An investigation of the human costs of software upgrades in organisations

This item was submitted to Loughborough University’s Institutional Repository by the/an author.

Additional Information:


Metadata Record: https://dspace.lboro.ac.uk/2134/27006

Publisher: © Javier Bajer

Rights: This work is made available according to the conditions of the Creative Commons Attribution-NonCommercial-NoDerivatives 2.5 Generic (CC BY-NC-ND 2.5) licence. Full details of this licence are available at: http://creativecommons.org/licenses/by-nc-nd/2.5/

Please cite the published version.
<table>
<thead>
<tr>
<th>Author/Filing Title</th>
<th>Base, J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accession/Copy No.</td>
<td></td>
</tr>
<tr>
<td>Vol. No.</td>
<td></td>
</tr>
<tr>
<td>Class Mark</td>
<td></td>
</tr>
</tbody>
</table>

8 Oct 1999

28 Jun 2003

0401660397
An Investigation of the Human Costs of Software Upgrades in Organisations

by

Javier Bajer

A Doctoral Thesis
Submitted in Partial Fulfilment of the Requirement for
the Award of Doctor of Philosophy of
Loughborough University

December 1997

Supervisor: Prof. Ken Eason
Director of Research: Dr. Mark Lansdale
Department of Human Sciences
Loughborough University
Loughborough, United Kingdom

© 1997 by Javier Bajer
To the millions of users
who will experience software upgrades
more often than they would like to

To my wife Sharon
for her loving support
ABSTRACT

The purpose of this research was to investigate the effects that continuous change in the form of software upgrades have on end-users. These effects could increase upgrade implementation costs in organisations.

Fifty-two managers working in three international consulting firms provided an initial perspective on how organisations deal with software upgrades, identifying the reasons behind them and typical implementation strategies.

This perspective is completed with another study in which sixty-two end-users reported on the different problems that occurred as a consequence of software upgrades. The major problem identified was the transfer of existing skills into new versions of the software; a paradoxical problem, as it was found to increase, as the differences between versions became minimal.

Based on these findings, this thesis proposes a model of skill transfer applied to the field of software upgrades, which analyses the interference effects caused by well practised previous skills, in the acquisition of new ones. The model identifies visual cues as a mechanism to minimise the interference effects.

This model was then tested through a field-based experiment in which forty-eight subjects were presented with a purpose-built upgrade simulation. Results from their performance indicated that visual cues had a positive effect in the minimisation of interference. However, perception of the change was found to have a much bigger impact on the successful performance in the new situation.

Finally, this thesis presents a preliminary strategy with the intend to aid organisations in a better management of the human end of software upgrades.
ACKNOWLEDGEMENTS

I am very thankful to many people. Without their support this work would have not been at all possible.

I wish to express my gratitude and appreciation to my supervisors Prof. Ken Eason and Dr. Mark Lansdale, as they provided direction in the areas of Organisational and Cognitive Psychology respectively. They both have been invaluable resources in my attempt to bridge the two fields of research. I know that dealing with a "mature student" (this is not a statement of self-admiration but a name given to older people that return to academic life) requires an extra amount of patience and support. They both exceeded their job, supporting me all along these years. Thank you!

I owe many debts to the people from Andersen Consulting, Perot Systems and Oracle Corporation, for participating in my experiments, and being key resources in understanding the Organisational drivers for software upgrades. Particularly I want to thank Fenny Ang, Dan Heany, Jacquie Heany, Paul Collins, Kim Rash, Phil Wise, Mitch Kotula, Sue Mosher, Wade and Katherine Ashley, Kate Lidbetter, Remedios Orrantia, Jose Domecq, Ana Rodicio, Tom Pate, Ed Evanick, Gloria Chem, Floyd Kemske, Judith Bealey, Alison Brooks, Guy Warren, Dawn Tostenson, Andrew Morris, Knut Karlsen, Vincent Wong, Lynn Kotwicki, Ashley Crandall, Paul Brightbill, Ahli Moore, Jiri Ron, James Birmingham, Kurt Romsons, Heidi Smith, Gregory Dominish, Ree Greteman, Dennis Bowers, Richard Chionh, Rachel Engel, Michael Martin, Katherine Rivers, Caroline Laurin, Wai-ming Yu, Rohan Ashton, Robin Caffrey, Vicki Sanders, Melissa Keasey, Thomas Gebhart, Kim Tam Lam, Lai Yee Ng, Tarquin Ralph, Brad Smith and Sara Ward for their input and open discussion on the subject.
My appreciation also goes to Mauro Garcia for his feedback and revisions to early drafts of this thesis, and to my colleagues at the Cognitive Ergonomics Research Group for their continuous encouragement.

Last, but most important, I specially want to thank my wife, Sharon, who not only helped with her comments on the thesis but has been a great friend and supporter along these years. I promise I'll wait another ten years for my next PhD.
TABLE OF CONTENTS

Abstract ............................................................................................................... i
Acknowledgements ......................................................................................... ii
Table of Contents ........................................................................................... iv
List of Tables .................................................................................................. xii
List of Figures .................................................................................................. xiv

1. Introduction .................................................................................................... 1
   1.1 Homo Habilis. The Handy Man? .............................................................. 1
   1.2 Total Cost of Ownership model (TCO) .................................................. 7
       1.2.1 TCO values for a fictitious organisation .................................... 9
   1.3 The reason for this thesis ........................................................................ 9

2. Methodology of this thesis ........................................................................... 12
   2.1 Chapter Summary .................................................................................... 12
   2.2 Introduction ............................................................................................. 12
   2.3 Scope ....................................................................................................... 13
       2.3.1 User population ............................................................................. 13
   2.4 The ecological validity problem ............................................................. 14
       2.4.1 An Ethnographic Perspective ....................................................... 16
   2.5 Choosing An Appropriate Research Strategy ........................................ 18
       2.5.1 Scientific Methods ....................................................................... 19
           2.5.1.1 Laboratory experiments ..................................................... 19
           2.5.1.2 Field experiments ............................................................... 19
           2.5.1.3 Surveys .............................................................................. 20
           2.5.1.4 Case studies ...................................................................... 21
2.5.1.5 Forecasting ................................................................. 21
2.5.1.6 Simulation ................................................................. 22
2.5.2 Interpretivist Methods ...................................................... 23
  2.5.2.1 Subjective / argumentative ........................................ 23
  2.5.2.2 Descriptive / interpretative research / Future research ... 23
  2.5.2.3 Action research ....................................................... 24
2.6 A Research Strategy for this thesis ....................................... 25
2.7 Chapter Conclusions .......................................................... 30

3. An Organisational View of Software Upgrades .................... 31
  3.1 Chapter Summary .......................................................... 31
  3.2 Introduction .................................................................... 31
  3.3 Method ........................................................................... 32
    3.3.1 Sample .................................................................... 33
    3.3.2 Questionnaire .......................................................... 33
    3.3.3 Distribution .............................................................. 34
    3.3.4 Return ratio .............................................................. 35
  3.4 Results ........................................................................... 35
  3.5 Question 1: “What drives organisations to upgrade their software?” .. 36
    3.5.1 Findings ................................................................. 36
    3.5.2 Direct Drivers ........................................................... 37
      3.5.2.1 Required functionality .......................................... 38
      3.5.2.2 Cost of maintaining existing system ...................... 38
      3.5.2.3 Competitive advantage ...................................... 38
      3.5.2.4 Organisational changes ..................................... 39
      3.5.2.5 Support of new job requirements ......................... 39
      3.5.2.6 Improved productivity ........................................ 40
3.5.2.7 The new version allows the handling of objects in a better way ................................................................. 41
3.5.2.8 Bug fixing ................................................................................................................................. 41

3.5.3 Indirect Drivers ........................................................................................................................ 41
3.5.3.1 The current version will not be supported by the hardware or software provider in the future .......... 42
3.5.3.2 "We don't want to fall behind" syndrome ........................................................................... 42
3.5.3.3 Feeling "behind" change ................................................................................................. 44
3.5.3.4 Organisations that want to utilise new technology .............................................................. 45
3.5.3.5 Hardware-Software dependency ......................................................................................... 45
3.5.3.6 Part of a "package" (Office, Smart Suite, etc.) .................................................................... 46
3.5.3.7 It comes with the new computer ....................................................................................... 47
3.5.3.8 Change in job requirements ............................................................................................... 47
3.5.3.9 Compatibility with other users / department / organisation / clients / suppliers .......................................................................................................................... 47
3.5.3.10 Compatibility with operating systems and tools ............................................................... 48
3.5.3.11 Upgrading to fix previous upgrade .................................................................................. 48
3.5.3.12 The upgrade is free ........................................................................................................ 49

3.5.4 Discussion ............................................................................................................................ 50

3.6 Question 2: "How do organisations manage the implementation of upgrades?" ................................................................. 51
3.6.1 Findings ................................................................................................................................. 51
3.6.2 Discussion ............................................................................................................................ 52

3.7 Question 3: "How much awareness organisations have of the hidden costs of upgrades?" ................................................................................................. 53
3.7.1 Findings and discussion ......................................................................................................... 53

3.8 The paradox of the minimisation of hidden costs .................................................................................. 54
3.8.1 The Zero Administration Initiative for Windows ..................................................................... 55
3.8.2 The paradox

3.9 Chapter Conclusions

4. A Survey of Problems Caused by Software Upgrades

4.1 Chapter Summary
4.2 Introduction
  4.2.1 Method
  4.2.2 Sample
    4.2.2.1 Number
    4.2.2.2 Variety in skills and experience
    4.2.2.3 Variety in usage
    4.2.2.4 Different platforms and technical environments
    4.2.2.5 Individual work or group work
    4.2.2.6 Different levels of support
    4.2.2.7 Age
  4.2.3 Questionnaire Design Rationale
    4.2.3.1 Question types
    4.2.3.2 Organisation of the Questions
  4.2.4 Pilot Test
  4.2.5 Distribution
  4.2.6 Responses
    4.2.6.1 Demographics
  4.2.7 Interviews
4.3 Analysis of Results - Questionnaire and Survey
  4.3.1 Overall findings
  4.3.2 Particular problems found
    4.3.3 Skill transfer problems
      4.3.3.1 Using old commands
4.3.3.2 Searching for old commands ............................................ 81
4.3.3.3 Using old procedures ........................................................ 82
4.3.4 Upwards / Downwards Compatibility problems ....................... 82
4.3.5 Response time problems ........................................................ 83
4.3.6 Loss of configuration problems .................................................. 85
4.4 Correlation Analysis ................................................................................. 86
4.4.1 Time to master .............................................................................. 87
4.4.2 Type of upgrade ............................................................................ 87
4.4.3 Experience .................................................................................... 88
4.4.4 User Satisfaction ........................................................................... 88
4.4.5 Lack of support ............................................................................. 88
4.5 Discussion ............................................................................................ 89
4.5.1 Transfer of Skills ........................................................................... 90
4.5.1.1 Transfer of Skills and the visibility of the upgrade ................. 91
4.5.1.2 Transfer of Skills and Stress levels ........................................ 92
4.5.2 Preliminary Hypothesis ................................................................. 93
4.6 Chapter Conclusions .............................................................................. 94

