix Case Study B

Given two of the six applications were rated to be of high complexity for software obsolescence, the whole software could be viewed as is at a high risk of going obsolete.

As a result of the above finding, the framework predicted that this software would go obsolete in the year five due to system upgrade; and predicted three obsolescence issues throughout its 10-year project life cycle. Each application would go through several obsolescence issue/s at different stages of its life cycle. Resolution approaches were identified for each potential obsolescence issue as detailed in the Table 8-14 below.

Table 8-14 potential obsolescence issue identified within case study B

Years	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8	Year 9	Year 10
Resolution Approach										
Functional Resolution										
Technological Resolution										
Logistical Resolution										

To estimate the cost, the following factorial (Table 8-15) was applied to the estimating formula.

Table 8-15 Matrix for Case Study B

Descriptions	Inputs	Values
Type of platform	Land	1
Level of integration	Low	1
Testing and requalification	Low	3
Level of modification	Medium	2
Number of applications	6	1

As a result, the Software Obsolescence Cost Analysis Framework predicted the Software Obsolescence resolution cost at 5.8M. This result was compared to the project teams technical estimate which was at £5.1M. Project team's estimate was developed by the technical team which was largely based on resource requirement, sub-contracting activity, and any third-party licencing requirement.

Case Study 3 – Navigation System (Project C)

This case study covers the software obsolescence management for the support of the Rx and Tx Software of a satellite communication chip set. This software project is in the in-service stage of the project life cycle, and has been contracted for 7 years of software support. Due to the nature of the contract, it has been mandated by the authority to have a robust plan for obsolescence and a good mitigation strategy in place for the chipset. However, it has not been mandated to have separate software obsolescence management and costed independently.

This case study was chosen due to its requirement to understand any potential future obsolescence issue, with a key focus was on the software element. This case study was carried out with the support from the senior Software Engineer who had more than 28 years of experience.

The summary of the key inputs has been detailed within the Table 8-16 below.

Table 8-16 Summary of Case Study C

Descriptions	Inputs
Type of platform	Air
Level of integration	High
Testing and requalification	High
Level of modification	High
Number of applications	12

The software obsolescence management level part of the framework was not officially completed, given the project team was very new to the idea of software obsolescence and therefore had agreed unanimously that they are at the Software Obsolescence Management Level 1. A recommendation was provided to the project team to transition from Software Obsolescence Management Level 1 to Level 2.

Complexity of this software was determined at the system level. Due to the nature of the project the maturity assessment was made at “Software” and “Firmware” Level. Findings from the Complexity matrix can be found in Figure 8-14 below.

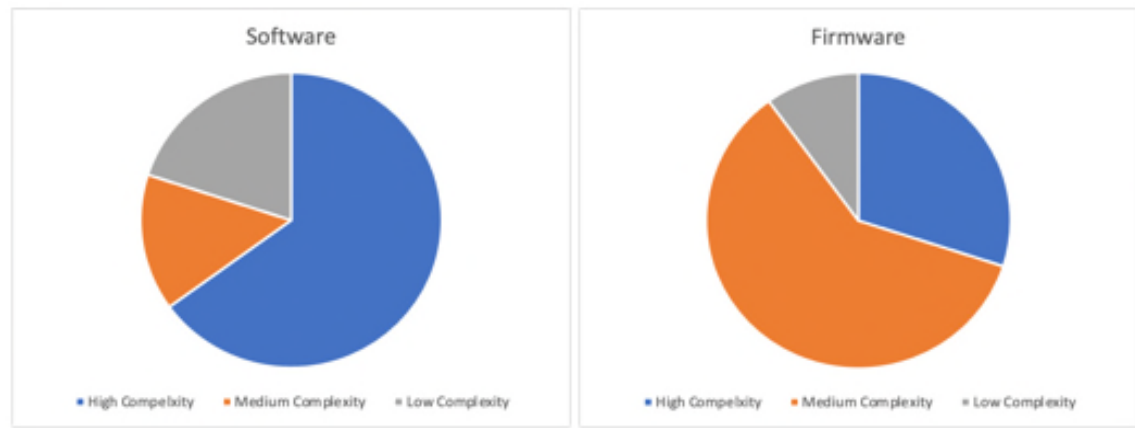


Figure 8-14 Software Obsolescence Complexity Matrix – Case Study C

Several applications within the software were classed to be at a high complexity for software obsolescence. However, on the firmware, there were no major issues identified with respect to software obsolescence.

The framework predicted a total of 24 software obsolescence issues, with 60% of the issues due to FPGA upgrade/update as described in the Component Obsolescence plan. The framework predicted the earliest the software will go obsolete is in year two. Each application would go through several obsolescence issue/s at different stages of its life cycle. Resolution approaches were identified for each potential obsolescence issue as detailed in Table 8-17 below.

Table 8-17 potential obsolescence issues identified within case study C

Years	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8	Year 9	Year 10
Resolution Approach										
Functional Resolution										
Technological Resolution										
Logistical Resolution										

To estimate the cost, the following factorial (

Table 8-18) was applied to the estimating formula.

Table 8-18 Matrix for case study C

Descriptions	Inputs	Values
Type of platform	Air	2
Level of integration	High	3
Testing and requalification	High	3
Level of modification	High	3
Number of applications	12	1

As a result, the Software Obsolescence Cost Analysis Framework predicted the Software Obsolescence resolution cost at £18.5M. This result was compared to the project teams technical estimate which was at £20.1M. Project team's estimate was developed by the technical team which was largely based on resource requirement, sub-contracting activity, and any third-party licencing requirement. The project team confirmed that they were optimistic on the resource estimate due to lack of data and the process by which they could estimate the cost of the software obsolescence.

Case Study 4 – Radar System (Project D)

This case study covers the software obsolescence management for the support of the firmware and software image synthesis for a radar system on an Air platform. This software project is in a mature state. Which has been bid for 7 years of software support contract. It has not been mandated to have separate software obsolescence management and costed independently, due to limited data or process.

This case study was chosen due to its requirement to understand any potential future obsolescence issue, with a key focus was on the software element. This case study was carried out with the support from the senior Principal Software Engineer with more than 20 years of experience.

The summary of the key inputs is detailed in the Table 8-19 below.

Table 8-19 Summary of Case Study D

Descriptions	Inputs
Type of platform	Air
Level of integration	High
Testing and requalification	High
Level of modification	Low
Number of applications	2

The software obsolescence management level part of the framework was not officially completed given the project team was very new to the idea of software obsolescence and therefore had agreed unanimously that they are at the Software Obsolescence Management Level 1. A recommendation was provided to the project team to transition from Software Obsolescence Management Level 1 to Level 2.

Complexity of this software was determined at system level. Due to the nature of the project the maturity assessment was made at “Software” and “Firmware” Level. Findings from the Complexity matrix can be found in Figure 8-15 below.

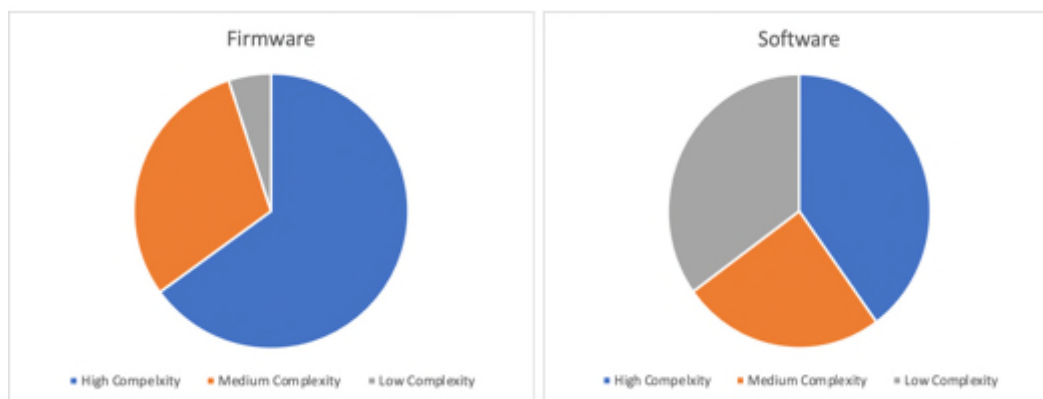


Figure 8-15 Software Obsolescence Complexity Matrix -Case Study D

Several applications within the software were classed to be at a high complexity for software obsolescence. However, on the firmware, only one major software obsolescence issue was identified.

The framework predicted a total of 7 software obsolescence issues, of which majority of them were due to update on the firmware. The framework predicted the earliest the software will go obsolete is in year 6. Each application would go through several obsolescence issue/s at different stages of its life cycle. Resolution approaches were identified for each potential obsolescence issue as detailed in the Table 8-20 below.

Table 8-20 potential obsolescence issue identified within case study D

Years	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8	Year 9	Year 10
Resolution Approach										
Functional Resolution										
Technological Resolution										
Logistical Resolution										

To estimate the cost, following factorial (Table 8-21) was applied to the estimating formula.

Table 8-21 Matrix for Case Study D

Descriptions	Inputs	Values
Type of platform	Air	2
Level of integration	High	3
Testing and requalification	High	3
Level of modification	Low	1
Number of applications	2	1

As a result, the Software Obsolescence Cost Analysis Framework predicted the Software Obsolescence resolution cost at £4.8M. This result was compared to the project teams technical estimate which was at £5.2M. Project team's estimate was developed by the technical team which was largely based on the resource

requirement, sub-contracting activity, and any third-party licencing requirement. The project team confirmed that they were optimistic on the resource estimate due to lack of data and the process by which they could estimate the cost of the software obsolescence.

Case Study 5 – Database Systems (Project E)

This case study covers the software obsolescence management for the support of the data management system of a training module for a land-based platform. This software project is in the in-service phase of the project life cycle, with a contract to support software for 10 years consisting major mid-year updates to the system. It has not been mandated to have software obsolescence management and costed independently. This is due to available data or process to understand software obsolescence is very limited.

This case study was chosen due to its requirement to understand any potential future obsolescence issue, with a key focus was on the software obsolescence. This case study was carried out with the support from the Group Leader QAT who had more than 32 years of experience.

The summary of the key inputs is detailed in Table 8-22 below.

Table 8-22 Summary of Case Study E

Descriptions	Inputs
Type of platform	Land
Level of integration	High
Testing and requalification	Low
Level of modification	Low
Number of applications	35

The software obsolescence management level part of the framework was not officially completed given the project team was very new to the idea of software obsolescence and therefore had agreed unanimously that they are at the Software Obsolescence Management Level 1. A recommendation was provided to the project team to transition from Software Obsolescence Management Level 1 to Level 2.

Complexity of this software was determined at the system level. Due to the nature of the project the maturity assessment was made at “Training Software”, “Application Software” and “Firmware” Level. Findings from the Complexity matrix can be found in Figure 8-16 below.

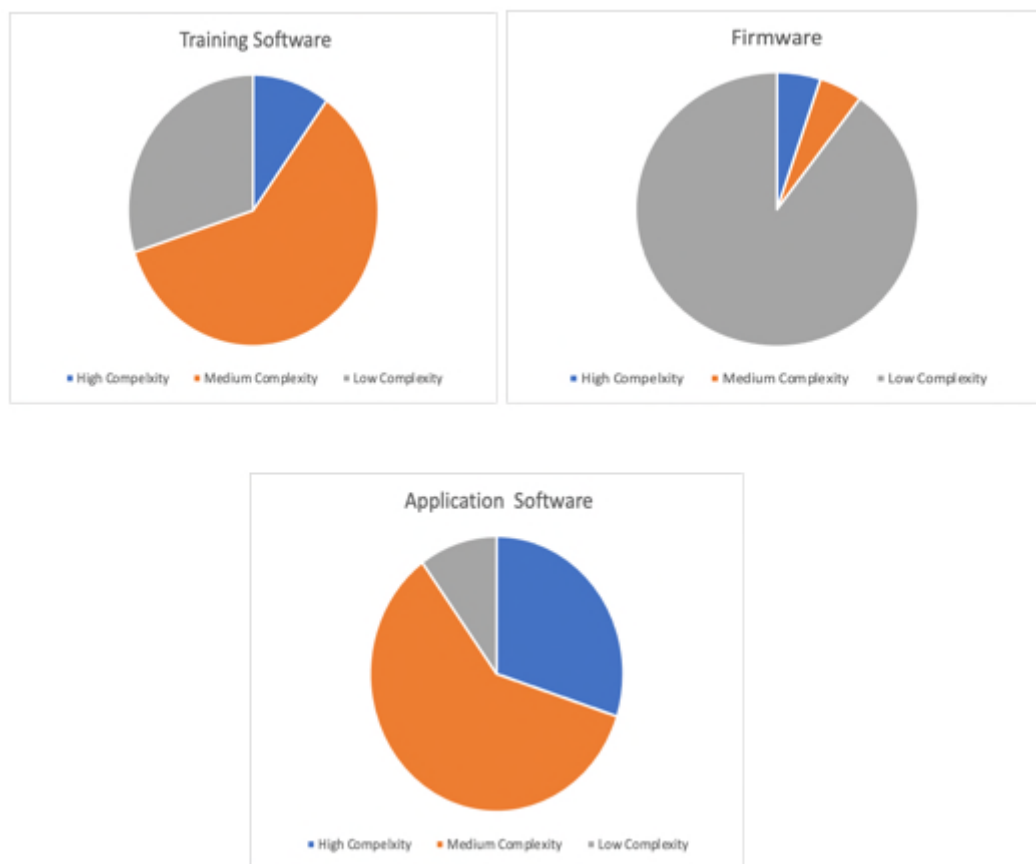


Figure 8-16 Software Obsolescence Complexity Matrix – Case Study

Several applications within the software were classed to be at a high complexity for software obsolescence.