5. Towards a Model of Transfer of Skills for Upgrades ......................... 95
5.1 Chapter Summary .............................................................................. 95
5.2 Introduction ...................................................................................... 96
5.3 Transfer: A Definition ......................................................................... 96
5.4 A Classification Of Transfer .................................................................. 97
5.4.1 Direction ....................................................................................... 97
5.4.1.1 Positive transfer ..................................................................... 98
5.4.1.2 Negative transfer ................................................................... 98
5.4.1.3 Non-existent transfer .............................................................. 98
5.4.2 Dimension ..................................................................................... 98
6.2.4 Tool and Tasks.........................................................................................128
  6.2.4.1 First task..........................................................................................130
  6.2.4.2 Second task......................................................................................131
  6.2.4.3 Third task........................................................................................131
  6.2.4.4 Visual cues......................................................................................134
6.2.5 Pilot study.............................................................................................134
6.2.6 Procedure..............................................................................................135
6.2.7 Questionnaire.......................................................................................135
6.2.8 Interviews............................................................................................137
6.3 Results and Analysis...............................................................................137
  6.3.1 Interpretation.......................................................................................140
  6.3.2 The role of perceived change..............................................................141
6.4 Chapter Conclusions................................................................................144

7. An Organisational Strategy to Minimise the Human Costs of
   Upgrades..................................................................................................145

  7.1 Chapter Summary....................................................................................145
  7.2 Introduction.............................................................................................145
  7.3 An initial big picture of the model........................................................147
  7.4 Planning an Upgrade...............................................................................150
    7.4.1 Inform Decisions..............................................................................150
    7.4.2 The decision to upgrade...................................................................151
      7.4.2.1 Different upgrade strategies..................................................151
  7.5 Designing an upgrade............................................................................154
  7.6 Implementing an upgrade.......................................................................156
    7.6.1 Communication...............................................................................156
    7.6.2 Training............................................................................................156
      7.6.2.1 Managing skills.........................................................................157
10.1 Appendix I - Organisational Survey .............................................................. 185
10.2 Appendix II - End-User Survey .................................................................... 186
10.3 Appendix III - Transfer Theories ............................................................... 187
10.4 Appendix IV - Word 200 Experiment ....................................................... 188
10.5 Appendix V - Job Aids .................................................................................. 189
LIST OF FIGURES

Figure 1-1, Pectolite hammer with handle of Cariou antler (left) and today's hammer (right)................................................................. 2
Figure 1-2, Gartner Group's distribution of costs .................................................. 8
Figure 2-1, Lewin's Action Research spiral of steps ........................................... 24
Figure 2-2, Research strategy adopted for this thesis ....................................... 27
Figure 3-3, Advertisement for Lotus Corporation, targeting Windows 3.1 users............................................................... 43
Figure 4-1, An example of open question ......................................................... 65
Figure 4-2, An example of closed question ....................................................... 65
Figure 4-3, An example of the open-closed combination used ......................... 66
Figure 4-4, Distribution of experience ................................................................ 74
Figure 4-5, Time to master new version ............................................................ 76
Figure 4-6, Upgrade requested by ...................................................................... 77
Figure 4-7, Types of support used for upgrades Note: The two training strategies (Training course and On-site training) are considered to be formal support actions as they are managed by the organisation................................................................. 78
Figure 4-8, Distribution of type of problems found in the survey....................... 80
Figure 4-9, Launching times of Microsoft Word for different versions of the software (from MacUser, 1995)................................................ 84
Figure 4-10, Word count times of Microsoft Word for different versions of the software (from MacUser, 1995)........................................ 84
Figure 5-1, Two different directions of interference ............................................ 106
Figure 5-2, The Skaggs-Robinson Hypothesis of Transfer ................................. 109
Figure 5-3: The retroactive interference surface as described by Osgood. 110
Figure 5-4, Using GOMS with Osgood's similarity paradox model............... 112
Figure 5-5, The addition of context to Osgood's model of transfer .............. 113
Figure 6-1, An application of the ERMIA model .................................... 122
Figure 6-2, Times per task grouped by group .......................................... 138
Figure 6-3, Times per task grouped by perception of similarity ................ 143
Figure 7-1, A method for the minimisation of human costs of software
  upgrades .......................................................................................... 149
Figure 7-2, Upgrade Strategies ................................................................ 152
LIST OF TABLES

Table 1-1: Microsoft's releases of different versions of Word® ........................................ 4
Table 2-1, Galliers distinction of information systems research approaches........................................................ 18
Table 3-1, Number of Direct and Indirect drivers. These figures are based in the total number of drivers indentified by all respondents (n=52)........................................................................................................ 37
Table 3-2, User Cost-Benefit Assessment (adapted from Eason, 1988)........ 53
Table 4-1, Codes used to identify the severity of the change ........................................ 86
Table 4-2, A comparison of findings between total and restricted sample..... 89
Table 6-1, Different methods for opening an existing document using Word for Windows............................................................................. 122
Table 6-2: Questions used in the questionnaire ......................................................... 136
Table 6-3: Average times taken to perform tasks one to three and standard deviations .............................................................................. 139
Table 7-1, Similarity between versions ....................................................................... 154
1. Introduction

"640 k should be enough for everybody"
Bill Gates, 1982

1.1 Homo Habilis. The Handy Man?

More than 1.5 million years ago the lands of Africa were inhabited by the first known species of our genus, Homo. It is attributed to the Homo Habilis (Latin for handy man) the creation of the first tools. During that period the Homo Habilis built basic shelters, gathered plant foods and hunted game with the aid of basic bone tools he (and she) produced. Since then, and during all stages of evolution, humans survived because of the use of their tools.

Humans sculpt their environment at the same time they adapt to it. It took several million years for mankind to develop to where it is today. Even if today we can send a rocket to the moon or create virtual environments, our cognitive capabilities "remain those of a self-propelled terrestrial creature designed to move at around three to four miles per hour under conditions of normal gravity and above ground" (Reason, 87). Our minds and bodies are still very much like those of the Homo Habilis.

Our perception of change is far from precise. We work better in detecting changes rather than in identifying exact values, such as temperature.
This slow adaptation of humans to their environment introduces the main question this thesis is intended to answer: How do people cope with frequent changes, specifically those related to the upgrade of their information technology tools?

![Figure 1-1, Pectolite hammer with handle of Cariou antler (left) and today's hammer (right)](image)

Most tools that aid our existence require specific skills from our part. However, once those skills are developed, they remain intact, or even become "second nature", permitting us perform other tasks at the same time, reducing probabilities of errors and relaxing the levels of attention needed to use that tool. Most tools remain unchanged, together with the skills required to used them. Or if they change, they do so over a long period of time.

For example, the design of some manual tools, such as a hammer, has remained intact for more than a million years (see Figure 1-1). Although this represents a somewhat extreme example, it helps to illustrate the fact that in order to survive, humans did not need to rapidly change the tools they use. Neither the skills to operate them.

Unlike manual tools, software tools are much more transient. Software products quickly "evolve" through time, taking the shape of upgrades.
An example of how software is upgraded can be seen by looking into the popular wordprocessor Microsoft Word®. Table 1-1 illustrates all versions of this product since it was released eight years ago.

<table>
<thead>
<tr>
<th>Versions for Windows®</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft® Word v98 for Windows®</td>
</tr>
<tr>
<td>Microsoft® Word v97 for Windows®</td>
</tr>
<tr>
<td>Microsoft® Word v7.0 for Windows® (also Word 95)</td>
</tr>
<tr>
<td>Microsoft® Word v6.0 for Windows®</td>
</tr>
<tr>
<td>Microsoft® Word v2.0 for Windows® &amp; Bookshelf® Video &amp; Sound</td>
</tr>
<tr>
<td>Microsoft® Word v2.0 for Windows® &amp; Bookshelf® Multimedia Edition</td>
</tr>
<tr>
<td>Microsoft® Word v2.0c for Windows®</td>
</tr>
<tr>
<td>Microsoft® Word v2.0 for Windows® Video &amp; Sound Edition</td>
</tr>
<tr>
<td>Microsoft® Word v2.0 for Windows® CDK</td>
</tr>
<tr>
<td>Microsoft® Word v2.0 for Windows® Courseware</td>
</tr>
<tr>
<td>Microsoft® Word v1.1 for Windows®</td>
</tr>
<tr>
<td>Microsoft® Word v1.0 for Windows®</td>
</tr>
<tr>
<td>Microsoft® Word v1.0 for Windows® Class-in-a-box</td>
</tr>
<tr>
<td>Microsoft® Academic Value Pack</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Versions for MS-DOS®</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft® Word v5.5 for MS-DOS®</td>
</tr>
<tr>
<td>Microsoft® Word v5.0 for MS-DOS®</td>
</tr>
<tr>
<td>Microsoft® Word v5.0 for MS-DOS® CIAB</td>
</tr>
<tr>
<td>Microsoft® Word v4.0 for MS-DOS®</td>
</tr>
<tr>
<td>Microsoft® Word v3.0 for MS-DOS®</td>
</tr>
<tr>
<td>Microsoft® Word v1.0 for MS-DOS®</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Versions for Macintosh®</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft® Word v6.0 for the Macintosh®</td>
</tr>
<tr>
<td>Microsoft® Word v5.1a for the Macintosh®</td>
</tr>
<tr>
<td>Microsoft® Word v5.0 for the Macintosh®</td>
</tr>
<tr>
<td>Microsoft® Word v4.0 for the Macintosh®</td>
</tr>
<tr>
<td>Microsoft® Word v4.0 for the Macintosh® Class-in-a-Box</td>
</tr>
<tr>
<td>Microsoft® Word v4.0 for the Macintosh® Courseware Developer's Kit</td>
</tr>
<tr>
<td>Microsoft® Word v3.02 for the Macintosh®</td>
</tr>
<tr>
<td>Microsoft® Word v1.0 for the Macintosh®</td>
</tr>
<tr>
<td>Microsoft® Academic Value Pack v1.0 for the Macintosh®</td>
</tr>
</tbody>
</table>
Table 1-1: Microsoft's releases of different versions of Word®

A new version (or release) of this product is launched into the market every two years or less, increasing the frequency almost logarithmically.

The following story summarises how frequent upgrades are received from the perspective of an end-user, describing the problems they might cause. This is a real story, based on information gathered through interviews with end users (these interviews are part of a study described in detail in Chapter 4). A user recalls:

"I remember 12 years ago when, to get a letter typed, you had to be on a first-name basis with the supervisor of the typing pool to get a letter sorted in two days.

The bank invested in Wang, and there was No One in the bank to rely on to figure it out, but there was a certain cachet and definitely a career value in putting in the time to learn it—"XX is one of the progressives—she can tell you what to do on that machine". With the manuals and codes, doing a letter or paper with revisions could take days and days.

Then we advanced to Mac and I remember spending two weeks at WordPerfect® training, then it took me about a year to be good at basic papers and then the department newsletter. I knew the WordPerfect® help desk support people—often in another city—very well."
Finally we got the PC's, and everything was so much easier, and fun! But we were still dependent on the technical experts to convert files (not everyone was so lucky) or the typing pool (the executives still used this diminished function) to re-type documents which our PC's couldn't read.

They installed Word® for Windows on the back of the conversion problem, even though the conversion software was finally advancing. At first Windows seemed to be the same, but it wasn't, and I, for one, was angry that we seemed to be taking a step back. The IT department (as it had become known) assured us that subsequent upgrades would solve the problems and eventually we'd be as good as before, with the benefit of all our computers being able to talk to each other.

The period after that seemed like whirlwind—E-mail, new "office Suites", lots of new "functionality"—it seemed that I was spending more time puzzling over my computer than doing actual work. I had more control over things like printing, but I was also forced to have control over things like virus checking and "customising my desktop". Also, being good at the computer wasn't a differentiation in my department anymore—everyone was basically in step. And complaining.

Now, every four months or so the men from the "PC support unit" come around to do this or that compulsory change—an upgrade to e-mail, an upgrade to the office suite, a change to the server. I KNOW that as soon as they leave my desk something will go wrong, that I won't be able to find things. For example, where is my file manager? Now I have to be on a first-name basis with the PC support unit".
There is a vast amount of literature that reflects the struggle that workforce goes through as it continuously adapts to new information technology tools. Toffler (Toffler, 1985) coins the term *novelty ratio* to represent the speed of technological change that organisations undergo in short periods of time. He cites the case of Bell System in North America, where the level of “newness” that people from top management to the lineman have to deal with has climbed dramatically, forcing them to “devalue and discard painstakingly acquired older skills” (pp.71-78).

In relation to upgrades, Gartner Group, a well reputed organisation that forecasts issues around the IT world writes: “OIS (Office Information Systems) migration is a journey and not a destination. Enterprises must prepare themselves for a continuous migration effort, driven by increasing waves of technological change expected to continue well beyond 2000. Successful approaches will focus on leadership in establishing the OIS infrastructure, engaging executive management in support of an ongoing commitment of resources, and intelligent selection of an appropriate and high-value starting point for the journey. Ineffectiveness in any of these areas is a prescription for OIS migration failure.” (Gartner Group, 1996a).

Upgrades can impact not only individuals but organisations as a whole. Organisational styles have to be continuously re-defined as changes in technology affect decision-making strategies. This mostly happens as knowledge gets re-distributed into different shapes within the organisation.
Some organisations are beginning today to develop an awareness of the impact that new information technology may have on them and on what the total costs of their information technology investments are.

As the period of enchantment with information technology as the only key to success in organisations begins to subside, new explanations on how much in reality IT costs to an organisation appear. An example of this is the Total Cost of Ownership model, described below.

1.2 Total Cost of Ownership model (TCO)

As a result of a joint effort between Gartner Group, Microsoft and Interpose Inc., the model’s intention is to account for direct and indirect costs associated with owning and using Information Technology in organisations. TCO is a business model developed to assist companies in understanding and managing the costs of Information Technology assets throughout their lifecycle.

The TCO model identifies as direct costs those which most organisations define as the major costs of upgrades: Hardware, software, labour, operations, administration, development, etc.

The indirect costs are those borne by the end users. Some of these are self and peer support, casual learning and productivity losses due to downtime. The TCO model even accounts for “futz” times, when users use the company’s computers to run their own software (i.e. games).
Microsoft already recognises that in an average organisation indirect costs can account for more than 50% of the Total Cost of Ownership. They also identify that these costs have tripled over the last ten years and are not normally accounted for by the client organisations (Microsoft, 1997).

Gartner Group also identifies a similar figure, estimating that fifty-six percent of the total system's lifecycle costs (over a three year period) falls under end-user operations (see Figure 1-2 below).

From the 56% of the costs attributed to end-user operations, they argue that 40% is end-user time spent on non-job related PC activities (Gartner Group, 1996b).
1.2.1 TCO values for a fictitious organisation

To illustrate the use of this model with an example, we will consider a fictitious network with 2,500 PC’s, an average of 8 applications per PC and an homogeneous operating system environment. The calculated costs per PC would be as follows:

- Direct costs (capital, support and administration) .......... $ 6,400 / year
- Indirect costs (casual learning, peer support) ............... $ 5,500 / year
- Total Cost of Ownership (per PC) ................................ $ 11,900 / year

According to the model, the indirect costs of IT for that particular organisation would be $ 5,500 per user, per year. An amount almost as high as the total software and hardware investment.

1.3 The reason for this thesis

During this introduction several topics were discussed. In order to present the reason for this thesis, the discussion in this chapter is summarised as follows:

1. The frequency in which software upgrades are released is increasing. This creates a need for a more frequent upgrade of skills at the user end.

---

1 The calculations for this example were performed using TCO Advisor, a software package developed by Interpose Inc. that calculates the Total Cost of Ownership of typical IT installations.

2 These figures in pounds: £ 4.125 and £ 3.437 approximately (total: £ 7.560)
2. The ability to continuously re-adapt to changing tools is not a strong human skill. The frequency of software upgrades could be causing problems to end users.

3. Research shows the existence of large indirect costs in the Total Cost of Ownership of IT installations. There could be a relation between these costs and software upgrades' impact on people.

It is the main concern of this thesis to develop an in-depth understanding of the many characteristics of human adaptation to software upgrades. Understanding the human side of software upgrades could assist in the creation of strategies for the minimisation of the existing costs.

In order to allow for a holistic understanding of this type of change, it is important to try and answer three main questions:

The first question is the one that refers to the characteristics of the problem itself. How deep is the problem, how is it affecting people's performance, are there any patterns by which this problem increases or decreases?

The second question is about the drivers or motivators that make organisations upgrade their software. Why do organisations upgrade their software? How aware are organisations of the possible hidden costs?

The last important question is about the implementation of the upgrade. How has upgrade implementation been supported? How much and what type of training was conducted?
The next chapter, Chapter 2, will describe the methodological alternatives for studying this problem, the different trade-offs and the method chosen to investigate the issue of software upgrades in this thesis.
2. Methodology of this thesis

2.1 Chapter Summary

The methodological aspects of this research are discussed in this chapter. We will first consider the problem of ecological validity of the research in this area, followed by an analysis of the different research methods available and finally the approach chosen for this particular thesis.

2.2 Introduction

A study of the human-related problems of software upgrades in organisations has to contemplate both the individual perspective of the users whose tools are upgraded and the organisational management of those upgrades. This research makes both cognitive science and organisational science work together for the benefits of a complete understanding of the issues involved and for a later analysis of possible solutions that can be implemented. One vision without the other would result in a partial view of the problem.

Most studies of transfer of learning of information technology tools concentrate on one of two areas:

1. The measurement of the amount of knowledge (or skills) that gets transferred between training and the practical application of the skill (Catrambone, 1988; Barnard, P.J. et al, 1982; Mack et al, 1983; Mayes et al, 1988))
2. The transfer of skills between different software products or between different technical platforms - i.e. Mac to PC (Poison and Kieras, 1985; Singley and Anderson, 1985; Singley and Anderson, 1987-1988).

No research was found where the effects of transfer are studied using versions of the same software package, a reality in today's organisational world.

2.3 Scope

The present thesis has two objectives:

- To evaluate the impact that software upgrades have on individuals working in organisations
- To identify mechanisms that could minimise the costs related to those upgrades

A software upgrade, in the context of this thesis, is defined as the manual or automatic installation of changes to an existing software tool, in the form of versions, releases or "fixes". The installation of a new system and changes to different software products are excluded from the scope of this thesis.

2.3.1 User population

The users targeted in this research are workers that perform their jobs in an organisation, excluding home users of technology tools and IT workers.
2.4 The ecological validity problem

The first methodological problem this thesis encounters is to give an answer to the following questions: Should research be conducted in an isolated form, where variables can be operated on and clearer results obtained? Or should research be more practical, including all complexities and concurrence of events, as they occur in real life? This methodological problem has been the concern of many, particularly those researching the human and social sciences (Neisser, 1978; Baddeley, 1993).

The concept of ecological validity proposes that research should be performed in, or very close to, the real world, involving all environmental variables that would be in place when the research is over. Extracting part of the elements and taking them into a laboratory setting will not guarantee a direct application of the findings back to the world.

This perception of the "direct application" or "usefulness" of research was supported by different people in different fields of study. For example Neisser (1976, 1978), in the human memory arena, pleads with the scientific community for a turn of the research in psychology towards an applied world. In the ergonomics field, Sanders and McCormick (1992) strongly support ecological validity in the design of physical environments, where "opinions and attitudes about an environment formed from a one- or two-minute exposure may be quite different from the opinions and attitudes about that environment developed after living or working in it for several weeks or months" (p. 486). Another advocate for this approach is Meister (1989), describing the ecological validity problem as "operational fidelity" when he talks about measuring problems in human factors (pp. 160).
In closer relation to this thesis, studies about transfer are usually performed in controlled environments where subjects have a clear definition of goals and tasks (see Polson 1988). Subjects' performance is generally tested against an expected output. These studies, although valuable for scientific research, do not necessarily reflect real field situations.

In the case of the more theoretical approaches, authors use diverse cognitive models for skill acquisition and transfer in relation to software. For their particular purpose a clean-cut small piece of interaction is needed and therefore it is well justified to create experiments in a controlled environment.

There is a need to observe the problem of transfer of skills in the information technology environment in real situations at an individual level. Charles Beard (1993) who conducted an extensive study on the subject of skill transfer noticed: “based upon a review of literature regarding the transfer of computer skills, it seems that there is a need for naturalistic studies of the transfer among actual software”. He found that current research was having too much focus on problems found through experimentation techniques instead of on the problems users have when using current software.