The framework predicted a total of 5 software obsolescence issues, of which majority of them were due to update within the firmware and application software. The framework predicted the earliest the software will go obsolete is in year 5, due to the planned mid-life upgrade. Each application would go through several obsolescence issue/s at different stages of its life cycle. Resolution approaches were identified for each potential obsolescence issue as detailed in Table 8-23 below.

Table 8-23 potential obsolescence issues identified within case study E

Years	Year 1	Year 2	Year 3	Year 4	Year 5	Year 6	Year 7	Year 8	Year 9	Year 10
Resolution Approach										
Functional Resolution										
Technological Resolution										
Logistical Resolution										

To estimate the cost, following factorial (Table 8-24) was applied to the estimating formula.

Table 8-24 Matrix for Case Study E

Descriptions	Inputs	Values
Type of platform	Land	2
Level of integration	High	3
Testing and requalification	Low	3
Level of modification	Low	1
Number of applications	35	1

As a result, the Software Obsolescence Cost Analysis Framework predicted the Software Obsolescence resolution cost at £3.1M. This result was compared to the project teams technical estimate which was at £2.9M. Project team's estimate was developed by the technical team which was largely based on the resource requirement, sub-contracting activity, and any third-party licencing requirement. The project team confirmed that they were optimistic on the resource estimate

due to lack of data and the process by which they could estimate the cost of the software obsolescence.

8.3.4 Case Study Summary

The result of all the five case-studies were compared against the budgeted or the contractual estimates as summarised in Table 8-25 below. As a whole, the cost agreed in the budget was c4% higher than the estimated software obsolescence resolution cost. As a result of the verification and demonstration of the Software Obsolescence Cost Analysis Framework, the experts from within the various project teams concurred that the approach and the calculation was valid, and the outputs consistent.

Table 8-25 Summary of the findings against the budget

Case Study	Contract Duration	Estimated Software Obsolescence Resolution Cost	Contractual/budgeted software obsolescence Cost	Difference
Project A	10	£7.5M	£6.2M	£1.3M
Project B	10	£5.8M	£5.1M	£0.7M
Project C	7	£18.5M	£20.1M	-£1.6M
Project D	7	£4.8M	£5.2M	-£0.4M
Project E	10	£3.1M	£2.9M	£0.2M

The experts agreed that the software obsolescence cost analysis framework was fit for purpose in the absence of any other framework. The experts agreed that the framework was consistent in approach and follows a process.

8.4 Cross- Case Study Analysis

Table 8-25 shows the results of quantitative demonstration of the framework, by which the estimated cost was compared against the actuals. A cross- case study analysis was carried out as a workshop, which was undertaken in QinetiQ with

expert representative from each of the case study project teams were invited. The format of the workshop is detailed below in the Figure 8-17 and followed the same principles as the other workshops.



Figure 8-17 Approach taken at the Case Study Demonstration

The details of the experts who attended the workshop are listed in Table 8-26 below.

Table 8-26 Roles and experience of experts

Experts	Roles/Responsibility	Years of Experience
Expert 1	Technical Leader- Software Development	38
Expert 2	Senior Software Engineer	28
Expert 3	Principal Software Engineer	20
Expert 4	Group Leader QAT	32
Expert 5	Principal Radar Scientist	20

The workshop started by introducing the Software Obsolescence Cost Analysis Framework and Software Obsolescence Cost Analysis Framework followed by an extensive Q&A where researcher was able to clarify the validator's questions. As the experts wanted to work with the framework in real time, the validators were

given feedback questionnaire, which they were allowed to take back with them, along with the Framework and the model following the Q&A. The feedback was sent over to the researcher via email two weeks post the workshop. In the meantime, the researcher was able to answer and clarify several questions raised. The feedback was collected and analysed by finding the common elements, such as the things they liked about the framework, what they thought was not relevant and any issues they found with the framework.

The key summary of the findings is listed in Table 8-27 below.

Table 8-27 Key summary of the findings

Validators	List of what the expert liked	Issues experts raised	What the expert thought was not relevant
Expert 1	1. Raises the awareness of Software Obsolescence 2. It supports Software providers with a mechanism to understand and quantify their risk 3. Provides the end user with visibility of their Obsolescence risk profile	1. The data flow needs to be more streamlined 2. There should be a stage to determine whether software obsolescence need to be considered for low value projects 3. Outputs are only as good as the quality of the inputs	1. Not known

		and a lot of guidance will need to be provided to users	
Expert 2	4. The comprehensive literature reviews, case studies and literature that have gone into producing this framework 5. Using the model will give a better idea of how applicable it is against different software projects 6. All the key cost drivers appear to be present.	4. The data collection phase of the framework comprised of 150 questions- this might be a bit too much at the bid stage. 5. The data flow in the framework appeared logical but would need to be tested proactively on a project with a feedback loop. 6. It is not clear how generally applicable the various factors in the cost and	2. The framework will have implications for purchasing and design of software projects, however it will not drive the changes across the supply chain

		optimisation formulas will be	
Expert 3	7. Very logical to understand	7. No issues	3. Not known
Expert 4	8. Very logical 9. Suitable to apply at the bid stage 10. Can be used at all stages of the software development life cycle	8. At bid stage sometimes there are unknowns that need to be taken into accounts. 9. More detail and context required would help to apply the framework	
Expert 5	11. This is a very comprehensive framework 12. It forces all involved in the software project to consider obsolescence at the very start of the project 13. All the key cost drivers are very comprehensive	10. Only limited time spent on the framework validating however if it was able to run the framework on a worked example, it would flush out any required	

	with good level of details	updates to the framework	
--	-------------------------------	-----------------------------	--

8.4.1 Detailed analysis of the feedback

In this section, the researcher will describe the detailed analysis of the data collected during the case study demonstration exercise.

The first four questions within the feedback form were high level data about the project such as the name of the project, organisation, the role, and years of experience.

Question 5 – Is the flow of information in the framework logical?

The validator was asked to mark from one to ten, where one was totally invalid and 10 being totally valid. Score between 2 and 5 meant they were valid with a major change or update and if the score was between 6 and 9, this would mean the framework was valid with minor changes or updates.

Validator	1	2	3	4	5	6	7	8	9	10
Expert 1								*		
Expert 2									*	
Expert 3								*		
Expert 4										*
Expert 5										*
Overall Score									*	

The average score for this question was 9 which meant that the flow of information in the framework was logical but needed some very minor changes. The changes highlighted were textual changes which was rectified in the final output.

Expert 1 **suggested that there needs to be more data to properly validate the framework and this required the company to invest in time and effort in collecting and maintaining the data for software obsolescence.**

Expert 4 **suggested not agreeing with the framework, due to the opinion that the topic was well advanced but agreed that the framework was very logical and made the expert think about software obsolescence more logically.**

Expert 3 **suggested that there needs to have minor changes in the flow of the information, and to have a flow chart that would provide the working of the framework would be very handy.**

Question 6 - **is the framework suitable at the bidding stage?**

The average score for this question was 8. All the project experts except Expert 2 scored 8. The main concern for Expert 2 was that there may not be enough time at the bidding stage to incorporate this framework or any output from the model, especially if there was not enough data available. Expert 1 commented that unless it's specified at the bid stage by the customer this may be overlooked at. To make this as a standard operating procedure, there needs to be a policy change where Project Managers are mandated to investigate it. Expert 3 admitted that making this mandate would help reduce the impact of software obsolescence at a later stage.

Validator	1	2	3	4	5	6	7	8	9	10
Expert 1								*		
Expert 2						*				
Expert 3								*		
Expert 4								*		
Expert 5								*		
Overall Score								*		

Question 7 – Can this framework be used at all stages of all software development life cycle?

All the project experts confirmed that this framework can be used at all stages of the software development life cycle. They also confirmed that this can be used for all the different types of development life cycle be that Waterfall, incremental or Agile.

Validator	Yes	No
Expert 1	*	
Expert 2	*	
Expert 3	*	
Expert 4	*	
Expert 5	*	
Overall Score	*	

Question 8 – Is the framework generalisable within the Defence and aerospace sector?

All the experts except expert 4 answered the question. The researcher therefore reached out to the expert about the question who confirmed that the framework can be used in Defence and aerospace industry. All the project teams confirmed that this is a valid framework which can be used within various sectors of Defence and aerospace. Expert 2 and Expert 5 was of the opinion that this framework could also be used in automotive, oil and gas and other areas where the life cycle of the software is more than five years.

Validator	Yes	No
Expert 1	*	

Expert 2	*	
Expert 3	*	
Expert 4		
Expert 5	*	
Overall Score	*	

Question 9- is the framework generalisable for other types of software?

All the experts answered the question except expert 4 who confirmed on the follow up call that they don't have enough experience to comment on this question. All the project teams confirmed that this framework can be used for other types of software such as Engineering Software, Support Software and Automated Information Systems. Expert 2 and Expert 3 agreed that this framework can also be used to manage COTS software to a certain extent, however data calibration would be needed to manage COTS software.

	Engineering Software		Support Software		AIS	
Validator	Yes	No	Yes	No	Yes	No
Expert 1	*		*		*	
Expert 2	*		*		*	
Expert 3	*		*		*	
Expert 4						
Expert 5	*		*			
Overall Score	*		*		*	

Question 10 – To what degree could the framework benefit a Software Project team/bidding team?

The experts were asked to score between 1 and 10, where a score of 1 meant “not beneficial” and 10 being “highly beneficial”. If the score was between 2 and 5 then the framework has got little use and if the score was between 6 and 9 then the framework is quite beneficial. Again, Expert 4 failed to answer this question and confirmed on the follow up meeting that they don’t have enough experience to answer this question. All the other experts confirmed that the framework is quite beneficial for the software project team at the bidding stage, however raised concerns that there may not be enough data to produce a robust estimate.

Validator	1	2	3	4	5	6	7	8	9	10
Expert 1										*
Expert 2							*			
Expert 3								*		
Expert 4										
Expert 5									*	
Overall Score							*			

Question 11- What are the potential limitation and challenge in using and implementing the framework?

Experts 1, 2 and 5 confirmed that the lack of software obsolescence awareness will present itself as a potential limitation in implementing this framework. All the project teams also confirmed that other reasons such as lack of method to monitor software obsolescence, lack of stakeholder engagement on this topic as well as lack of software obsolescence data plays an important part in limiting this framework to be implemented.

	Lack of S/W Obs Awareness		Lack of methods to monitor S/W Obs		Lack of Stakeholder Engagement		Lack of Software Obsolescence data	
Validator	Yes	No	Yes	No	Yes	No	Yes	No
Expert 1	*		*		*		*	
Expert 2	*			*	*		*	
Expert 3			*					
Expert 4								
Expert 5	*		*		*		*	
Overall Score	*		*		*		*	

Question 12- Will this framework have potential implication across the software supply chain?

All the project team experts apart from Expert 2 confirmed that this framework will have potential implication across the supply chain. However, Expert 2 argued that the framework will have a high implication for purchasing and design decisions at an individual project level but will not drive the change across the supply chain. However, experts confirmed that good communication and proactive management of software obsolescence will drive this change.

Validator	Yes	No
Expert 1	*	
Expert 2		*

Expert 3	*	
Expert 4		
Expert 5	*	
Overall Score	*	

Question 13- What are the strongest feature of the framework?

This was discussed extensively at the workshop and the following comments were recoded both at the workshop and in the feedback.

- This framework raises the awareness of software obsolescence
- This provides a mechanism to understand and quantify the software obsolescence risk
- This framework provides the end users with a visibility of their software obsolescence risk profile
- It forces the software project team to consider software obsolescence at the very start of the project
- The comprehensive literature reviews, case studies and interviews that were utilised to produce this framework
- The framework was very logical to understand
- This was a very comprehensive framework

Question 14 – What are the weakest features of the framework?

This was discussed extensively at the workshop and the following comments were recoded both at the workshop and in the feedback

- Outputs are only good as the quality of the inputs. A lot of guidance would need to be provided to users on how to ascertain their inputs.

- The scale factors and the linear equations needs to be tested further with more data once it is available.

Question 16- Does the framework provide enough initial information?

Experts 1, 2 and 4 suggested that the framework does not provide enough initial information. Following observations were noted.

- For an inexperienced user of the framework, a lot of supplementary information and guidance needs to be provided.
- It was recommended to provide a series of use cases covering various product types alongside the framework.
- More detail and context needs to be provided that would help using the framework accurately.

Question 17- Are the terminology and the concepts used in this framework consistent?