A naturalistic study of this kind would help understand how much of the theoretical findings apply to the real world, where events don't happen in isolation and user's performance is influenced by a wide range of variables, most of which happen in an uncontrolled manner.
However, laboratory controlled experiments can certainly tell an important part of the story that could be missed by the complexity of the findings from a field-based research method. Having more control on the variable(s) in an experiment would support a clearer link between action and the responses for that action, therefore building a more solid science.

The need for empirical naturalistic data is clear but the problem is to find an appropriate method of research that would enable the collection of data in an efficient and effective way.

2.4.1 An Ethnographic Perspective

Anthropologists utilise a rather different research perspective to the ones presented in the previous section. They propose a much wider approach than just observing people (as subjects), tasks and actions people perform in order to achieve their goals.

Anthropologists believe that appropriate research should consider the social context in which work takes place. Their research strategy is based on having the researcher directly involved with the context and people being studied, in order to understand the real interactions. The researcher should therefore be able to identify all factors that in one way or another affect performance.
In contrast to the more formal research methods, anthropologists argue that people are goal oriented and in most cases they do not have clear plans for each step they will have to perform. Instead, people respond to each unanticipated situation as it appears. Lucy Suchman (1987) uses the term plans, to refer to the more organised, abstract, analytical way in which people are able to describe their actions. However, she argues that people perform using what she calls situated actions, recruiting different resources that apply to the context in which a new problem is found. Existing plans would only be a weak input in addition to all other resources people use to resolve a problem. For a more detailed coverage of this debate see Suchman (1987) and Monk (1995).

The debate on whether human sciences researchers should be observers or active participants is still open. In the case of this thesis, a purely ethnographic approach would have proved to be inefficient due to the characteristics of the topic being researched: problems with software upgrades, as will be discussed further along this thesis, do not happen in a predicted manner, therefore, intending to participate from the problem could have been a lengthy process.
2.5 Choosing An Appropriate Research Strategy

At the outset of this research, different strategies were analysed in order to choose an appropriate one for this study. This analysis was based on Galliers' identification of research strategies for Information Systems research. Galliers identified two different groups of research approaches: A scientific, empirical one characterised by objective and rigorous observation performed in a repeatable, reductionist and refutable way; and an interpretivist one which argues that the scientific values are misplaced in social sciences research.

The main reasons for this assertion are the fact that social phenomena can have different interpretations and that scientists impact the social systems as those are being studied.

Galliers concluded that different research approaches can be appropriate under different circumstances and a decision should be made in each particular situation (Table 2-1).

Table 2-1, Galliers distinction of information systems research approaches

<table>
<thead>
<tr>
<th>Scientific</th>
<th>Interpretivist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laboratory experiments</td>
<td>Subjective/argumentative</td>
</tr>
<tr>
<td>Field experiments</td>
<td>Reviews</td>
</tr>
<tr>
<td>Surveys</td>
<td>Descriptive / interpretative</td>
</tr>
<tr>
<td>Case Studies</td>
<td>Future research</td>
</tr>
<tr>
<td>Forecasting</td>
<td>Action Research</td>
</tr>
<tr>
<td>Simulation</td>
<td></td>
</tr>
</tbody>
</table>
These methods will be now analysed in detail, evaluating their usefulness and appropriateness for this thesis.

2.5.1 Scientific Methods

2.5.1.1 Laboratory experiments

These represent the most common research approach in the area of skill transfer. In most of the cases, subjects are divided into groups and are given different tasks to perform. Their levels of performance are then compared.

Laboratory-type of experiments tend to have a good control of the variables involved so cause-effect relationships can be clearly observed. These type of experiments will be studied in chapter 5 as they are the underlying principles of skill transfer.

2.5.1.2 Field experiments

An ideal strategy for collecting naturalistic data would be to identify the subjects who are about to upgrade their information technology tools and to perform in vivo observations (or experiments) while subjects experience skill transfer problems. This approach could present a series of obstacles:

- Normally problems occur in a scattered way during the working day,
- Problems do not happen under the label of "problem". Users tend to automatically blame themselves for the poor levels of performance with the new tool, a reaction that would mask the problems.
• The experimenter should have a deep understanding of the user goals and task being performed. This is not always possible, especially as users may not always be aware of their low-level goals.

• Observing over the shoulder is a problematic technique, as it disrupts the subjects on one hand and can cause an increase on the user’s performance on the other (i.e. Hawthorn Effect in Roethlisberger & Dickson, 1939 and Wilson & Corlett, 1992).

• There is no guarantee that problems will occur. As will be discussed later not all the users experience problems of transfer.

2.5.1.3 Surveys

Surveys can be an effective way for the identification of problems as they can reach large number of subjects. Similar to the fishermen when they approach a new area and don’t know about the types of fish they might catch, surveys are good at “casting a wide net”. A broad spectrum of problems can be collected and the researcher can decide the areas in which there is a need to perform further explorations with the use of a different technique (e.g. interviews, focus groups, etc.).

Other advantages of surveying techniques are that they cause little disruption as users choose the most appropriate moment to complete the survey. They allow for anonymity and they can be very cost-effective.

On the disadvantage side it is important to mention that surveys tend to be subjective as they only represent the users’ opinion and his or her interpretation of the problems.
The survey method was therefore chosen for this particular stage of the thesis. A detailed account of the design rationale can be found towards the end of this chapter.

2.5.1.4 Case studies

A case study is a "war-story" type of research. It describes the experience that an organisation goes through as it implements a new type of technological solution or different process.

This approach is very useful as it supports conclusions with real data and tends to account for a wide range of variables, providing the cause-effect relation over time. On the other hand, this type of research strategy provides a limited number of unique events, making generalisations quite difficult.

A case study would not be appropriate for this thesis as there is no solution to analyse, but a need to identify possible existing problems. Another reason for not choosing this research method is that case studies tend to consider the big picture of the workforce, instead of a detailed account of individual problems.

2.5.1.5 Forecasting

This method consists in the extrapolation of future trends based on data from the past (i.e. using mathematical formulae such as regression analysis). This approach would have been appropriate had the main objective of this study been the estimation of version changes in the future, instead of the impact of such changes.
2.5.1.6 Simulation

This strategy is used to test solutions that are either not safe to test in real life (e.g. aeroplane crash procedures) or do not exist at that moment (e.g. testing the usage of mobile phones before they were created).

Examples of simulation techniques are front-end prototypes, Velcro dashboards and the Wizard of Oz technique. This last one is performed by having a person role-play the future behaviour of the tool as the user interacts with it. The Wizard of Oz technique is a cheap way of anticipating issues before the technology (or solution) is available for testing (see Nielsen, 1993, pp. 96).

A simulation was not an appropriate technique in the case of this thesis as the main objective was to observe transfer problems with current technology as they occur naturally. Simulating a software upgrade through dashboards or low-fidelity prototypes would not support an ecological validity approach.
2.5.2 Interpretivist Methods

2.5.2.1 Subjective / argumentative

This approach is based on the researcher's opinion and speculation rather than on observation on a particular subject. This is normally an individual, rather than group, activity. Whether this method should be considered valid research is questionable, although it provides inspiration and creativity that could later be tested through more formal approaches. More than an opinion, the research conducted for this thesis needed real data from real users, as described under the Ecological Validity heading. A subjective method would have not supported it.

2.5.2.2 Descriptive / interpretative research / Future research

This is a scholarship type of method, based on accumulated wisdom and empirical studies on a specific subject. The researcher performs a review on the subject analysing trends and forecasting situations. This is an approach used by authors like Charles Handy (Handy, C., 1994) or Alvin Toffler (Toffler, A., 1985; Toffler, A., 1970). Although this strategy could prove useful at certain stage of this thesis, it would still require the collection of naturalistic data.
2.5.2.3 Action research

This is a popular participatory approach in education research, based on the involvement of the teacher (the researcher) in the analysis of the results of his or her actions. This iterative approach is based on the seminal work of the social psychologist Kurt Lewin (McNiff, J., 1988). He suggests that research be conducted in a naturalistic way, as a spiral of steps between planning, acting, observing and reflecting (see figure 3). In this way, the researcher is aware of the effects of the different strategies being implemented and can take remedial actions as appropriate.

![Figure 2-1, Lewin's Action Research spiral of steps](image)

Action Research could be a useful approach for iterative development of solutions, specially in the organisational context. This approach would prove useful in longitudinal type of research, where there is an opportunity for multiple iterations. However, a PhD thesis needs a more concrete "blue print" definition of the method, as described next.
2.6 A Research Strategy for this thesis

The methodology used to support this thesis is a combination of some of the methods described earlier, as no one single method would have been appropriate (see Figure 2-2) to investigate the human costs of software upgrades in organisations.

There are two different but complementary streams in this thesis. One is an organisational perspective on how upgrades are perceived and managed (organisational psychology). Another one is the view that the individual users (and group of users) have as they receive and use software upgrades to perform their business tasks (cognitive psychology).

In order to evaluate the effects of upgrades in organisations it was critical to start with some type of study that would inform us about the reality of upgrades, the level of awareness management has regarding possible human costs and the methods organisations use to support the implementation of new versions of software packages.

Chapter 3, will explore the drivers that trigger a software upgrade in an organisational setting as well as identify the implementation strategies organisations use to support those upgrades.

The methodological strategy chosen for this first study is a survey, capturing the perspective from decision-makers and major stakeholders on software upgrades processes in different organisations (the details of the method are explained in chapter 3).
The reason this thesis begins by searching for a higher level, managerial picture of the problem of upgrades is because this provides scope and direction for other, more targeted studies, and helps in the understanding of what is possible (or not) to manipulate in an organisational context (see 2.3 Scope, pp. 13).

Figure 2-2 illustrates how the organisational views on software upgrades will eventually inform the recommended strategy, together with the findings obtained from a cognitive psychology perspective (this is indicated by a dotted line linking point 3 to point 7).
Figure 2-2, Research strategy adopted for this thesis
The next stage was to obtain the individual's perspective from their experience with previous upgrades. Chapter 4 covers this by gathering answers for the question "How do you experience software upgrades?". This was also performed through a survey, much larger than the one described earlier (details in chapter 4). These two studies differ in that:

• First, the target audience was different in each case: Management consultants and implementers were the target for the first study, while secretarial personnel (end-users) were the target for the second one.

• Second, the first study was characterised by open-ended questions, intended to collect opinions rather than quantify facts, which was the objective of the second survey.

• Third and most importantly, the first survey looked for upgrade drivers and implementation strategies, while the second one looked for the problems users experience, due to the upgrade of software.

The end-user survey (chapter 4) revealed compelling information about the types of problems people experience as a consequence of an upgrade on their software.

One of the major problems identified concentrates around the issue of transfer of existing skills. Chapter 5 reviews the literature in this area with the aim of developing a model that could explain some of the problems found in the previous survey. The search for the model provides a better understanding of the problem of transfer of skills and interference.
Building on the model developed in Chapter 5, Chapter 6 describes a field experiment that tests some of those theories in an organisational setting.

From the analysis made earlier in this chapter, and the discussion about the importance of ecological validity in the research for the applied world, this experiment was designed to be entirely field-based, targeting real users in their real environment as they experience a simulation of an upgrade. This simulation was created mimicking the type of changes upgrades normally have (details in chapter 6).

An "ideal" method to research this problem would have been to use the installation of a real upgrade, performing observations of user's behaviours. This would have proved methodologically difficult because:

- people rarely upgrade at the same time in an organisation;
- the researcher would have had to wait until an upgrade happens;
- management is sensitive about giving permission to observe real users in their day-to-day world;
- it would not have been cost-effective to observe for long periods of time until a problem appears.

As a consequence, this part of the research combined the power of an experimental situation where some variables were manipulated, with the ecological validity of a real environment.
The following Chapter 7 puts both research streams (individual and organisational) together, analysing different ways in which the interference problem (chapters 4, 5 & 6) can be minimised given the organisational limitations in the management of software upgrades (Chapter 3).

The last chapter (Chapter 8) concludes with the lessons learned as a PhD candidate and identifies further possible lines of research in this area.

2.7 Chapter Conclusions

This chapter reviewed various methodological approaches used for research in both cognitive and organisational psychology areas. Some of these methods were combined to meet the requirements of the present research, its strengths and limitations.

Using a top-down approach, two surveys were conducted using questionnaires and interviews, initially opening the research and narrowing it down as issues were identified. A model was created and then tested using a field experiment to preserve the ecological validity of the findings. Finally, a strategy is presented and recommendations for action research are made.

It is the next chapter (Chapter 3) then, that brings the organisational perspective on the research topic of this thesis: The human costs of software upgrades.
3. An Organisational View of Software Upgrades

"An upgrade of this product is available now. Please click <here> to start downloading"

Message displayed in a pop-up window as a user started a program one morning

3.1 Chapter Summary

This chapter explores the organisational perspective on the issue of software upgrades. It investigates the reasons behind upgrades, the implementation mechanisms organisations use to support upgrades and the level of awareness management have of the associated costs.

3.2 Introduction

Chapter 1 introduced the concept of Total Cost of Ownership (see page 7). In this model, organisations such as Microsoft Corporation and Gartner Group recognise the existence of important hidden costs of using software. They do not analyse these costs specifically in relation to software upgrades, however, as upgrades are an important part of today's use of software, they could be responsible for some of these hidden costs.

Today, it is difficult to find an organisation that is not undergoing (has not undergone recently or will not undergo in the near future) a software upgrade of some kind. The question then is whether or not organisations seriously consider all costs when deciding to upgrade their software.
The study that follows explores the drivers that make organisations upgrade their software, the implementation strategies they use and the awareness management has regarding the costs of an upgrade of software.

3.3 Method

To investigate this issue, direct interviews with decision makers in organisations could have been conducted, collecting information on why they upgrade, how they manage upgrades and what is their awareness level regarding the costs (particularly the human ones) of those upgrades. Identifying and gaining access to decision makers inside organisations would have proved difficult, as these people generally are less available for research purposes.

Another approach would have been to address users directly. Asking employees about how organisations manage their upgrades would have resulted in a array of diverse opinions, as people would report on how they experience the upgrades instead of how management intended those upgrades to be implemented.

In this study, in order to reach the decision makers in organisations, a group of management consultants were contacted instead. Their knowledge of the subject of upgrades is based on their experience in working in organisations and they were accessible to the researcher. Targeting the questions towards management consultants had the added value that their answers were based on their experience from working with several organisations, providing a well informed perspective, as they are professionals trained in analytical thinking.
3.3.1 Sample

The initial sample which the survey was directed to, consisted of forty management consultants with at least 4 years international experience in the design and installation of bespoke and off-the-shelf software. They represent three major consulting firms: Andersen Consulting, Perot Systems and Oracle Consulting in different locations: Australia(4), Canada(3), Czech Republic(1), Malaysia(1), Norway(1), Singapore(3), Spain(3), UK(13) and USA(23).

On average, based on my experience working in the consulting world, each of these consultants would have experienced a minimum of five projects with large client organisations, interacting with decision makers for the upgrade of software.

3.3.2 Questionnaire

A short questionnaire was designed (see Appendix 1) containing a brief introduction, a definition of software upgrades in the context of this study, and the following three questions:

1. What drives organisations to upgrade their software?
2. How are the implementation of upgrades supported in organisations?
3. How much awareness of the costs of upgrades (other than hardware and software costs) organisations have?
These questions were designed as open questions, to allow respondents to report freely, without restricting their answers to a prescribed list of possibilities. Since no similar study could be found it would have been difficult to identify the options for closed responses.

The number of questions was limited to three to keep the survey short, and encourage responses from the consultants.

3.3.3 Distribution
The survey was distributed to forty consultants through e-mail. Distributing the survey in an electronic way proved to be a sound strategy for the following two reasons:

1. All subjects work with e-mail on a day-to-day basis. Messages distributed through e-mail (for consulting organisations) are work related, and therefore not ignored. Paper-based surveys would have had to compete with junk mail in their offices.

2. Presenting the survey on-line, for this particular population, made it easier to reply. The survey was designed with appropriate spacing so subjects could reply to the original message adding their responses, without any editing effort.

Subjects were asked to reply and to forward a copy of the survey to colleagues, particularly those that were managing an upgrade in client organisations at the moment.
This method of distribution proved to be beneficial because it reached more people than the initial forty. In many cases, when subjects that received the survey felt there were other, more experienced colleagues that could reply to the message, they forwarded the original survey to them.

3.3.4 Return ratio

An interesting effect of the method used for the distribution of the surveys was that more responses (n=52) were received than original surveys (n=40) sent. At the time of writing, some responses were still being received, yet at a much slower frequency.

3.4 Results

Subjects responded with examples that illustrated their experiences, some of these are presented in this chapter. Descriptive responses were read and grouped into categories, according to their similarity, to facilitate the analysis and the later presentation to the reader. These categories were created as needed, to cater for all responses. Findings are reported in this thesis following each of the questions posted.
3.5 Question 1: “What drives organisations to upgrade their software?”

3.5.1 Findings

The drivers that make organisations upgrade their software are various and very diverse. For their analysis, they will be divided into two categories: Direct and Indirect drivers.

The term direct drivers in this thesis is used to reflect the internal explicit forces that request a software upgrade in order to improve the performance of workers in alignment to the organisational vision. Indirect drivers, on the contrary, are those that are not directly linked to the better performance of the workers but to other, more secondary objectives.

The distinction between direct and indirect only reflects the organisational perspective. The point here is that a driver considered indirect from an organisational point of view, could be, in fact, a direct driver for an individual, group or external organisation’s perspective. An example of this issue is the one described in 3.5.3.3., the “falling behind” syndrome. For an individual, upgrading their skills to the latest version of a software package makes him or her more marketable. However, using that as the only reason for an upgrade of software will not directly benefit an organisation.

---

3 A complete list of subjects' responses is offered in Appendix 1. Answers to question 1 (upgrade drivers) have been processed, representing the amount of direct and indirect drivers each subject identified, instead of using lengthy descriptions. Questions 2 and 3 are shown in textual form and have, in some cases, been edited to summarise key points.
Responses show that *indirect* drivers were more frequently mentioned than the *direct* drivers, as indicated in Table 3-1.

<table>
<thead>
<tr>
<th>Direct Drivers</th>
<th>Indirect Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number</td>
<td>Mean</td>
</tr>
<tr>
<td>53</td>
<td>1.01</td>
</tr>
</tbody>
</table>

The lists of drivers for both categories should not be considered exhaustive accounts of the forces behind software upgrades. Neither should any prioritisation be read in the order in which the items in these lists are presented.

The sole purpose of this part of the study is to identify and describe the reasons that most frequently drive upgrades of software in organisations (this was described as "casting a wide net" in the previous chapter).

**3.5.2 Direct Drivers**

The following 8 direct drivers are not organised in any particular order.
3.5.2.1 Required functionality

In this case, change is triggered when the organisation, department or worker needs new functionality. For example, in the case of a department that produces newsletters, a print preview function would allow them to view the end product before sending it to print. This is a function that they will benefit from.

3.5.2.2 Cost of maintaining existing system

Sometimes software requires a series of fixes to adapt to functional and technical changes in the environment. Maintenance costs could increase to a point at which an upgrade to a new version of the software is cost effective.

3.5.2.3 Competitive advantage

Subjects reported that some organisations upgrade their software to minimise steps and time required to perform particular tasks, resulting in faster turnaround. Companies expect a competitive advantage from a new version of the system. For example, in a call centre, a customer service representative is able to see more information per screen (characteristic of the new version of the software). This minimises navigation time, reducing telephone waiting time and increasing customer satisfaction.
3.5.2.4 Organisational changes

When two organisations merge, for example, it is one of them that most probably will have to upgrade to the software being used by the other one, in order to standardise their tools, processes and even training programs. This can also happen as a consequence of centralisation or decentralisation of processes.

3.5.2.5 Support of new job requirements

This is the case in which a job is enlarged and a worker has to perform new tasks in addition to the ones he or she was already performing. Job requirements change and the existing tool do not support work anymore.

A respondent wrote about her own experience:

"Many of us feel that we barely know what our jobs are. We aren't working in the same job we had last year, or even last month. For many of us, the job changes radically and constantly."
Look at the counter people in banks, for example, many of whom thought they were going to spend some measurable portion of their lives as bank tellers. How could they know that one day they would wake up as customer service representatives? Or how could your average insurance underwriter anticipate that one day he would be doing actuarial calculations for pets?"

An example of this issue: A report was produced in a particular department and printed for local distribution. It now needs to be mailed to other departments. As the tool does not support e-mail connection, the tool needs to be upgraded.

3.5.2.6 Improved productivity

This is the case of the wizards, small programs that automate and simplify functions that used to be too complex for a novice user. For example, the steps required to print the same letter using a database of people and addresses have been replaced with a new function: mail merge. This new function takes the user "by the hand" through what used to be a function that only expert users could master.
3.5.2.7 The new version allows the handling of objects in a better way

This is the case in a department that produces a particular report. It had to "typewrite" vertical titles for that report, as the version of PowerPoint they used would not allow text rotation. They upgraded to a new version of the tool that permitted handling text as if it was a graphical object.