All the experts confirmed that the terminology and the concept in this framework was consistent.

Validator	Yes	No
Expert 1	*	
Expert 2	*	
Expert 3	*	
Expert 4	*	
Expert 5	*	
Overall Score	*	

Question 18- Are all the key cost drivers considered in this cost estimating framework?

All the experts agreed that all the key cost drivers were considered. Following comments were captured during the workshop and in the feedback forms.

- Very comprehensive list with a good level of detail
- All the key cost drivers appear to be present. A concern may be that the relative weighting of those drivers could vary from project to project.

Validator	Yes	No
Expert 1	*	
Expert 2	*	
Expert 3	*	
Expert 4	*	
Expert 5	*	
Overall Score	*	

Question 19 – Is the list of resolution techniques indicated in the framework comprehensive?

All the experts apart from Expert 1 confirmed that the resolution techniques used in the framework was very comprehensive. Expert 1 confirmed that they don't have the knowledge to answer this question.

Validator	Yes	No
Expert 1		*

Expert 2	*	
Expert 3	*	
Expert 4	*	
Expert 5	*	
Overall Score	*	

Question 21/22/23/24 – Is the Software Obsolescence Management level, Complexity level, Obsolescence resolution approach, Obsolescence Key Cost Drivers defined in the framework valid?

All the experts agreed that the different parts of the Software Obsolescence Cost Analysis Framework be that Management Levels, or the resolution approach are well defined and very valid. However, Expert 4 advised to undertake more case studies to gain more confidence in the framework. Expert 5 and Expert 2 agreed that there is a substantial lack of data on software obsolescence and expressed view that there should be more research undertaken to develop a monitoring tool/model to collect data on software obsolescence.

Validator	Yes	No
Expert 1	*	
Expert 2	*	
Expert 3	*	
Expert 4	*	
Expert 5	*	

Overall Score	*	
------------------	---	--

The feedback from the first demonstration workshop and the case study workshop converged and reached a saturation point and therefore further change to the framework was deemed not necessary.

8.5 Summary of Observation

In this chapter, verification and demonstration of the software obsolescence cost analysis was presented. A total of five demonstration case studies with live projects were undertaken with the support of five software experts from Defence industry.

In the second section of this chapter, the research methodology followed for the demonstration of the framework was presented, which also included the descriptions of the experts who took part in the demonstration.

In Section 8.3, the researcher presented the qualitative demonstration of the five experts with details of the five-case study was presented. The feedback was collated and analysed and presented in this section. The collective demonstration feedback was also presented in this section.

In the next chapter, the author describes the conclusions and how the research gaps have been addressed.

9 Conclusion

9.1 Introduction

The aim of this research, as set out previously, was as follows:

“To develop a software Obsolescence Cost Analysis Framework to predict the Software Obsolescence Resolution cost of custom build software at early stages of the product life cycle.”

And the objectives were,

1. To develop a list of sources of software obsolescence for a bespoke defence real time software.
2. To develop key cost drivers and cost elements for software obsolescence resolution.
3. To develop a framework to support the management, identification, and resolution of software obsolescence.
4. To develop a cost model to estimate the cost of software obsolescence resolution for a bespoke real time software.

The conclusions drawn relating to each of these objectives will be presented in turn, following which areas identified by this research as targets for future study will be discussed.

The first objective was to develop a list of sources of software obsolescence for a custom build Defence real time software, to plan for the resolution approach during the support contract.

- The researcher undertook a cognitive case study to determine the source of the software obsolescence, how software obsolescence happens and if there was a way to predict this at very early stages of the software development.
- The cognitive case study concluded that the cognitive behaviour of the software programmer/developer was highly motivated and dictated by the available technology, internal policies, and legal requirement.

- It was also evident that the software programmer selected the technologies based on the future availability of the support and maintenance requirement.
- It was noted that there was a direct link between technology selection during the development of software and the possible resolution approach that should be in place to reduce the software obsolescence during support contract.
- Another key observation that came out of this cognitive case study was that there was a need to have stable requirement documents that specify the technology selection and various other internal and external requirement. This would help in planning for software obsolescence at early stages and to predict the possibility of software going obsolete. This will also help in estimating the cost of software obsolescence resolution.
- There was a general lack of standard process, procedure, and guidance to manage, monitor and resolve software obsolescence, although, JSP 886 and BS62402 described the process for managing obsolescence issues in general.
- It has been noted that there was a serious lack of understanding about the resolution cost involved in the resolution of software obsolescence issues.
- The software does not go obsolete on its own, however the systems that host the software would become obsolete which will make the whole system obsolete.
- Software obsolescence was generally disregarded and not managed at all in industry due to lack of maturity on this topic.

The second objective was to develop the key cost drivers for software obsolescence resolutions of custom build real time software. The researcher identified that

- There were five software obsolescence key cost drivers
 - o Resolution Approach
 - o Level of integration

- o Testing and requalification
- o Level of modification
- o Number of applications

The third objective was to develop a framework to estimate the cost of software obsolescence resolution. The researcher achieved this objective:

- By developing software obsolescence management level in order understand the current level of competence in managing software obsolescence at project or organisation level. The software obsolescence management level provides the guidance to improve the management of software obsolescence issue. This will also help to determine the software obsolescence management strategy which will in turn help in determining the appropriate resolution strategy required.
- By developing software obsolescence resolution approach to determine the appropriate resolution strategy to tackle software obsolescence. The resolution approach selected will determine the cost of software obsolescence resolution. The resolution approach was determined based on the project and system parameters with software management level resolution technique identified for individual software components rather than software program.
- By developing Software Obsolescence complexity matrix to determine the complexity of the software obsolescence issue. Software obsolescence complexity matrix helped the project manager in undertaking appropriate resolution approaches based on the complexity. It determined that highly complex software component required proactive management of its obsolescence issues.

The final objective was to develop a cost model to estimate the cost of software obsolescence resolution for a bespoke real time software. Based on the literature review and information collected from various organisations in the Defence and aerospace industry both in UK and US, the researcher was able to exhibit that:

- The research on software obsolescence is at an infancy stage and the focus on software obsolescence was largely centred on software maintenance, reliability, upgrades, and midlife upgrade.
- It was noted that there was a considerable lack of research on software obsolescence
- None of the existing cost models that the researcher looked took cost of software obsolescence into account and treated the problem as a risk which was passed on to the OEM.
- The researcher was able to develop a cost model that estimated the cost of software obsolescence resolutions and incorporated the key cost drivers/cost element into the cost model.
- It needed to be noted that a provision for risk and uncertainty analysis was made in the model, however, due to lack of any data the researcher was unable to undertake the analysis.

9.2 Research Limitations

9.2.1 Scope

Initially the researcher looked at obsolescence in all types of bespoke software in defence and aerospace industry, however it was soon identified that this would be a mammoth undertaking given software in general takes different types and forms with several applications. Software dominates both commercial and defence world with increasing dependency. As such this research then focused on the critical types of software that were heavily used in defence, that is the real-time software which were either bespoke or costume build. This narrowed down the research considerably and helped in to focus on the delivery of the aim and objectives of this research.

This also meant that several de scoping activities were undertaken to narrow down the research question. De scoping was undertaken either due to several robust research that already been undertaken and or due to substantial lack of historic or current data. Following activities are out of scope for this research.

1. Software and hardware integration cost- this is extensively covered and researched in academic and commercial world. There are several existing cost estimating tools such as Price TP and SEER.
2. Software Quality, Reliability and maintenance –related cost drivers – this is again extensively covered and researched, and several commercial tools have got capability to estimate this cost such as Price TP and SEER.
3. Uncertainty assessment and analysis in the cost model – The framework developed as part of this research does not undertake an uncertainty analysis due to lack of availability of any historic or current data on software obsolescence. However, provision has been made in the cost model if the data becomes available in the future.

One of the key limitations of this research was that it focuses on only one type of software, however this limitation was mitigated by developing the software obsolescence resolution framework that was more applicable and generalised. This was validated by the industry experts during the demonstration phase of this research.

Several assumptions were made in the development of Software Obsolescence Cost Analysis Model and the framework, and this in- turn paved for possible limitations. Some of the limitations are listed below.

- This model does predict the cost of software obsolescence resolution during a contracted period, not necessarily predict when the software obsolescence will happen. This framework estimates the cost at an engineering level, which was one of the reasons why year on year escalation of cost due to inflation was not considered. However, inflation and NPV (Net present Value) would need to be considered as a further enhancement to the model to convert the cost into Price.
- This model predicts the cost of software obsolescence resolution during a support contract and assumes that there would not be any mid-life tech refresh and any major midlife upgrades.
- The other limitation was that the model/framework needs to be used by an expert in software obsolescence, given the presence of some key cost

drivers and as such linking it to selection of a particular resolution approach, level of integration etc. that are highly sensitive to cost. Due to these reasons the need for a Software Obsolescence expert is highly mandated.

9.2.2 Access

Software obsolescence is something that is not monitored or managed as business as usual in the defence and aerospace industry as evident from this research.

A major issue was identified during the early phase of the research mainly at the data collection phase. This issue was largely due to the qualitative nature of the research whereby much information was collected and filtered through keywords etc. and therefore, would be prone to subjective interpretations. To mitigate this potential bias, the researcher completed the data collection by having a feedback loop, whereby the results were communicated back to the participants/experts from whom the data was collected for demonstration.

Another key limitation was the availability of the experts on time as software obsolescence experts were extremely difficult to find, however the researcher was able to find obsolescence experts who dealt with the software obsolescence issue in various forms through networking at the IIOM and DMSMS workshops and therefore was able to arrange meetings and workshops. To make sure that the experts were available for the workshop and meeting on time every effort were made to ensure that the following measures were taken

- A detailed proactive planning of the workshop and case studies were undertaken
- Collaborated with the experts well ahead in time to ensure that they have a good understanding of the requirements for the case studies
- Use of multiple sources for data collection and maintaining audit trail by collecting the evidence through consistent note taking and wherever possible collecting audio and video recording of the meetings/interviews and workshops especially the cognitive case study.

- Setting feedback sessions with the experts to validate the findings

9.2.3 Implementation challenge

Software obsolescence cost analysis framework is a novel idea and very new. There are standards such as BS 62402 and JSP 886 which reflect on best possible ways to manage obsolescence. However, this has not mandated, as such implementing this framework is crucial to manage, implement, identifying the resolution approach and to estimate the cost of the same.

Following are the key implementation challenges that are identified.

- Availability of the data
- Types of data that needed to be collected such as software obsolescence monitoring data etc
- Who will collect the data – suppliers/contractors or customers
- Development of the supplier and buyer relationships in managing software obsolescence and data collection
- Contractual inclusivity – this needs to be business as usual moving forward
- Organisational and key stakeholders buying

9.2.4 Other limitations

Other limitations that have been identified are as below-

- For Delphi method – the key limitation has been the sample size. Only 15 experts participated in each round of the Delphi workshop. In the first workshop each experts provided data related to different elements of the framework- this can be scrutinised and argued that the uncertainty is high, however in the second Delphi workshop where participated experts validated the initial findings which mitigated this limitation to a certain extent.
- The experts who took part in the workshops and case studies were from various profession such as software engineers, supply chain managers, obsolescence managers, cost engineers etc. This was a key limitation given their different point of views about the software obsolescence

resolution approach and cost estimation. A detailed trend analysis and refinement helped mitigate these limitations. This ensured that the framework was coherent and any deviation from this limitation been rectified.

- Another key limitation was that there was no actual cost data collected and analysed and therefore the validity of the study could be questioned. This was due to lack of availability of any historic data; however, this was mitigated by following a systematic approach to the development and implementation of the research methodology by bringing in various obsolescence experts from different organisation with different viewpoints and then streamlining and refining the findings. It was suggested that in the future, these software obsolescence cost matrices could be revalidated using actual cost data as and when the OEM starts collecting and storing software obsolescence data, as it would increase the reliability of the framework.
- The ideal approach to validate the framework for each case study would have been to compare the outcome with the actual cost from the project, however, this was not possible as the project of which the data was used was either at early stages of the maturity or it was an ongoing project which meant the researcher would have to wait for at least seven to ten years to be able to compare the results with the actual cost data. This was due to lack of data on software obsolescence that could be accessed for use within the case study. It would be extremely laborious and expensive for the OEM/sponsors to undertake this activity. Therefore, the majority of the comparison within the demonstration was done against the estimated cost that was modelled in house at the system level and compared against the SME's resource cost to mitigate the issue at the system level.

9.3 Key Contribution

The key contributions of the research are listed below,

- Develop a cost estimating process to estimate the cost of software obsolescence resolution.

- Identify the cause and reason for technology selection by the software engineers which had an effect on software obsolescence by deploying cognitive case study methodology.
- Develop key cost drivers/cost elements for software obsolescence resolution.
- Develop the process that helped to identify, manage and mitigate software obsolescence.

This research, through engagement with the industry and literature review has identified the ever-increasing issue of software obsolescence especially within the Defence and aerospace industry. However, the limited research in this area has shown the growing need for identification and management of software obsolescence. This research has therefore successfully addressed this knowledge gap by developing the Software Obsolescence Cost Analysis Framework and the subsequent Cost Model.