3.5.2.8 Bug fixing

As technical problems are detected and reported, new versions are released in order to fix those problems. This has been one of the major drivers of software upgrades in the past, getting the name of "fix", instead of "version".

3.5.3 Indirect Drivers

There are other reasons, not directly related to the support of the performance of the workers, why organisations upgrade their software. All respondents said they had experienced or observed at least one of the drivers listed in this category.
3.5.3.1 The current version will not be supported by the hardware or software provider in the future

Discontinuing support of a particular version of a product is perceived by some as a strategy software vendors use to encourage upgrades and, therefore, increase sales. Vendor's justification is that they need to use their support resources to service requests associated with "current" versions. For example, Microsoft discontinued as of this year all support for Word® 1 under both the Macintosh and Windows platforms. Although the number of users that currently use those versions is unknown, this creates an upgrade pressure for those that use this version of Word®.

Microsoft, a young software company with only 20 years in the market, has published a list of the software they created which they now consider "obsolete" (and therefore without support). This list contains 102 software products.

3.5.3.2 "We don't want to fall behind" syndrome

Management welcomes upgrades as a means of not "falling" behind the market.

This driver was one of the most frequently reported in the survey. The need to be "up to date" was reported to originate from technical departments or even from top management. This need was found to be non-related to the support of the performance of the end-users. Moreover the users were rarely consulted on the decision.
In a recent study of organisations upgrading to Windows 95, Dataquest reported that "(managers) do not want to fall too far behind the technology curve, regardless of whether their businesses are functioning well with Windows 3.1" (Dataquest, 1996).

Vendor organisations appear to take advantage of the "keeping up with the Jones" phenomena in their commercial approach (see Figure 3-3, below).

Figure 3-3, Advertisement for Lotus Corporation, targeting Windows 3.1 users.
3.5.3.3 Feeling "behind" change

This driver is similar to the previous one, but driven by the end-user's need to "keep up with the Jones". Some users feel they need to catch-up with the version of the software they use, in order to be more "marketable" inside and outside the organisation.

It has become vox populi that continuously mastering the latest version of a software package is an important component of today's jobs.

One of the respondents described this driver as follows:

"Why do we feel we are always behind? (We're) tapping into the most primal fear of the modern worker. It is the negative side of the job transformation idea. You know your job is going to change, whether you like it or not.

If you wake up 'behind' the change, you are a fatality. You can't afford to let any job-changing events happen outside your control, so you upgrade your software compulsively"."
3.5.3.4 Organisations that want to utilise new technology

The survey showed that some "organisations are largely reactive to software innovations", as one of the respondents put it, keeping up with emerging technologies.

This is the case when management has a strong commitment towards technology and want to invest in IT whenever possible. This is characteristic of companies whose main product is IT and who want to use the latest technology available and even manifests itself in some organisation's mission statements.

However, even in non-IT companies this phenomenon appears. For example, according to one of the respondents, the mission statement for a water company in the north of England reads: "We want to be the most technically advanced water company in the world...".

3.5.3.5 Hardware-Software dependency.

This is the case when in order to run a new version of software there is a need for an upgrade of hardware. At the same time, new purchased hardware comes with "free" software, that would take maximum advantage of all the characteristics of the hardware. Therefore the original software is upgraded.

This circular effect can be observed when installing modems, pointing devices, memory, math co-processor, etc.
One of the respondents described a situation in which users, attempting to connect to a new on-line service, found they had to install a faster modem than the one they already had. Connecting the new hardware required a careful installation process. This was supported by a piece of software, part of the package.

After the installation was completed, this software suggested that a particular communication package was installed to fully benefit from the capabilities of the modem that was being installed. To install the new communication program users had only to confirm with <OK> to a question in a dialogue window.

3.5.3.6 Part of a “package” (Office, Smart Suite, etc.)

Software “suites” are packages that bundle different desktop applications together (i.e. word-processing, spreadsheets, graphic package, etc). When installing a suite such as Microsoft Office or Lotus Smart Suite, users inevitably replace their existing tools with the newly installed ones. Maintaining an older version of one of the tools and upgrading the others would result in added technical complexity and incompatibilities between the tools.
3.5.3.7 *It comes with the new computer*

Many respondents identified this as a very frequent driver of software upgrades. Software is pre-installed with most new computers today and is a good marketing strategy for vendors. Very rarely would users uninstall the new software, replacing it with the one they currently use. In most cases it is taken as an unavoidable call for an upgrade of skills.

3.5.3.8 *Change in job requirements*

Respondents identified that changing software has become an excuse for the exploration for the purpose of a job in some cases. It has become *vox populi* that for job redesign to be significant it has to involve a change in the IT component, even when the previous version could have supported the new functions of the job.

3.5.3.9 *Compatibility with other users / department / organisation / clients / suppliers*

This driver was also raised in some of the items of the previous list: Direct Drivers. When organisations merge, some departments would upgrade their software to be compatible with the rest of the organisation; and there is nothing wrong with that.
In this other case we refer to the situations in which users (or groups of users) play “leap-frog” with other parts of the organisation, upgrading their software to gain compatibility. This micro management of upgrades could solve some compatibility problems, but it creates others.

3.5.3.10 Compatibility with operating systems and tools

This is a similar issue to the one mentioned before, with the difference that in this case the compatibility problems are experienced within each computer. This is the case of Windows 95, which forced users to upgrade their versions of Word® 6 to Word® 7, so they could run under the new platform.

3.5.3.11 Upgrading to fix previous upgrade

Management decides on buying a particular version of software. This product is installed and either does not completely satisfy the functional needs of the organisation or it has many technical problems to run to satisfaction. Management decides on an upgrade to support their original decision, in the hope that the new version will solve some of those problems.
3.5.3.12 The upgrade is free

Upgrades for some software (i.e. Netscape, Virus protection software) are offered free of charge. Users can just download new versions from the internet and clever programs will install the upgrade in their machines, identifying existing versions to be deleted (see an example of a free upgrade Screen 3-1, below).

Screen 3-1, Advertisement where Microsoft offers a free upgrade to Exchange 5.0
3.5.4 Discussion

Twenty different reasons that drive organisations to upgrade their software were found in the present study. Eight of them directly support the performance of the user population while the other twelve had a different, less value-added origin. These twelve indirect drivers showed that organisations embark on software upgrades, even when there is no real need for them.

In most cases, upgrades are perceived to be symptoms of progress, modernisation and competitiveness. Many organisations are “victims” of software upgrades, not having much control over when to upgrade so it would be beneficial to the workers and therefore to the organisation itself.

The emergence of “push” technologies encouraging software upgrades could be one of the primary reasons for this lack of control. Most respondents seemed to agree with what the author Thomas Landauer writes in his new book on upgrade decisions in organisations: “...buyers are often left largely at the mercy of vendors” (Landauer, 1996).
3.6 Question 2: “How do organisations manage the implementation of upgrades?”

3.6.1 Findings

According to the survey (see Appendix 1), organisations believe that upgraded versions will be intuitive if their employees already function with the current version of the same software. Therefore, dealing with upgrades is in most cases a technical issue, dealt with by technical departments. The major decision is whether the installation of the new version should be done automatically, using the networks for its distribution, or if personnel from the technical department should install each workstation individually.

The survey showed that most organisations do not consider an upgrade as an important change that has to be managed. The reasoning behind this is that the implementation effort has already taken place when the application was installed in the first place. Software upgrades are considered to be the addition of benefits, anticipating very little or no costs from the user's perspective.

The most frequently mentioned approach is to make changes as invisible as possible for users in order to minimise disruptions. Upgrades are normally performed outside of working hours and users are notified of the new version through internal mail.

Managing upgrades from a user perspective was rarely reported. In best cases, management notifies users of the upcoming version of the software while support departments distribute an upgraded manual.
One of the respondents summarised the organisation’s approach to the management of upgrades as "Poor and with a fair amount of wasted effort". Another subject, in relation to the user’s reaction to how upgrades are managed reported: "...I have seen it so bad that there were people taking stress leaves and threatening strike...".

3.6.2 Discussion

The implementation stage should be considered as one of the most critical phases of a system installation (or of an upgrade in this case). The fact that upgrades are often made for indirect reasons makes the implementation of the upgrade even more critical.

When users do not really need the upgrade, their capacity for the assimilation of the change will be negatively affected. A different situation would be when end-users require an upgrade as they know it will provide solutions to their existing problems.

Much has been written about managing the implementation of software, and its associated costs. Eason (1984), for example, describes the importance of dealing with “emotional” changes before users embark into learning and cognitive gains. He proposes a checklist for identifying benefits versus costs of a new software implementation (see Table 3-2).
Table 3-2, User Cost-Benefit Assessment (adapted from Eason, 1988)

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ability to perform tasks more effectively</td>
<td>Financial costs (these are the most frequently accounted for)</td>
</tr>
<tr>
<td>Desirable changes in the nature of the job</td>
<td>Effort it takes to use a system</td>
</tr>
<tr>
<td>Improved salaries</td>
<td>Loss of job security</td>
</tr>
<tr>
<td>Greater power</td>
<td>Effort to learn and adapt</td>
</tr>
<tr>
<td>Influence or enhanced career prospects</td>
<td>Risk of failure</td>
</tr>
<tr>
<td></td>
<td>Loss of job satisfaction</td>
</tr>
<tr>
<td></td>
<td>Loss of privacy</td>
</tr>
</tbody>
</table>

In the case of the survey, in which the software implementation is an upgrade, respondents did not recognise the benefits listed in the model proposed by Eason, but did identify most of the costs, leaving the equation for the upgrades somewhat unbalanced.

3.7 Question 3: “How much awareness organisations have of the hidden costs of upgrades?”

3.7.1 Findings and discussion

All respondent agreed that organisations do not account for any costs other than software, hardware and training costs when deciding on an upgrade of software (a detailed account of the answers can be found in Appendix 1).
The benefits and counter-balancing costs of the introduction of information technology in the workplace are widely accepted these days. Almost no-one would argue today that a new IT implementation does not impact the way people work, their relationships, perception of self, etc. However, organisations tend to underestimate costs when software is upgraded. A respondent wrote: "The training costs and productivity impact seem to be overlooked in the upgrade planning process".

Not only does management not account for costs other than software, hardware and initial training but they believe that such costs do not exist or are so distributed that they should not be considered. Opposite to this are the figures that The Garner Group obtained through their research (discussed in section 1.2, page 7). They estimate that 56% of the TCO can be attributed to end-user operations. However, in the research conducted for this thesis, 86% of the subjects reported that organisations either are not aware or do not consider these costs when deciding on a software upgrade.