9.4 Future Research

Software obsolescence is not widely understood and managed appropriately due to lack of understanding and availability of data; however, this research opened several possibilities of future research not only on software obsolescence management but also on the cost estimation of software obsolescence resolution. One of the very first recommended future research would be on the demonstration of the cost model/matrix with the actual data provided sufficient information/data exist. As said, this may not be easy, given the serious lack of historic data including academic, literature and commercial.

Further work is also required on understanding, collating, normalisation, and distribution of the software obsolescence data. This could be done by developing custom build software obsolescence tools, processes, and methods. This will help in the identification and predication of the software obsolescence at very early stages of the project life cycle.

This research focuses on the real time custom build military software and there is a scope for research in software obsolescence issues within COTS software or off the shelf software.

Future academic research should also focus on the development of the detailed correlation between software obsolescence and software maintenance. Initial findings and observations are detailed in this research, however more research needs to be undertaken to improve the understanding.

And finally, it is suggested that future research on this framework and the cost model may address the limitation of the current framework, including the one discussed above, or the previous section would make the framework more robust and reliable.

In summary, this thesis has achieved the stated aim and objectives by demonstrating that the cost of software obsolescence resolution can be systematically estimated with the support of software obsolescence cost analysis framework at the early stages of support contracts.

REFERENCES

- AACE (2006) *Required Skills and Knowledge of Cost Engineering*.
- Ahrens, J.D., Prywes, N. and Lock, E. (1995) *Software Process Reengineering: Toward a New Generation of CASE Technology*.
- Babbie, E.R. (2007) *The practice of social research*. Thomson Wadsworth.
- Bartels, B., Ermel, U., Pecht, M. and Sandborn, P. (2012) *Strategies to the Prediction, Mitigation and Management of Product Obsolescence*. Wiley.
- Berg, B. (2009) 'A Dramaturgical Look at interviewing', *Qualitative Research Methods for the Social Sciences*
- Bev Littlewood (1987) *A Bayesian Modification to the Jelniski Miranda Software Reliability Model*.
- Bjoern Bartels, Ulrich Ermel, Michael Pecht and Peter Sandborn (2012) *Software Obsolescence*.
- Boehm, B.W. and Boehm Pdf, B.W. (1981) *SOFTWARE ENGINEERING ECONOMICS BY DOWNLOAD EBOOK: SOFTWARE ENGINEERING ECONOMICS*.
- Born, G. (1986) 'Controlling software quality', *Software Engineering Journal*
- Bowlds, T.F., Fossaceca, J.M. and Iammartino, R. (2018) 'Software obsolescence risk assessment approach using multicriteria decision-making', *Systems Engineering*, 21(5) John Wiley and Sons Inc., pp. 455–465.
- Bradley, M. and Dawson, R.J. (1998) 'An Analysis Of Obsolescence Risk In IT Systems', *Software Quality Journal*, 7(2), pp. 123–130.
- Bryman, A. and Bell, E. (2007) *Business Research Methods*. 2nd edn. Oxford: Oxford University Press.
- BS EN 62402:2007 (2007) *Obsolescence management-Application guide*.

Burns R (2000) *Introduction to research methods*. Sage Publications Limited, London.

CEBoK (2020) *CEBoK*.

Chou, Y.-C. (2013) *THE IMPACT OF TECHNOLOGY REFRESHMENT ON THE DEFENSE ACQUISITION LIFE CYCLE PUBLISHED BY THE DEFENSE ACQUISITION UNIVERSITY PROJECT ADVISER: MATT KENNEDY ENGINEERING AND TECHNOLOGY DEPARTMENT, CAPITAL AND NORTHEAST REGION, DAU THE SENIOR SERVICE COLLEGE FELLOWSHIP PROGRAM ABERDEEN PROVING GROUND, MD*.

Condra, L. (1999) *Combating Electronic Component Obsolescence by Using Common Processes for Defense and Commercial Aerospace Electronics*.

Creswell J. W (1998) *Qualitative Inquiry and Research Design: Choosing among five traditions*. SAGE Publications Inc., USA.

Davis G B (1997) *Management information systems; The Blackwell encyclopedic dictionary of management information systems*. Oxford, England: Blackwell.

Doyle, J., Viktor, H. and Paquet, E. (2009) 'Long-term digital preservation: Preserving authenticity and usability of 3-D data', *International Journal on Digital Libraries*, 10(1), pp. 33–47.

Easterby-Smith, M., Thorpe, R. and Lowe, A. (2002) *Management Research: An Introduction*. 2nd edn. SAGE publications, London.

Erkoyuncu, J.A., Ononiwu, S. and Roy, R. (2014) 'Mitigating the risk of software obsolescence in the oil and gas sector', *Procedia CIRP*. Elsevier B.V., Vol.22, pp. 81–86.

Flaherty, N. (2005) 'New parts for old', *IEE Review*, , pp. 34–37.

Formenton, D. (2020) 'ARTICLE Digital Preservation challenges, requirements, strategies and scientific output'

Francis, L. (2006) Tackling obsolescence, *Electronics Weekly*,, pp. 20–21.

Frank, B. and Morgan, R.L. (2007) 'Obsolescence_mitigation_and_management_of_electronics_test_equipment', *Autotestcon, 2007 IEEE*, , pp. 527–536.

Galliers, R. (1992) 'Choosing information systems research approaches. Information Systems Research', *Methods and Practical Guidelines*, Blackwell Science.

GAO (2009) *GAO Cost Estimating and Assessment Guide - Best Practices for Developing and Managing Capital Program Costs (GAO-09-3SP)*.

Gerasimou, S., Kechagia, M., Kolovos, D., Paige, R. and Gousios, G. (2018) 'On software modernisation due to library obsolescence', *Proceedings - International Conference on Software Engineering*. IEEE Computer Society, pp. 6–9.

Giovannoni, B.J. and Boyles, C. (2016) 'Hidden costs of unsupported software, obsolescence and non-standards; The importance and value of a multi-mission software program', *14th International Conference on Space Operations, 2016*. American Institute of Aeronautics and Astronautics Inc, AIAA.

Greene, J.C. and Caracelli, V.J. (eds) (1997) *Advances in Mixed-Method Evaluation: The Challenges and Benefits of Integrating Diverse Paradigms*. San Francisco: Jossey-Bass Publishers.

Guest, G., Namey, E.E. and Mitchell, M.L. (2013) *Collecting Qualitative Data: A Field Manual for Applied Research*. 1 Oliver's Yard, 55 City Road London EC1Y 1SP : SAGE Publications, Ltd.

Gummesson E (1991) *Qualitative Methods in Management Research*. Sage Publications, USA.

Hall, T. and Wilson, D. (1996) 'Views of software quality', *IEEE Proceedings* , 144(2)

Hecht, H. and Hecht, M. (1986) *Software Reliability in the System Context*.

Herald, T., Verma, D. and Lechler, T. (2007) 'A Model Proposal to Forecast System Baseline Evolution due to Obsolescence through System Operation',

Proceedings of 5th Annual Conference on Systems Engineering Research, Hoboken, New Jersey.

Houseman, O., Coley, F. and Roy, R. (2008a) 'Comparing the cognitive actions of design engineers and cost estimators', *Journal of Engineering Design*, 19(2), pp. 145–158.

Houseman, O., Coley, F. and Roy, R. (2008b) 'Comparing the cognitive actions of design engineers and cost estimators', *Journal of Engineering Design*, 19(2), pp. 145–158.

Howard, M.A., (2002) 'Component obsolescence – It's not just for electronics anymore.', *Proc.FAA/DoD/NASA Aging Aircraft Conference, San Francisco, CA*

Jang, E., Johnson, M., Burnell, E. and Heimerl, K. (2017) 'Unplanned obsolescence: Hardware and software after collapse', *LIMITS 2017 - Proceedings of the 2017 Workshop on Computing Within Limits*. Association for Computing Machinery, Inc, pp. 93–101.

Javier Romero Rojo, F. (2011) *DEVELOPMENT OF A FRAMEWORK FOR OBSOLESCENCE RESOLUTION COST ESTIMATION SCHOOL OF APPLIED SCIENCES*. PhD. Cranfield : Cranfield University .

Jay Mandelbaum (2012) *Software Obsolescence Considerations for the SD-22, SD-22 Diminishing Manufacturing Sources & Material Shortages (DMSMS) Guidebook*,

Jenab, K., Noori, K., Weinsier, P.D. and Khoury, S. (2014) 'A dynamic model for hardware/software obsolescence', *International Journal of Quality and Reliability Management*, 31(5) Emerald Group Publishing Ltd., pp. 588–600.

Johnsen, T., Howard, M. and Miemczyk, J. (2009) 'UK defence change and the impact on supply relationships', *Supply Chain Management*, 14(4), pp. 270–279.

Johnson, P. and Harris, D. (2002) 'Qualitative and quantitative issues in research design', in *Essential Skills for Management Research*. , pp. 99–116.

Josias, C., Terpenney J P and McLean K J (2004) 'Component obsolescence risk assessment', *Proceedings of the 2004 Industrial Engineering Research Conference (IERC)*

Kim, S., Jung, B., Han, D. and Lee, C. (2022) 'Technology Prediction for Acquiring a Must-Have Mobile Device for Military Communication Infrastructure', *Applied Sciences (Switzerland)*, 12(6) MDPI

Kingsman, B.G. and de Souza, A.A. (1997) 'A knowledge-based decision support system for cost estimation and pricing decisions in versatile manufacturing companies', *International Journal of Production Economics*, 53(2), , pp. 119–139.

Kumar R (2005) *Research Methodology: A Step-by-Step Guide for Beginners*. Sage 2005.

Leonard, J. v., Wolf, J. and Stolinski, R. (1988) 'Microcircuit parts obsolescence', *IEEE Region 5 Conference, Spanning the Peaks of Electrotechnology.*, pp. 229–231.

Li, Q. and Pham, H. (2017) 'NHPP software reliability model considering the uncertainty of operating environments with imperfect debugging and testing coverage', *Applied Mathematical Modelling*, 51 Elsevier Inc., pp. 68–85.

Linstone, H.A. and Turoff, M. (1975) *The Delphi method: techniques and applications*. Addison-Wesley, London.

Lomakin, M.I. and Murav'ev, A. v. (2016) 'Managing the Process of Re-Engineering of Information Systems Based on Integrated Monitoring of Obsolescence', *Measurement Techniques*, 58(10) Springer New York LLC, pp. 1102–1106.

Lyu, M.R. (2007) 'Software Reliability Engineering: A Roadmap', *IEEE Future of Software Engineering*

Marion, R.L. (2001) 'Mitigating COTS obsolescence in military test', *AUTOTESTCON (Proceedings) 2001. IEEE Systems Readiness Technology Conference*, , pp. 746–756.

Mehra, A., Seidmann, A. and Mojumder, P. (2014) 'Product life-cycle management of packaged software', *Production and Operations Management*, 23(3) Wiley-Blackwell, pp. 366–378.

Merola, L. (2006) 'The COTS software obsolescence threat', *Proceedings - Fifth International Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems.*, Vol.2006, pp. 127–133.

Meyer, A., Pretorius, L. and Pretorius, J.H.C. (2004) 'A model to manage electronic component obsolescence for complex or long life systems', *Engineering Management Conference, 2004 Proceedings. 2004 IEEE International.*, pp. 1303–1309.

Miles, M.B. and Huberman, A.M. (1994) *Qualitative data analysis: An expanded sourcebook*. An expanded sourcebook (2nd ed.). Sage Publications, Inc.

MoD (2014) *JSP 507 Investment Appraisal and Evaluation Part 1: Directive JSP 507 Part 1 (V6.0 Jan 14)*.

MoD (2013) *JSP 886 THE DEFENCE LOGISTICS SUPPORT CHAIN MANUAL VOLUME 7 SUPPORTABILITY ENGINEERING PART 8.13 OBSOLESCENCE MANAGEMENT*.

Moyle, W. (2002) 'Unstructured interviews: challenges when participants have a major depressive illness', *Journal of Advanced Nursing*, 39(3), pp. 266–273.

Muñoz, R.G., Shehab, E., Weinitzke, M., Bence, R., Fowler, C., Tothill, S. and Baguley, P. (2015) 'Key challenges in software application complexity and obsolescence management within aerospace industry', *Procedia CIRP*. Elsevier B.V., Vol.37, pp. 24–29.

NASA (2015) 'NASA Cost Estimating Handbook', *NASA Cost Estimating Handbook*

Nguyen, V. (2010) *IMPROVED SIZE AND EFFORT ESTIMATION MODELS FOR SOFTWARE MAINTENANCE*.

Niazi, A., Dai, J.S., Balabani, S. and Seneviratne, L. (2006) 'Product Cost Estimation: Technique Classification and Methodology Review', *Journal of Manufacturing Science and Engineering*, 128(2), , pp. 563–575.

Okoli, C. and Pawlowski, S.D. (2004) 'The Delphi method as a research tool: an example, design considerations and applications', *Information & Management*, 42(1), pp. 15–29.