3.8 The paradox of the minimisation of hidden costs

Microsoft is already concerned about non-accountable costs of owning software (as discussed in section 1.2 page 7). In their attempt to minimise (and possibly eliminate) those operating costs, Microsoft is launching the Zero Administration Initiative for Windows.
3.8.1 The Zero Administration Initiative for Windows

Microsoft is planning the creation of a tool that will permit central administration of the workstations in a large company. With this new software, off-the-shelf products and operating system upgrades could be sent automatically to the PCs on the network, causing no disruptions to workers and reducing installation times and IT personnel. The IT department would re-gain control over what software is used and who uses it; a control that was lost with the advent of personal computers.

3.8.2 The paradox

This reduction of the administrative costs of software could have no effect on the user costs of owning software. On the contrary, we can hypothesise that reducing the visibility of upgrades, minimising the contact users have with the installers of upgrades, taking control away from the users, and having automatic installation of upgrades (that will help upgrade faster and more frequently) will exaggerate problems users have in relation to upgrades, increasing their resistance to new technologies.

3.9 Chapter Conclusions

The study described in this chapter could have benefited from the addition of demographic data and some information on the individual weight of the responses in the survey.

However, there are two main conclusions that can be inferred from this chapter:
The first one is that there is a danger that organisations upgrade their software not in response to end-users or real business needs, but as a consequence of the different pressures they are target of (i.e. vendors, competition, market).

The second conclusion is that upgrades are managed as a technical issue, underestimating the effects they could have on the user population. Those responsible for upgrade decisions assume that users will benefit from the upgrades, providing more functionality to the existing tools. There is an assumption that user skills don't need “upgrading”, as people will “intuitively” adapt to the new version.

In conclusion, if upgrades are triggered by external factors and they are managed only technically, we could predict that most of the people on the receiving end of upgrades would experience difficulties. The next chapter of this thesis will explore the end-user's experiences on the upgrade of software.
4. A Survey of Problems Caused by Software Upgrades

"If you stop and think about it, you realise that the rate of innovation in the personal computer industry is great for consumers. Every couple of years work gets easier because tools are so much better. What's wrong with that?"


4.1 Chapter Summary

The present chapter describes a study designed to identify and understand the problems end-users experience when upgrading software tools in order to contrast these findings with those obtained from an organisational perspective in the previous chapter.

4.2 Introduction

The study described in the previous chapter (see Chapter 3) identified that most organisations base their decisions of upgrading software on motives external to user needs. The study also identified that very little is done to support users assimilate the upgrades. Most organisations ignore the total costs upgrades, particularly those related to how users cope with the almost continuous arrival of new versions of software.
The findings from the previous chapter suggest the following questions: Are organisations right to "ignore" the human side of the implementation of a software upgrade? Do users easily adapt to upgrades after brief learning curves? How transparent are upgrades to end-users? These are the main questions the present chapter will intend to answer, completing the organisational view first presented in Chapter 3.

4.2.1 Method
A higher level explanation of the methodology for this thesis was described in Chapter 2. We will concentrate here on specific methods used in the decomposition of the steps taken for this particular part of the study.

To answer the question of how end-users experience software upgrades inside an organisation a number of methods could have been used: Direct observation, self reporting, surveys, interviews or focus groups. All other methods which could have been used, would have ignored the user (i.e. asking the management) or the organisational context (i.e. laboratory experiments).

Direct observation was initially considered but later discarded because of impracticality. Problems caused by software upgrades do not necessarily appear altogether, at a particular time. This would have caused the experimenter to perform observations for long periods of time without a guarantee of success. On another note, the presence of the experimenter would have caused a Hawthorn effect (Wilson & Corlett, 1992), influencing the performance of the subject (for a discussion on this topic see 2.5.1.2 Field experiments, page 19).
Another possible method to gather this type of data could have been self-reporting, in which subjects are given a diary where they report the problems they encounter with software upgrades as they happen. This method was discarded on the grounds that the target population (see 4.2.2 - Sample, page 61) wouldn't have had the incentive to keep a diary with the problems related to software upgrades. This would have also affected the ecological validity of the study, since creating an extra task that would have continuously reminded them of the existence of a new version of the software.

The method that could best gather the data needed was a questionnaire. This decision was supported based on the following facts:

- Questionnaires could be easily distributed through the university's internal mail system;
- A large population could be initially reached, since at this stage there was no information on who the possible subjects were (we define as subjects those users that have upgraded software over the last year);
- Questionnaires are not as intimidating as direct observations can be;
- Questionnaires in which users recall a previous experience do not need to interrupt subjects normal work.

Some drawbacks for this method were also considered. Despite of the fact that questionnaires could have low return ratios and that “after the fact” reporting could affect people’s recall of details, the use of questionnaires showed to be the best alternative for this part of the research.
It is important to mention that the main purpose of the initial questionnaire was to identify possible subjects from a large population. This cascade method permitted an identification of subjects that had experienced a software upgrade over the last year, followed by the identification of the problems they could report on. Once key subjects were identified, one-to-one interviews were conducted to discuss the problems they experienced.

In summary, the sequence of methodological steps followed in this study were:

- **Sample identification:** The sample was first identified, analysing its representation of the overall population in terms of age, type of usage of IT tools, distribution, support levels, background, etc.

- **Questionnaire design:** A first version of the survey was then designed following an analysis of the type of questions that would prove more efficient.

- **Pilot:** The survey was pilot tested, modified and distributed to the sample.

- **Distribution:** The survey was distributed to all "possible" target subjects, as there was no information that would identify who had experienced an upgrade over the last year. Phone and e-mail support was provided throughout the period in which surveys were being completed, helping respondents understand questions from the survey.
• Interviews: Subjects who use off-the-shelf software and had upgraded over the last year were finally identified, together with a high level picture of the problems they experience in relation to software upgrades. Twenty of the identified subjects were contacted for individual interviews in order to explore with more depth the type and frequency of the problems described in the survey. The data from the questionnaire and that from the interviews was processed and analysed.

The following is a detailed explanation of the steps listed above.

4.2.2 Sample

The population target of this study is constituted by full-time white collar workers, that use off-the-shelf information technologies to perform their daily tasks in an organisational context.

The decision to study off-the-shelf technology upgrades (i.e. word-processors, spreadsheets, presentation and communication packages) reflects findings from Chapter 3, where Organisations were found to be largely influenced by vendors in the area of off-the-shelf technology (indirect drivers).

After an analysis of the characteristics of the population in study, the group of secretaries at Loughborough University was chosen as it was considered to be a valid sample. The following are the reasons that support that decision as well as a description of the chosen sample:
4.2.2.1 Number

Loughborough University employs approximately 450 clerical workers, most of them performing secretarial type of work. There was no one source of information that would identify who, out of the 450 workers uses off-the-shelf technology and has experienced an upgrade over the last year. For that reason, the survey was sent to all 450 employees, knowing that not all of them would be target for this study.

4.2.2.2 Variety in skills and experience

The University employs people with a variety of skills and backgrounds for secretarial tasks, as these change from department to department. The University has a surprisingly low turnover of personnel (less than 13% in 1994 considering transfers to other departments as part of this figure). It is not rare to find employees that have worked at the University for nearly 20 years.

The University has also a part-time working scheme allowing the employment of younger individuals with less experience while they finish their studies at local colleges. Some individuals are highly experienced in the use of information technology tools as they have had in-depth exposure to different platforms during their work experience. Others managed to learn "just enough" as was reported during some interviews. The issue of whether this reflects minimisation by the learners or constraints of the task environment they are given remains to be seen.

Most of the user population falls in between these two extremes and does represent a distribution of user types and skill levels.
4.2.2.3 Variety in usage

Although subjects were all secretaries working at the same University, they belong to different departments, showing a wide range in usage of the information technology tools provided: from processing student records to logging access to the University gates, from simple word-processing to the creation of newsletters, from the transcription of chemical formulae to the operation of communication packages.

4.2.2.4 Different platforms and technical environments

Although the University has defined standards, these are not enforced. Very diverse combinations of hardware and software are used on campus, ranging from Personal Computers to Apple Computers to centralised mainframes.

The most common software packages that are used at Loughborough University are Word for Windows (versions 2,4,5,6 and recently 7) under PC and Macintosh platforms, WordPerfect, Excel (versions 5 and 6), PowerPoint, Corel Draw, Netscape and Eudora, among others.

4.2.2.5 Individual work or group work

There are some groups at the University that are highly integrated in their tools, sharing information as part of their daily routines. Others have information technology tools for their own benefit, and create and maintain documents that will not be shared with any other individual or department at the University.
4.2.2.6 Different levels of support

Some departments have their own information technology “manager”, responsible for the installation of software and support in its usage. In most of the cases this person acts as an active link to Computing Services, facilitating the standardisation of solutions.

In other situations, as is the case with smaller departments, people get support from colleagues, documentation or phone calls to technical support representatives in Computing Services. There is an important number of workers that perform their jobs in a very isolated environment (e.g. secretary of a small group) experiencing very low levels of external support.

4.2.2.7 Age

The age represented among the secretarial staff at the University ranges from 19 and 60. This range overlaps well with the age from the population in study.

4.2.3 Questionnaire Design Rationale

A sample of the questionnaire utilised can be seen in Annex I.

4.2.3.1 Question types

This survey utilises both Open and Closed questions (Oppenheim, 1992). Open questions are those that allow the respondent to “fill in the blank” with their own responses to questions or statements (this is exemplified in Figure 4-1).
An advantage of this type of questions is the freedom and spontaneity of the answers. On the other hand, having to write the answer to questions is time-consuming and could be discouraging for respondents. Open questions can prove useful when the researcher needs more information about a particular problem. Not only does this type of question demand more effort from respondents but from the researcher as well, as answers need to be codified to be systematically analysed, being this a tedious and unreliable process.

The other type of questions used in surveys are the closed questions. In this case respondents have to choose from an existing selection of possible answers. Using the same example as before,
Contrary to open questions, closed questions require little time for the respondent and are much easier to process for the researcher. On the negative side, answers to these types of questions are quite crude, forcing the respondents to fit their opinions into a pre-established set of categories.

The objective of the survey was to identify the problems people have as a consequence of a software upgrade. In order to gather as much information as possible about the type of problems, the usage of open questions was appropriate. However, people that receive a self-administered survey may have no particular strong motivation to complete it. For this reason every effort was made to keep it short, simple and still get the information that was needed, using mostly closed questions. Even in those cases where an open question was used, it was done in combination with a closed one, to capture a high level answer before the respondent decides to abandon the survey because it was being time consuming. An example of this approach is,

<table>
<thead>
<tr>
<th>How would you rate your experience with the photocopier machine?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poor □ Fair □ Good □ Very good □</td>
</tr>
</tbody>
</table>

Please describe

Figure 4-3, An example of the open-closed combination used
The main objective of this survey was not to quantify the problem but to have proof of its existence that would permit further investigation through a different method. Quantification would have required a more constrained or defined scope, targeting the survey for a particular group of people upgrading a particular tool in a comparable way.