Ozkan, B.E. and Bulkan, S. (2022) 'Multi-objective approach to forecast design refresh time due to COTS obsolescence', *Systems Engineering*, 25(3) John Wiley and Sons Inc, pp. 224–241.

P. Singh and P. Sandborn (2013) 'A Journal Devoted to the Problems of Capital Investment Obsolescence Driven Design Refresh Planning for Sustainment-Dominated Systems.', *The Engineering Economist*, , pp. 37–41.

P Stein and S Wady (2008) *Technology Partnership in Defence - Maintaining a cutting edge for planning and investing in Defence R&D*.

Pressman R S and Ince D (1992) *Software engineering: a practitioner's approach*.

Quyoun, A., Dar, M.-U.-D. and Quadri, S.M.K. (2010) 'Improving Software Reliability using Software Engineering Approach- A Review', *International Journal of Computer Applications*, 10(5), pp. 41–47.

Rajagopal S (2020) 'Software Estimating - is the probelm solved ', *SCAF*.

Rajagopal, S., Erkoyuncu, J.A. and Roy, R. (2015) 'Impact of software obsolescence in defence manufacturing sectors', *Procedia CIRP*. Elsevier B.V., Vol.28, pp. 197–201.

Rajagopal, S., Erkoyuncu, J.A. and Roy, R. (2014) 'Software obsolescence in defence', *Procedia CIRP*. Elsevier B.V., Vol.22, pp. 76–80.

Ralph L., K. and von Winterfeldt, D. (1988) 'Probabilities are useful to quantify expert judgments', *Reliability Engineering & System Safety*, 23(4), pp. 293–298.

Redling, T.J. (2004) 'Considerations for upgrading aging military avionics systems with state-of-the-art technology', *The 23rd Digital Avionics Systems Conference*, 2(04), pp. 7.1-7.7.

Robinson, B., Hughes, B., Bauer, R. and Harnack, J. (2015) 'Proactively Managing Obsolescence with Test System Architecture', *IEEE AUTOTESTCON*

Robson C (2002) *Real World Research: A Resource for Social Scientists and Practitioner-Researchers*. Blackwell Publishers Ltd., UK.

Romina Torres (2007) 'Mitigating the Obsolescence of Specification Model of Service based System', *Computer*, 40(11), pp. 38–45.

Rowe, G. and Wright, G. (1999) 'The Delphi Technique as a Forecasting Tool: Issues and Analysis.', *International Journal of Forecasting* 15(4), , , pp. 353–375.

Roy, R. (2003) *Cost engineering: why, what and how?*.

Rush, C. and Roy, R. (2001) 'Expert judgement in cost estimating: Modelling the reasoning process.', *Concurrent Engineering*, 9(4), pp. 271-284., , pp. 271–284.

Sandborn, P., Prabhakar, V. and Ahmad, O. (2011) 'Forecasting electronic part procurement lifetimes to enable the management of DMSMS obsolescence', *Microelectronics Reliability*., Vol.51, pp. 392–399.

Sandborn, P.A. (2007) Software obsolescence - Complicating the part and technology obsolescence management problem *IEEE Transactions on Components and Packaging Technologies*.

SCEBoK (2020) *SCEBoK*.

Schmid, E., Kosugi, G., Ibsen, J. and Griffith, M. (2016) 'Don't get taken by surprise: planning for software obsolescence management at the ALMA Observatory', *Software and Cyberinfrastructure for Astronomy IV*. SPIE, Vol.9913, p. 99131A.

Shuai, Y., Zheng, Y. and Huang, H. (2018) 'Hybrid Software Obsolescence Evaluation Model Based on PCA-SVM-GridSearchCV', *2018 IEEE 9th*

International Conference on Software Engineering and Service Science (ICSESS). IEEE, pp. 449–453.

Singh, P. and Sandborn, P. (2005) 'Forecasting technology insertion concurrent with design refresh planning for COTS-based electronic systems', *Reliability and Maintainability Symposium, 2005 Proceedings Annual.*, pp. 349–354.

Singh, P., Sandborn, P., Lorenson, D. and Geiser, T. (2002) 'Determining Optimum Redesign Plans for Avionics Based on Electronic Part Obsolescence Forecasts', *SAE Technical Paper Series*

Singh, T., Singh, R. and Mishra, K.K. (2018) 'Software cost estimation using environmental adaptation method', *Procedia Computer Science*. Elsevier B.V., Vol.143, pp. 325–332.

Siva, M., Nayak, D.P. and Narayan, K.A. (2019) 'Strengths and weaknesses of online surveys', *IOSR Journal of Humanities and Social Sciences (IOSR-JHSS)*, 24(5)

Skulmoski, G.J., Hartman F.T. and Krahn, J. (2007) 'The Delphi method for graduate research', *Journal of Information Technology Education*, 6(1), , , pp. 1–21.

Standish, T.A. (1984) *An Essay on Software Reuse*.

Tashakkori, A. and Teddlie, C. (2003) *Handbook of mixed methods in social & behavioral research*. SAGE, 2003. ISBN 0761920730, 9780761920731.

Thomas M. Pigoski (2008) *Practical Software Maintenance: Best Practices for Managing Your Software Investment*. 1st edn. whiley.

Torrado, N., Wiper, M.P. and Lillo, R.E. (2013) 'Software reliability modeling with software metrics data via gaussian processes', *IEEE Transactions on Software Engineering*, 39(8), pp. 1179–1186.

Torresen, J. and Lovland, T.A. (2007) *Parts Obsolescence Challenges for the Electronics Industry*.

Trenchard, M. (2003) 'The_obsolescence_minefield', *IEE Review*, , pp. 29–29.

US DoD (2020) *DOD COST ESTIMATING GUIDE OFFICE OF THE SECRETARY OF DEFENSE Cost Assessment and Program Evaluation*.

Vergara, D., Extremera, J., Rubio, M.P. and Dávila, L.P. (2020) 'The technological obsolescence of virtual reality learning environments', *Applied Sciences (Switzerland)*, 10(3) MDPI AG

Walsh, M. (2001) *Research Made Real*. Nelson Thornes Ltd.

Weaver, P. and Ford, M. (2003) 'ATE Obsolescence Solutions; Costs and Benefits BACKGROUND TYPICAL ATE SYSTEM', *Proceedings AUTOTESTCON. IEEE Systems Readiness Technology Conference*

Webb, L.T. and Bil, C. (2010) 'Through-life support and the aircraft lifecycle', *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*.

Weltzin, C. and Schlonsky, S. (2011) 'Reducing Obsolescence of Linux-Based ATEs with Virtualization', *IEEE Instrumentation and Measurement magazine*

Wnuk, K., Gorschek, T. and Zahda, S. (2013) 'Obsolete software requirements', *Information and Software Technology*, 55(6), pp. 921–940.

Wu, B., Jiang, Z., Zhu, S., Zhang, H., Wang, Y. and Zhang, Y. (2022) 'Data-Driven Decision-Making method for Functional Upgrade Remanufacturing of used products based on Multi-Life Customization Scenarios', *Journal of Cleaner Production*, 334 Elsevier Ltd

Wubben, H. (2007) 'DELIVERING ON OBSOLESCENCE PROTECTION', *IEEE Autotestcon*

Zaabar, I., Arango-Miranda, R., Beauregard, Y. and Paquet, M. (2021) 'A sustainable multicriteria decision framework for obsolescence resolution strategy selection', *Sustainability (Switzerland)*, 13(15) MDPI

APPENDICES

Appendix A Interview Questions

Name:

Project Team:

Role:

General

1. Who is responsible to measure obsolescence and technological maturity?
2. Who is responsible for Software reliability?
3. Who is responsible for Software Maintenance?
4. What is software obsolescence?
5. How is software project reviewed?
 - a. What was reviewed?
 - b. Who reviewed?
 - c. What were the findings and conclusions

Software Quality

1. Who performs software evaluations?
2. Who performs audit and reviews? What were the findings and conclusions?
3. What is the error reporting and tracking process?
4. What are the documents produced by software quality assurance team?

Estimating

1. Do you prepare models to estimate cost?
2. Do you focus on each stage of the life cycle separately? What kind of models are you using to make estimation? Can you tell us specific examples of software that you use?
3. How do you check a quoted price in a contract?
4. How do you predict costs at the bidding stage?

Obsolescence Modelling

1. What strategies are used to predict and mitigate obsolescence?
2. How do you define mitigation strategy?
3. How do you select mitigation strategies? Could you please explain the logic behind the selection?
4. What issues are present in this process? And how would you improve these issues?
5. How do you incorporate obsolescence into cost model?
6. How do you predict obsolescence?
7. How do you define mitigation strategy to tackle obsolescence?

8. How do you incorporate the cost related to obsolescence mitigation?
9. What methods are used to predict and mitigate obsolescence?
10. How is obsolescence included within the costing?
11. What are the challenges in incorporating obsolescence into cost estimation? How could this be improved?
12. How is obsolescence estimated at the bidding stage?
13. What issues are present in this process?
14. How would you improve this process?
15. What level of information about the project definition is available at the bidding stage, so that could be used to analyse obsolescence?
16. Which activities within in-services stage gets affected due to obsolescence and why?
17. In this project what are the cost drivers for obsolescence?
18. Do you forecast obsolescence and how?
19. What cost metrics are used?
20. Who oversees costing obsolescence at the bidding stage? How is it done?
21. What limitation do you feel you have in costing obsolescence?

Obsolescence Management

1. Is there any standard policy for the management of obsolescence?
2. Do you use any tool for monitoring the health of the software in terms of software obsolescence?
3. Who oversees managing obsolescence and solving any obsolescence issues, the customer, or the supplier?
4. who is responsible for the cost of it?
5. Who decides what the most appropriate mitigation approach is to tackle an obsolescence issue? What are the drivers of this selection? Is it done in a reactive or proactive way?
6. Which is more expensive to deal with obsolescence, reactive or proactive way? Explain?
7. Do you plan the mitigation strategy at the early stages?

COTS application software

1. What is the basis for changing version/ revision levels?
2. What is the current version/ revision of the software?
3. When was this product first available for sale?
4. Has your version/revision been discontinued?
5. How long will license agreements for this version be obtainable?
6. Will licence downgrade be available? For how long?
7. How long will support for your version be obtainable?
8. Will third party support be available after that? For how long?
9. What is the planned product end of life?
10. Is there a planned replacement product?
11. Is the planned replacement backwards compatible?
12. What are the different technical characteristics between the old and new version?
13. Which operating systems are compatible with the software?

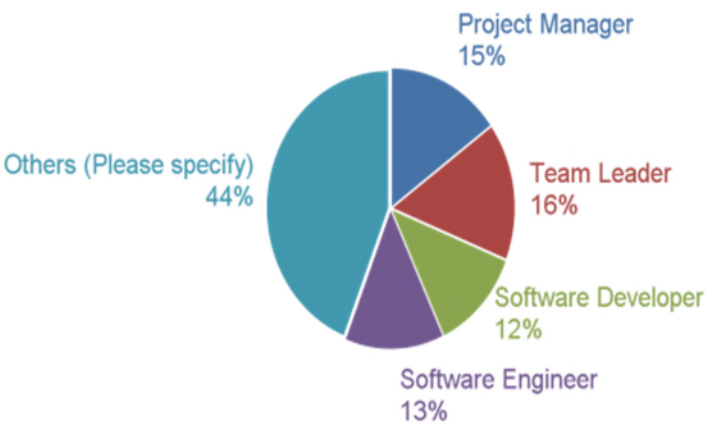
14. What are the minimum hardware requirements (if any), e.g., processor speed or memory?

Custom application software

1. What is the current version/revision of the software?
2. Could you modify the software?
3. Is the source repository maintained?
4. Are the development tools maintained?
5. Are there any third-party components?
6. Are you able to compile those third-party components?
7. What is the planned product end of life?
8. What is the planned product end of support?
9. Will third party support be available after that? For how long?
10. Is there a planned replacement product?
11. Is the planned replacement backwards compatible?
12. Which development and testing hardware and software infrastructure are needed to maintain the software?

Appendix B Detailed online survey data Analysis

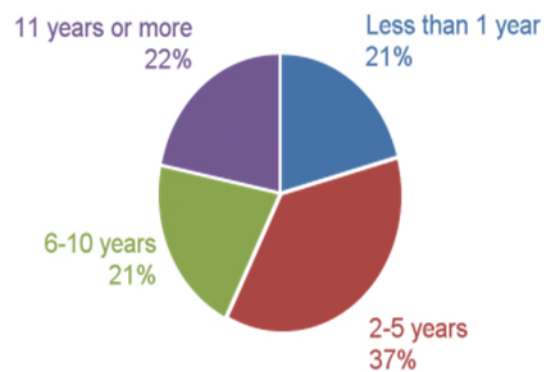
1. What is your role in the project team?



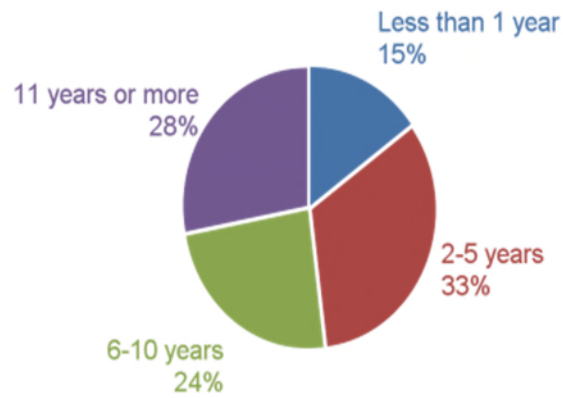
Others (Please specify)
Researcher Consultant
Obsolescence Management Lead
Cost Management Consultant
Lead Engineer
Commercial Manager
Senior Estimator
Software Reliability Engineer
OM policy Guidance & Advice
Business development and application engineering
Risk and Opportunity support

Sales
Quality assurance analyst
Business Manager (Director)
Business representative
Technical Lead
ILS Engineer
Support
Engineering Authority

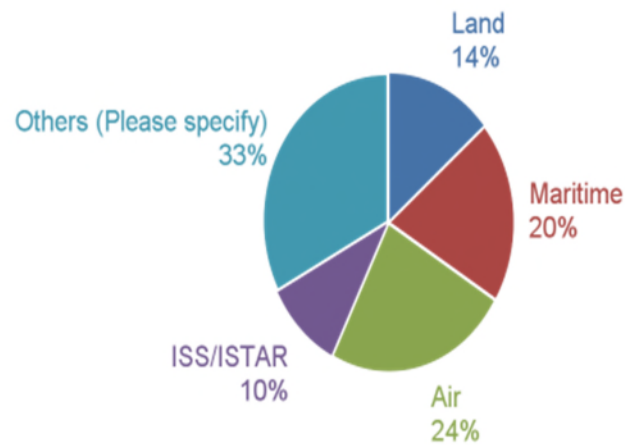
2. How long have you been in the current role managing software obsolescence?



3. What is your current level of expertise in dealing with software maintenance and software obsolescence?

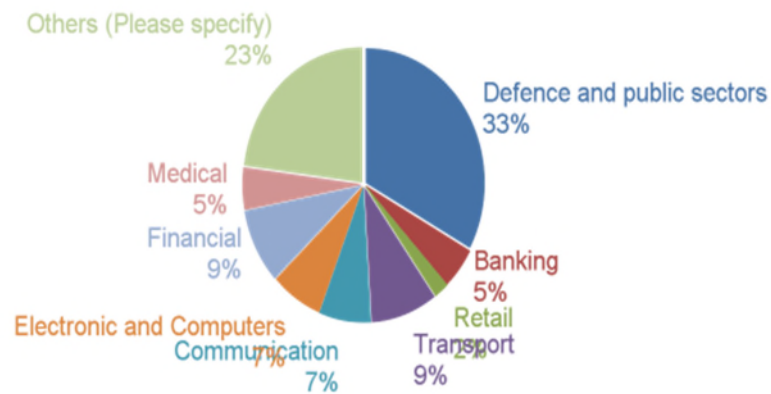


4. What is the domain/platform of your project (MoD only)?

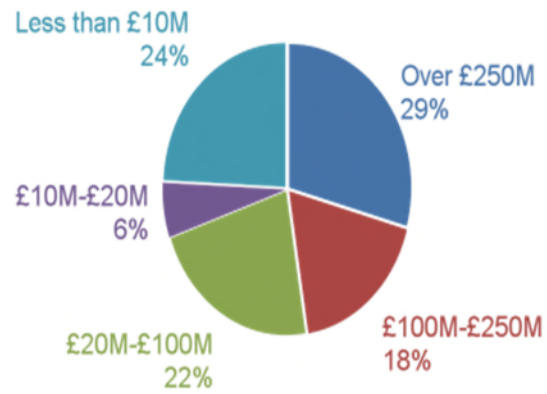


Others
CAAS
Planetary Mission/Mars Exploration
Healthcare
Mortgage domain
Marketplaces
Construction
Online Travel
Clinical
Air, Maritime, and Land

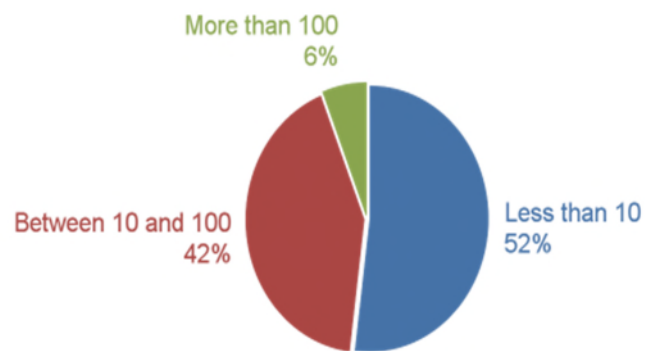
5. What is the domain/platform of your project (non-Defence)?



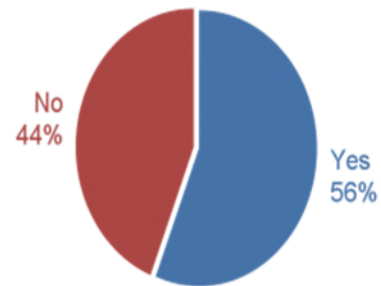
6. What is the value of your project?



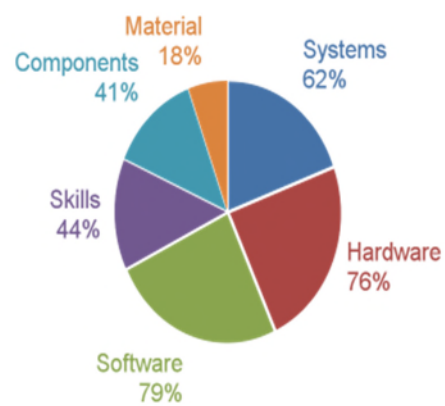
7. How many software applications have you got in your project?



8. Do you have obsolescence management strategy?



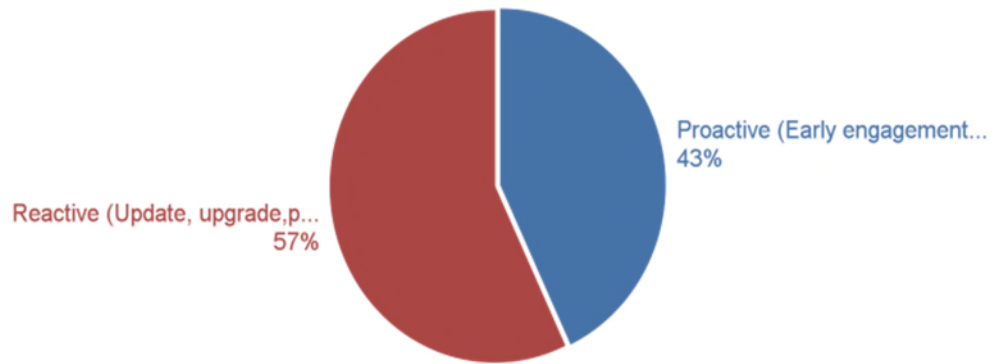
9. What does your obsolescence management strategy cover?



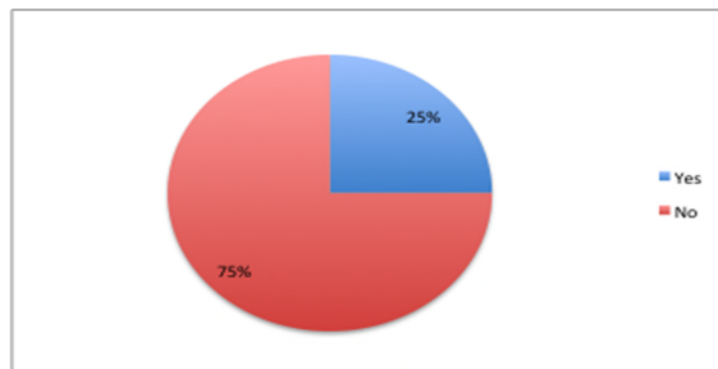
10. What does your obsolescence management strategy consist of?

Replacing hardware and software with new approved systems with the capability for been supportable (hardware, software, and diagnostic tool) for another 25 years
Hardware failure rates/ obsolescence issues, identification of potential alternatives, maintaining adequate SQEP resources
The service-oriented architecture will make this programme everlasting
Software support policy advice, guidance, and training
Obsolescence management plan integral with company's governance, functional policies, and specific process used by functions and projects
Database tracking of hardware systems and CM management of deployed software
Managing obsolescence proactively to identify as early as possible the most cost-effective mitigation/ resolution approach
Proactive and reactive strategies based on the assessment of the technologies involved in the design and the lifecycle of the equipment
Based on IEC 62402:2007
Awareness of industrial position, skills shortage, demographics, rate of technological advancement, the need to have adaptive open architecture to accommodate changes

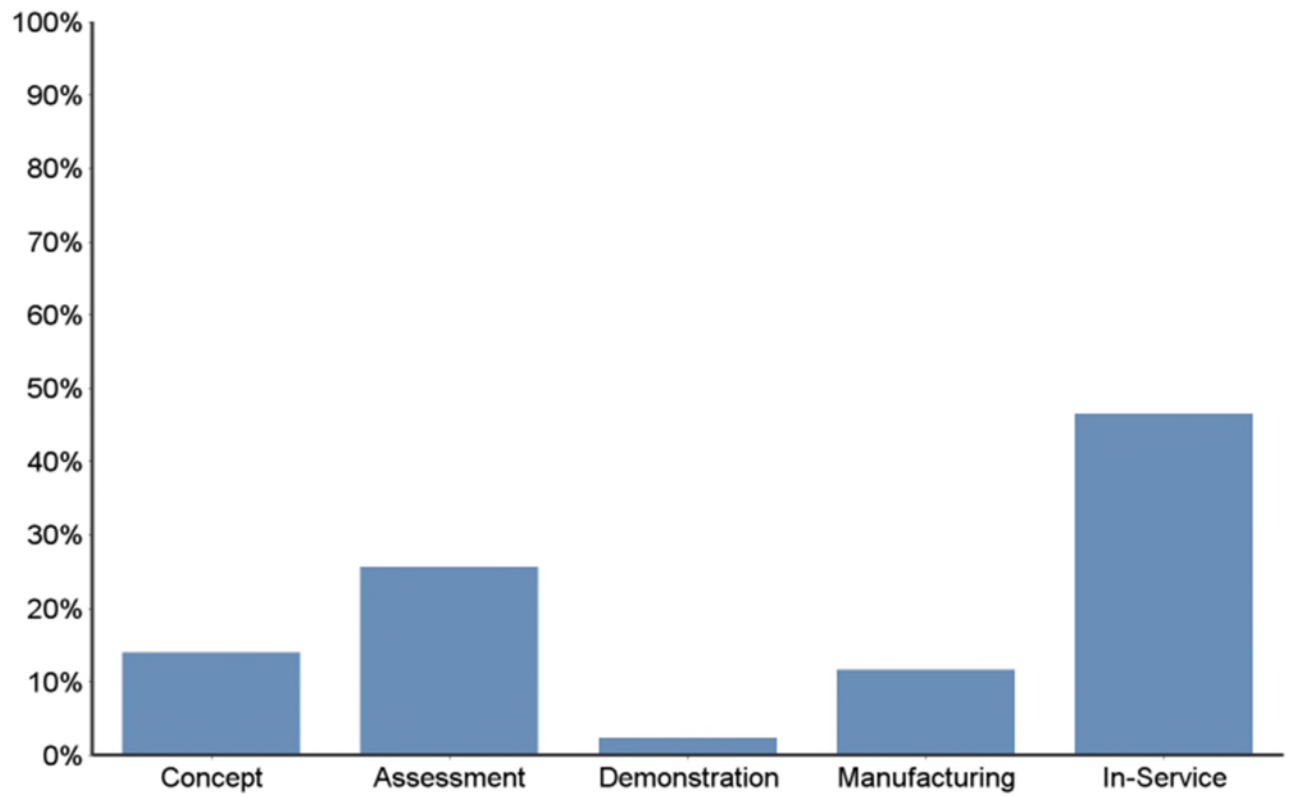
11. What software obsolescence management strategy do you use?



12. Do you follow any software obsolescence management standards?



13. When do you plan for software obsolescence?



14. Do you monitor software obsolescence?



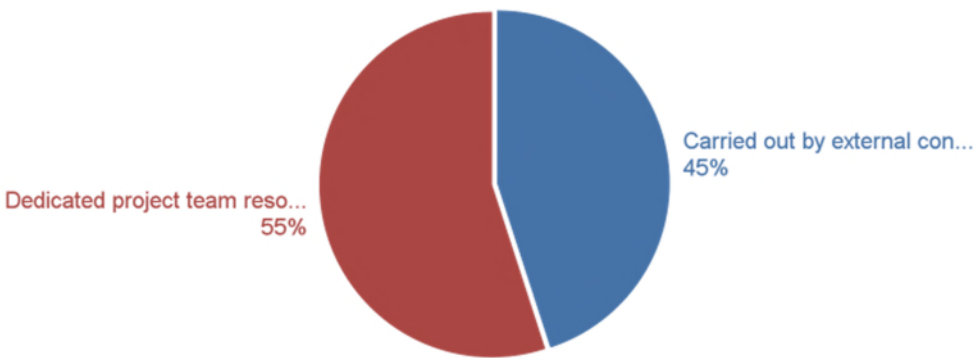
15. If yes, how do you monitor software obsolescence?

Evidence based
Direct contact with the owner/creator weather commercially available or bespoke
Progress meeting with OEM. Relay on update of OEM obsolescence management plan
Manufacturer updates and support model
Following up the software releases, patches, and the industry standards
Ensuring a software management plan is in place. Baseline regularly managed and monitored regularly
Review OEM end of life dates

16. Who is responsible for software obsolescence within your project?

Others
ILS Manager
Business Manager
IT Department
Software Reliability Manger
Support Team Manager

17. Who is responsible for the software maintenance in your project?



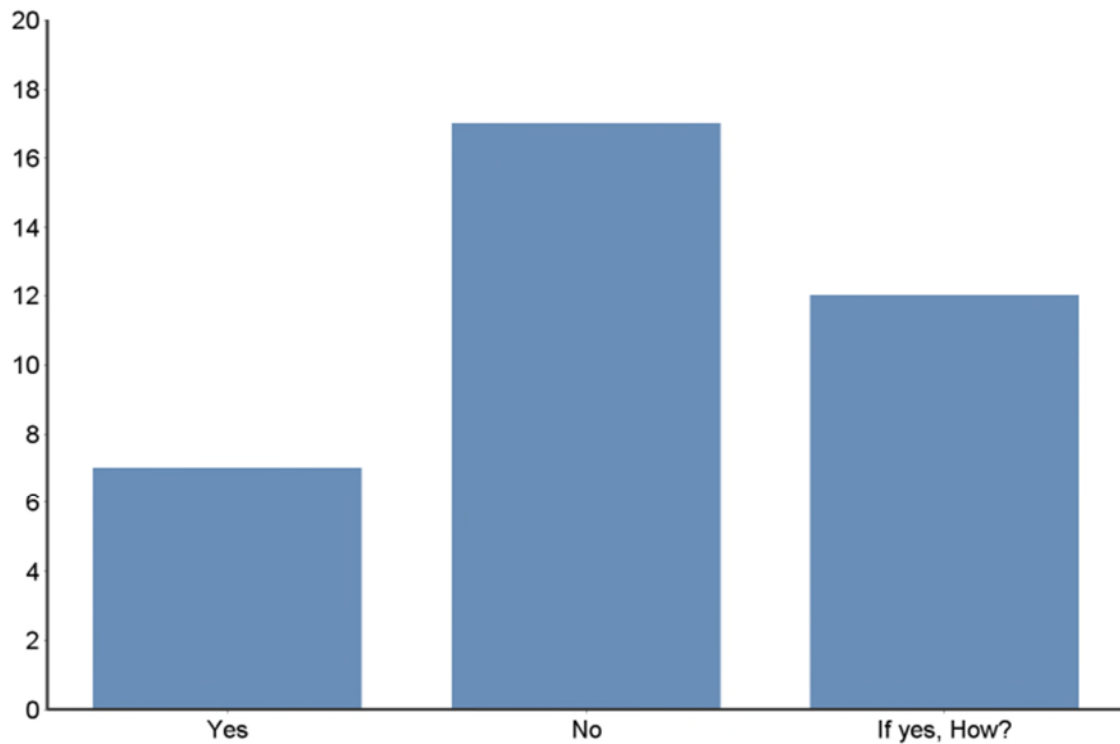
18. How is software different from obsolescence management?

Software maintenance is part of software obsolescence management and able to be done until operating system and firmware capability is exhausted and/or skill no longer available to keep maintaining.
The magnitude of the change level
Due to our support contract, there is no difference
Maintenance: - bug fixes, improvement; Obsolescence: - replacement of entire application by a new one
Software maintenance is mostly bug fixing and ensuring software continues to perform as designed. Software obsolescence management is the discipline of ensuring that the software can still be developed, maintained, and executed.
Software obsolescence management is making sure we keep current with the hardware upgrades and vendor supported software.
Software maintenance is related to the process of debugging the code and ensuring that the software is working as intended. Software obsolescence management need to be applied when application/program/operating system/compilers changes and trigger compatibility issues with other parts of the system (both software and hardware)

19. Has your hardware or system obsolescence triggered software obsolescence

If yes, how?
Cannot buy hardware anymore
We have experienced hardware obsolescence requiring software changes to enable the new hardware to run the system
The development environment, test environment, target environment, adjacent systems can trigger the software obsolescence
To maintain capability and to continue requirements within a changed host system or environment the software must be changed
When hardware becomes obsolete and is replaced with a different technology, it will usually require changes in the software, making the existing version of the software obsolete.
Old software not being compatible with the new hardware
Windows application not been supported by Microsoft
Computing hardware that can no longer be procured, software/firmware to be updated as the proposed replacement has different interface
Hardware has been delivered with the new software which is not backward compatible

20. Has software obsolescence triggered system or hardware obsolescence?



If yes, how?

Skills fade for modification

When software becomes obsolete and is upgraded, the existing hardware may not meet the requirement of the new software and then become obsolete

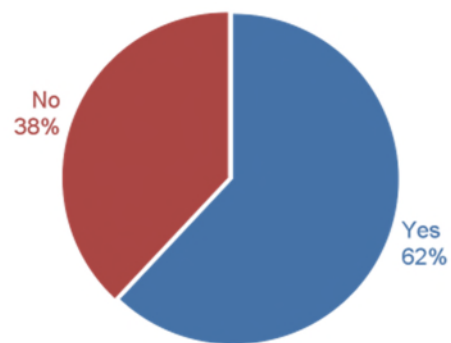
Windows 7 not been able to run on existing PC's

Old platforms been replaced with new for example VAX to PC

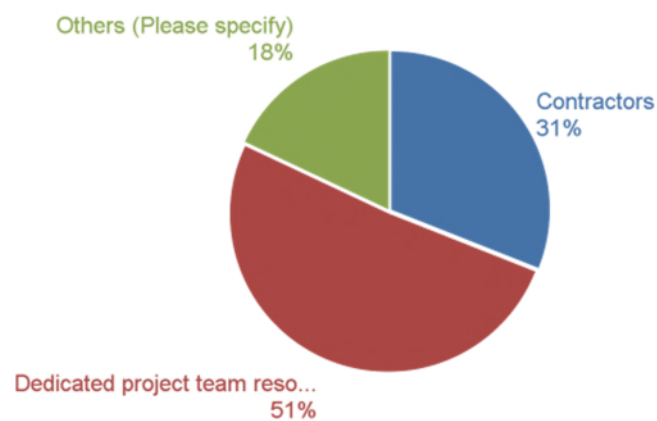
Software that can no longer be procured with support e.g., firewall requires a new version that requires a multi-core processor to provide required performance

New release of the third-party software demands a high spec hardware
Third party software such as OS or COTS/MOTS software many no longer be supported in legacy systems
The physical spec of software tends to drive the hardware upgrades

21. Do you perform software evaluation and reviews?



22. Who performs software evaluations and reviews?



Others (Please specify)
Classification society
Any of the above
Consultant employed to provide cost predictions
IT
Contractors and the project teams

23. If yes, what evaluation and review mechanism do you use?

Undertaken by system design authority supported by an engineering or project manager and specific domain experts
Software demonstration and verification along with software audit
If anything, we will have user acceptance testing of new software prior to general release
Peer review process
DO 178B and FAA SOI process, carried out by contractors with authority oversight
Mandated software support policy
Contract team software evaluations
Review of obsolescence Register
Dedicated team to track the throughput of the system and raise alarm as and when the system started reaching threshold performance

24. How do you define software obsolescence?

Software obsolescence to be the inability of software to run a system because change to the systems through replacement of parts or hardware made the software obsolete

Software does not go obsolete; it is changing in the environment, which render software unable to deliver to requirement. Software upgrades increases the complexity to a point where the build standard becomes unmanageable

The software that is no longer maintained, buggy and security risk

Software binaries cannot be inherently obsolete. Instead, software obsolescence is obsolescence of tools, languages, test equipment's etc. software should only be considered obsolete if it is no longer possible to maintain and support the software

Software suffers obsolescence when the function it performs requires updating and that update cannot be done in timely manner

The non-availability of software to support the hardware/firmware requirement

25. What is part of software obsolescence mitigation plan?

Operating system, programme update to resolve issues when peripherals are changed. Regularly research/review cost of replacement system with envisaged 25 years of supportability
Migrations
Confirmation management and software support plan
Assurance that it will be possible to upgrade, enhance, maintain, and sustain the software over the whole of the lifecycle from concept to retirement
What else is available in the marketplace
Monitoring of programming environment, languages, people, skills, and test environment
Through life support plan
ESCROW agreement (to maintain the legacy software)- System dependency mapping (to understand the knock-on effect of changes in the system)
Proactive approach
Impact on capability, safety, security etc.
Constant refresh and renewal of packages to enable operability
Identify a specific budget. Use a rule that software obsolescence will occurs within 2 years of original release

26. How do you pass on software obsolescence risks to the contractors (Supply chain)?

Please explain with an example.

Contractually OEM is required to keep the capability support for x number of years
Contract for availability, these places risk on the supplier to ensure the capability is maintained
Establish a software support contract which includes sustainment
Highlight all obsolescence risks and verify the contractor has provisioned for the risks
By giving them full control and responsibility about the system design
During deployment and maintenance
Ask the contractors to do the support work until the software usage ceases
The requirement for management is outlined in the SOW
Highlight hardware technology and upgrade frequency, scarcity of the skills available to maintain/upgrade software
Contract for the supply of output and not by product

27. How do you mitigate software obsolescence? Please list the technique used.

Customer signs off risk or provides funds
OEM engaged contractually until end of system life

Support planning. This will be dependent upon the criticality of the software. It can include third party support, ownership of the IPR, designing for changes and open software systems
No specific mitigation strategies in place
Its informal on a case-by-case basis
Hardware lifetimes buy, preserve the development and modification environment, predict skill fade
ESCROW-minimising changes to the systems and then performing technology refresh.
Proactive strategy selection to identify the software and hardware to ensure that both are available or resolved.
The obsolescence standards require an obsolescence register to be kept and monitored regularly
Keep on top of updates provided by third party suppliers
1. Track the load increases, 2. Track the number of customers using the systems, 3. Track overall performance of the systems
Lifetime buys. Lock down the system so that the new functionality can't be introduced. Spiral development where you plan to replace software regularly

28. Identify the most popular software obsolescence mitigation strategy and why?

Reactive and proactive strategies as identified in EN 62402, however there is a relationship between software and hardware and only some strategies will be valid if the software is bespoke to a particular application.
Adoption of rigorous system engineering mind-set such that software obsolescence is treated as tradable attribute along with others.
Contracting for availability
Strategic planning – to be ready for items that need to be reviewed for obsolescence
Preserve the development and modification environment
Commercially controlled
Ignore and continue

29. What are the major cost drivers of software obsolescence mitigation?

Legal claims
Support cost
The potential for plant stoppage, loss of production is the major cost drivers
Cost of ownership
Forced and unplanned expenditure on new hardware, new software, new infrastructure, resurrection of the skill base, reverse engineering of the design
Any kind of security issue

Funding available for upgrades, changes to the hardware
Skills fade
Re-qualification, testing and acceptance
Cost of maintaining a software team to make changes to software for safety, security etc.
Building new solutions
Lack of cultural awareness and willingness to act on evidence
Software code testing

30. From the cost required to solve a software obsolescence problem, can you rank the mitigation techniques starting from the most expensive one?

Skill fade, preserve the development and modification environment, hardware lifetime buys
Commercially controlled
Reactive solution, planned upgrade, contract support
Unsure
Software of unknown pedigree, single source supplier, proprietary software, in house maintenance, open-source software

31. What are the major cost drivers for proactive obsolescence management?

Internal project management costs. OEM uploading software to computers on site
The potential loss of manufacturing capability and risk to the profitability of the business
Planning for the through life change. Designing open system upfront
Sustainment of the skill base in the workforce and sustainment of infrastructure
Driven by security issues
The number of different hardware platform being used, the number of software packages running in the hardware system, the number of years the system must operate
Skill buys in
System mapping
Manpower effort
Time investment when reviewing equipment and systems
The lack of a link between project conceptualisation and adequate budget setting

32. What capabilities would you need to be proactive in software obsolescence management?

At quarterly progress meeting ask OEM to report on obsolescence and progress. Updating bespoke user and maintenance manual. Updating bespoke training equipment and information
System knowledge and the availability of up-to-date skills are essential to manage obsolescence when it happens and to proactively manage it. These skills will normally have to be brought in to resolve or assess risk
SQEP
An appropriate contractual model involving the customer, contractor, supply network etc. that recognise the cost
Financial support
Time and money
Complete system architecture
To have full understanding of system architecture to be able to target key items for monitoring. To understand the degree of software embedded in hardware to measure the impact of obsolescence if it were to occur.
Rapid availability of cost and schedule revision to test alternative mitigation strategies

Appendix C Source Code for Software Maintenance Database

```
CREATE DATABASE IF NOT EXISTS `Software_Maintenance_Support`
/*!40100 DEFAULT CHARACTER SET latin1 COLLATE latin1_bin */.

USE `Software_Maintenance_Support`.

-- MySQL dump 10.13 Distrib 5.6.24, for osx10.8 (x86_64)

-- Host: 127.0.0.1  Database: Software_Maintenance_Support

-- -----

-- Server version 5.6.25

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;

/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;

/*!40101 SET NAMES utf8 */;

/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;

/*!40103 SET TIME_ZONE='+00:00' */;

/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0 */;

/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
```

```
/*!40101          SET          @OLD_SQL_MODE=@@SQL_MODE,  
SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */.
```

```
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */.
```

-- Table structure for table `Software Maintenance Data`

```
DROP TABLE IF EXISTS `Software Maintenance Data`.
```

```
/*!40101 SET @saved_cs_client  = @@character_set_client */.
```

```
/*!40101 SET character_set_client = utf8 */.
```

```
CREATE TABLE `Software Maintenance Data` (  
  `Software ID` int(11) NOT NULL,
```

```
  `Year of Development` varchar(45) COLLATE latin1_bin NOT NULL,
```

```
  `Age of Software` varchar(45) COLLATE latin1_bin NOT NULL,
```

```
  `Total Maintenance Hours` int(11) NOT NULL,
```

```
  `Corrective Maintenance Hours` int(11) NOT NULL,
```

```
  `Total Support Hours` int(11) NOT NULL,
```

```
  `Organisation Type` varchar(45) COLLATE latin1_bin NOT NULL,
```

```
  `Industry Sectors` varchar(45) COLLATE latin1_bin NOT NULL,
```

```
  `Application Size` varchar(45) COLLATE latin1_bin NOT NULL,
```

```
  `Application Size Approach` varchar(45) COLLATE latin1_bin NOT NULL,
```

```
  PRIMARY KEY (`Software ID`, `Year of Development`, `Age of Software`, `Total  
Maintenance Hours`, `Corrective Maintenance Hours`, `Total Support Hours`,  
`Organisation Type`, `Industry Sectors`, `Application Size`, `Application Size  
Approach`)
```

```

) ENGINE=InnoDB DEFAULT CHARSET=latin1 COLLATE=latin1_bin;

/*!40101 SET character_set_client = @saved_cs_client */;

-- Dumping data for table `Software Maintenance Data`

LOCK TABLES `Software Maintenance Data` WRITE;

/*!40000 ALTER TABLE `Software Maintenance Data` DISABLE KEYS */;

INSERT INTO `Software Maintenance Data` VALUES
(11808,'2012','1',823,758,0,'Governemnt','Government','384','IFPUG'),(12558,'2
012','16',0,0,0,'Government','Government','350
','IFPUG'),(12650,'2010','16',0,0,0,'Governemnt','Government','7000','IFPUG'),(1
3532,'2009','10',8,7,0,'Government','Governemnt','78','IFPUG'),(13560,'2010','3',
174,174,150,'Government','Government','150','IFPUG'),(13754,'2012','10',238,2
17,0,'Government','Governemnt','330','IFPUG'),(14013,'2012','10',888,758,0,'Go
vernment','Governemnt','650','IFPUG'),(14020,'2008','0',823,278,142,'Medical','
Governemnt','852','COSMIC'),(14042,'2012','4',266,108,0,'Government','Govern
ment','1350','IFPUG');

/*!40000 ALTER TABLE `Software Maintenance Data` ENABLE KEYS */;

UNLOCK TABLES;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;

/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;

/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;

/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;

/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;

```

```
/*!40101 SET  
COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
```

```
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;
```

```
-- Dump completed on 2015-07-27 21:21
```

Appendix D Questioner for Demonstration Feedback

Software Obsolescence Cost Analysis Framework

Feedback form

Name: _____

Company: _____

Designation: _____

Years in Post: _____

Please mark your selection using 1-5 marking scheme of which 1 being acceptable and 5 being not acceptable. Please also indicate what needs to be added or considered for each element of framework in the box under each question.

1. Do you agree with the principal of Software Obsolescence Cost Analysis Framework?

1 2 3 4 5

2. How confident are you in using Software Obsolescence Cost Analysis framework to identify the Software Obsolescence issues?

1

2

3

4

5

3. Would you use Software Obsolescence Cost Analysis framework to manage your Software Obsolescence issues?

1

2

3

4

5

4. Do you agree with Software Obsolescence Cost Analysis Framework's Obsolescence Management Level?

1

2

3

4

5

5. Would you be comfortable is applying the principals of Obsolescence Management Level to manage your Software Obsolescence?

1

2

3

4

5

6. Do you agree with Software Obsolescence Complexity Level?

1

2

3

4

5

7. Would you use the Software Obsolescence Complexity Level to identify and rank your Software Obsolescence issues?

1

2

3

4

5

8. Do you agree with Software Obsolescence Resolution Approach?

1

2

3

4

5

9. Would you consider these Software Obsolescence Resolution approach as mitigation action for your Software Obsolescence issues?

1

2

3

4

5

10. Would you be able to have better understanding of your Software Obsolescence issue if you were to apply SOCAT framework?

1 2 3 4 5

11. Would you be interested in testing the tool that will be developed as part of this framework to identify the cost of software Obsolescence?

1 2 3 4 5

12. Any other comments that will help me to improve the SOCAT Framework.

Appendix E Case Study Demonstration Questioner

General

- 1. Name
- 2. Organisation
- 3. Role
- 4. Years of Experience

Logic

5. Is the flow of information in the framework logical?

1	2	3	4	5	6	7	8	9	10
Totally invalid	Valid with changes/update			major	Valid with changes/update			Minor	Totally Valid

If there is change/update required, please describe them

6. Is the framework suitable for the bid stage?

1	2	3	4	5	6	7	8	9	10
Totally unsuitable	Suitable with major changes/update				Suitable with Minor changes/update				Totally suitable

If it is not suitable, please explain why?

7. Can it be used at all stages of the software development life cycle

Yes	No
-----	----

If yes, select the appropriate one from the below list

Requirements	Yes	No
Design	Yes	No
Development	Yes	No
Testing	Yes	No
Maintenance	Yes	No

Generalisability

8. Is the framework generalisable within the defence and aerospace sector

Yes	No
-----	----

9. Is the framework generalisable for other types of software

Engineering Software	Yes	No
Support Software	Yes	No
AIS (Automated Information System)	Yes	No

Benefits of using the framework

10. To what degree could the framework benefit a Software Project team/bidding team?

1	2	3	4	5	6	7	8	9	10
Not beneficial	Little use				Quite Beneficial				Highly Beneficial

Limitation of the Framework

11. What are the potential limitations and challenges in using and implementing the framework?

Lack of Software Obsolescence awareness	Yes	No
Lack of methods to monitor Software Obsolescence		
Lack of stakeholder engagement		
Lack of Software Obsolescence data		

12. Will this framework have potential implications across the software supply chain?

Yes	No
-----	----

Usability of the framework

13. What are the strongest features of the framework?

14. What are the weakest features of the framework?

15. How intuitive is the framework?

1	2	3	4	5	6	7	8	9	10
Counter intuitive	Low intuitive				Quite Intuitive				Very intuitive

16. Does the framework provide enough initial information?

Yes	No
-----	----

a. If not, please explain

--

17. Are the terminology and concepts used in this framework consistent?

Yes	No
-----	----

a. If not, please explain

--

18. Are all the key cost drivers considered in this cost estimating framework?

Yes	No
-----	----

a. If not, please explain

--

19. Is the list of resolution techniques indicated in the framework comprehensive?

Yes	No
-----	----

a. If not, please indicate the missing resolution technique

--

20. Is the framework flexible enough to adapt to the different levels of information available?

Yes	No
-----	----

a. If not, please explain

--

21. Is the Software Obsolescence Management level defined in the framework valid?

Yes	No
-----	----

a. If not, please explain

--

22. Is the Software Obsolescence Complexity level defined in the framework valid?

Yes	No
-----	----

a. If not, please explain

--

23. Is there any other comment that you want to add?

--

Appendix F The Coding Scheme for Cognitive Case study.

	Problem Domain
1	Micro Strategy
1.1	<i>Analysis the problem</i>
1.1.1	Analysing the problem
1.1.2	Consulting information about the problem
1.1.3	Evaluating the problem
1.1.4	Postponing information about the problem
1.2	<i>Proposing a solution</i>
1.2.1	Proposing a solution
1.2.2	Clarifying a solution
1.2.3	Retracting a previous solutions
1.2.4	Making a decision
1.2.5	Consulting external information
1.2.6	Evaluating external information
1.2.7	Postponing an actions
1.2.8	Looking ahead
1.2.9	Looking back

1.2.10	Making an assumptions
1.2.11	Applying cost
1.3	<i>Analysing the solution</i>
1.3.1	Analysing the proposed solution
1.3.2	Justifying a proposed solution
1.3.3	Calculating
1.3.4	Postponing analysis of proposed solutions
1.3.5	Evaluating the proposed solutions
1.4	<i>Explicit strategies</i>
1.4.1	Referring to application knowledge
1.4.2	Referring to a domain knowledge
1.4.3	Referring to an software obsolescence strategy
1.4.4	Explaining day to day task
2	Macro strategy
2.1	Writing notes
2.2	Top-down approach
2.3	Decomposing the problem
2.4	Back tracking

2.5	Opportunist
3	Level of abstraction
3.1	System
3.2	Interaction
3.3	Subsystem
3.4	Details
3.5	Requirement
4	Problem Area
4.1	<i>Low level designing</i>
4.1.1	Defining the Objects
4.1.2	Defining the Use Case
4.1.3	Defining the data base and data flow
4.2	<i>Standards</i>
4.2.1	Defining software development standards
4.2.2	Defining documentation and product manuals
4.3	<i>Development Life Cycle</i>
4.3.1	Waterfall
4.3.2	Agile

4.3.3	Test Driven Development
4.3.4	Continuous integration test driven development
4.3.5	Prototyping
4.3.6	Incremental
4.3.7	Iterative
4.3.8	Spiral
4.4	<i>System Requirement</i>
4.4.1	Defining the requirement
4.4.2	Key Drivers that effect development
4.4.3	Contingency plan
4.5	<i>Validation</i>
4.5.1	Defining test requirement
4.5.2	Defining the test case
4.6	<i>Maintenance (Support)</i>
4.6.1	Defining the support team
4.6.2	Defining the support structure
4.6.3	Defects rates and error code documentation

Appendix G The day rate escalation table

Resources	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030
Project Manager	88.24	90.45	92.71	95.02	97.40	99.84	102.33	104.89	107.51	110.20	112.95
Software designer	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Senior software Developer	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Junior Software Developer	69.86	71.61	73.40	75.23	77.11	79.04	81.02	83.04	85.12	87.25	89.43
Trainee Software Developer	53.3	54.63	56.00	57.40	58.83	60.30	61.81	63.36	64.94	66.56	68.23
Software Tester	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Project Manager	88.24	90.45	92.71	95.02	97.40	99.84	102.33	104.89	107.51	110.20	112.95
Software designer	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Senior software Developer	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Junior Software Developer	69.86	71.61	73.40	75.23	77.11	79.04	81.02	83.04	85.12	87.25	89.43
Trainee Software Developer	53.3	54.63	56.00	57.40	58.83	60.30	61.81	63.36	64.94	66.56	68.23
Software Tester	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Project Manager	88.24	90.45	92.71	95.02	97.40	99.84	102.33	104.89	107.51	110.20	112.95
Software designer	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Junior Software Developer	69.86	71.61	73.40	75.23	77.11	79.04	81.02	83.04	85.12	87.25	89.43
Software Tester	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Project Manager	88.24	90.45	92.71	95.02	97.40	99.84	102.33	104.89	107.51	110.20	112.95
Software designer	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Junior Software Developer	69.86	71.61	73.40	75.23	77.11	79.04	81.02	83.04	85.12	87.25	89.43

Software Tester	76.05	77.95	79.90	81.90	83.94	86.04	88.19	90.40	92.66	94.98	97.35
Project Manager	88.24	90.45	92.71	95.02	97.40	99.84	102.33	104.89	107.51	110.20	112.95
Systems Engineers	86.62	88.79	91.01	93.28	95.61	98.00	100.45	102.96	105.54	108.18	110.88
Software Engineers	79.58	81.57	83.61	85.70	87.84	90.04	92.29	94.60	96.96	99.38	101.87
Technicians	79.58	81.57	83.61	85.70	87.84	90.04	92.29	94.60	96.96	99.38	101.87
Project Manager	88.24	90.45	92.71	95.02	97.40	99.84	102.33	104.89	107.51	110.20	112.95
Systems Engineers	86.62	88.79	91.01	93.28	95.61	98.00	100.45	102.96	105.54	108.18	110.88
Software Engineers	79.58	81.57	83.61	85.70	87.84	90.04	92.29	94.60	96.96	99.38	101.87
Technicians	79.58	81.57	83.61	85.70	87.84	90.04	92.29	94.60	96.96	99.38	101.87