4.2.3.2 Organisation of the Questions

The questionnaire was designed to be short in length, consisting of only four pages. It begins with an introductory letter stating the purpose and scope of the study and guaranteeing confidentiality for the respondent. It concludes thanking respondents for their time, giving support contact numbers and offering the distributions of results from the study.

The questions in the survey are divided into four categories: Demographic, IT experience, Context of the upgrade and Problems with upgrades. The first two questions gathered demographic data from the respondents. As confidentiality was guaranteed, respondents had the choice of identifying themselves and their departments.

It was important to obtain information about subjects level of exposure to previous changes, IT experience, etc. This was collected through the third question:

<table>
<thead>
<tr>
<th>Experience with computers</th>
</tr>
</thead>
<tbody>
<tr>
<td>less than a year □</td>
</tr>
<tr>
<td>1 to 5 years □</td>
</tr>
<tr>
<td>5 to 10 years □</td>
</tr>
<tr>
<td>more than 10 years □</td>
</tr>
</tbody>
</table>
This question was important for a later correlation analysis between problems experienced and level of previous exposure to information technology tools.

Questions 4, 5, 6 and 8 collected contextual information on the upgrade itself, identifying the tool, version, how the upgrade originated and how it had been supported.

What software package have you recently (e.g. over the last year) upgraded?

name __________________________ from version ________ to version ________
PC ☐ Mac ☐ PC ☐ Mac ☐

The upgrade was requested by
me ☐ my department ☐ the university ☐ other ☐

All members of my department upgraded at the same time Yes ☐ No ☐

Did you experience any incompatibility problems because people were using different versions of the same product?

Yes ☐ No ☐
Please describe __________________________

Did you receive any formal support to learn the new version of the software?

Yes ☐ No ☐
If yes, what sort of support did you receive?

a training course ☐ on-site training ☐ colleagues in my department ☐
telephone (hotline) ☐ manuals ☐ other ☐
Although these questions are not focusing on the problem of transfer directly, they were important as they helped in the better identification of problems. As will be discussed later, it was important to try to establish possible correlation between number of problems and levels of support or magnitude of change.

The next questions (numbers 7 and 9 to 14) collected subjective information on the problems, comparing the “old situation” with the “new one”. Subjects had to judge through different variables their perception of the new version.

<table>
<thead>
<tr>
<th>The new version, compared to the previous one... (please place an “x” in the scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
</tr>
<tr>
<td>is slower</td>
</tr>
<tr>
<td>is more difficult</td>
</tr>
<tr>
<td>I make more errors</td>
</tr>
<tr>
<td>is more difficult to learn</td>
</tr>
<tr>
<td>slows me down</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>does NOT help me with my work</td>
</tr>
</tbody>
</table>

The scale used proved to be an effective way to collect subjective information in a quantifiable manner. This was translated into a 10 point scale, considering 5 to represent “no change” between old and new. This scale was again used to capture the respondent’s overall satisfaction level with the change.

<table>
<thead>
<tr>
<th>In general, with the new version I am... (please place an “x” in the scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
</tr>
<tr>
<td>less satisfied</td>
</tr>
</tbody>
</table>

69
Question 16 was designed to collect information about the time during which the respondent's performance with the new tool was perceived to be lower than before.

<table>
<thead>
<tr>
<th>How long did it take you to master the new version?</th>
</tr>
</thead>
<tbody>
<tr>
<td>less than a day ☐</td>
</tr>
<tr>
<td>more than a month ☐</td>
</tr>
</tbody>
</table>

The following two questions were created to capture other possible issues not covered with the existing items. They were designed to be open ended.

**What advantages did you experience with the new version?** (please comment briefly)

**What disadvantages did you experience with the new version?** (please comment briefly)

The next questions collected information on the problems of transfer. Because at this stage it was important to keep an open mind about the types of problems to be found, questions 19 and 20 are very general. Responses on these questions were later used to identify types of problems that had to be discussed during the follow-up interviews.
Did you ever try to use commands from the old version that are no longer available in the new one?

Yes ○ No ○

If Yes, please comment

Is this still happening?

Yes ○ No ○

If Yes, how often?

The last question (21), an open-ended one, was intended to capture other related issues that were not reported before.

Can you add any other comments related to the upgrade of software?

21

4.2.4 Pilot Test

A first version of the survey was piloted with three randomly selected subjects from the sample group. The pilot was intended to test the questionnaire for readability, ease, accuracy and time to complete. This was performed in a self-administered way and was followed by a detailed interview where subjects gave their interpretation on the questions of the survey and verbalised their answers.
After this interview some questions had to be re-phrased and the presentation of the survey re-formatted. Some examples of these changes are: the addition of a question to gather information about hardware changes from Macintosh to PC (experienced by many of the respondents), the addition of a question related to compatibility issues and the rephrasing of question three, from “has more errors” to “I make more errors”. The survey was then ready for distribution to the rest of the sample.

4.2.5 Distribution

The distribution and collection of the survey was done through internal mail. A return address label was provided with each copy of the survey, to help respondents return the survey easily. The department of Staff Training and Development “sponsored” the survey through an internal communication to the senior secretaries, encouraging a prompt response. Senior secretaries would then encourage their teams to respond to the survey.

4.2.6 Responses

From the 420 questionnaires distributed by internal mail 63 (15%) of them were completed and returned, most of them within two weeks. The survey was sent to all secretarial and clerical staff at the University, regardless of whether they had recently upgraded their software, or whether they were using computers at all. A low return ratio was therefore expected.
The University does not have centralised information on who uses information technology tools for their work. Far less realistic was to try and target only those that upgraded their tools over the last year so they could report on it. In order to compensate for this lack of specificity in the distribution of the survey, it was made clear in the introductory paragraph that the survey was only to be completed if the subject had experienced a software upgrade over the last year.

Of 63 responses, 43 (10% of the total) were from users reporting on software upgrades that had taken place over the last year, 20 responses came from subjects that either did not use IT or had not had experienced an upgrade lately. Only the responses of the 43 who experienced upgrades are reported in the results section below.

Because only subjects that use IT and had experienced an upgrade over the last year were encouraged to respond to the survey, there is no data about the number of clerical workers at the University that do not use IT or have not experienced an upgrade over the last year. The IT department at the university does not have this kind of information, fact that again supports the findings from Chapter 3.

The completion and return rate for the survey was considered acceptable as a much larger population was initially targeted. The purpose of the study was to identify the problems caused by the upgrade of software and it was achieved.
4.2.6.1 Demographics

Most of the respondents (46%) have between 5 to 10 years experience with computers (the distribution of experience is shown in Figure 4-4). Eighty-eight percent are full time employed by Loughborough University (12% are part-time) and 100% of the respondents were female. As described above, the university could not provide information to ascertain that the sample was representative of the total population.

Figure 4-4, Distribution of experience

4.2.7 Interviews

The questionnaire described so far helped us identify major areas were problems could be found. The purpose of the interviews described in this topic, however, was to verify, explore and have a first hand observation of the problems reported.
Twenty subjects from the forty-three originally identified were randomly selected for a personal interview. It is important to note that not all forty-three provided an identification and some of them would not be available for an interview.

The interviews were conducted in a semi-structured way. A script was used to conduct the interview providing an organised flow (see Appendix 2). Subjects were strongly encouraged to demonstrate the problems they were describing, using their own equipment. This helped the interviewer develop a clearer understanding of the nature of the problem. The length of the interviews was 35 minutes on average.

As subjects were reporting on their own performance problems, it was very important to maintain the confidentiality of the meeting. It was initially intended to record the meetings on tape, however, the pilot studies showed that subjects were less likely to report problems if recorded. Notes were taken whenever necessary as this did not discourage subjects from talking freely.
4.3 Analysis of Results - Questionnaire and Survey

4.3.1 Overall findings

Subjects were asked to report on the length of time it took them to feel comfortable with the new version of the software. Responses indicated that adaptation to the new versions were not immediate (only 35% of the respondents took a day or less). There is a significant number of respondents that could not adapt to the change without effort (Figure 4-5).

![Figure 4-5, Time to master new version]

Subjects were also asked to identify the origin of the upgrades. Data shows that most upgrades (72%) did not happen as a need for the end-user, but were requested by either their departments or the University (Figure 4-6).
Subjects were also asked to report on implementation aspects of the upgrades. To the question of whether the upgrades were formally supported, only 19 subjects (44%) reported receiving a specific course on the new version. Data shows that most upgrades were only supported through the use of user manuals (30%), leaving users to "figure it out" by themselves (Figure 4-7).
End-user's responses on the implementation of the upgrade and its drivers reinforce the findings obtained in the previous chapter (Chapter 3), from an organisational perspective.

The reported times to master the new versions of software is an indicator of possible problems for the end-users when upgrading software. Next section will explore those problems.
4.3.2 Particular problems found

The following analysis is based on the data from the survey. Textual information from the survey (i.e.: observations, “what do you think about...”, etc.) were tabulated to aid the analysis (this can be seen in the last 5 columns of the spreadsheet reporting the results from the survey in Appendix 2).

From the 43 respondents of the survey, 10 were not software upgrades as we defined for this thesis (see 2.3 Scope). Most commonly, these surveys reported changes of software (i.e. WordPerfect to Word) or changes of hardware (i.e. Mac to PC) instead of an upgrade of version of the same software tool in the same context. However, all responses were considered for the analysis bearing in mind the type of change subjects reported on (a complete account of all the responses can be found in Appendix 2).

Data from the survey shows that 87.9 % of the respondents experienced problems of some kind when upgrading their software. The problems reported belong to one or more of the following categories: Transfer problems, compatibility problems, response time problems and configuration problems.
The figures given in the chart (see Figure 4-8) are percentages of respondents that reported problems in each particular category. Thus, individual subjects reported problems in more than one category.

Most users reported problems in the first two categories: Skill transfer and compatibility. Fewer respondents complained about response time and loss of configurations. Other problems outside these four categories were also reported (i.e. loss of data) but were not frequent enough to be considered in this study. A detailed analysis of the four major categories follows.
4.3.3 Skill transfer problems

Through the interviews it was observed that most problems were related to the usage of commands or methods learned in older versions of the software. All subjects surveyed had used similar software before and most of them (72.7%) had not been given formal training with the upgrade to the new version.

Transfer problems caused by previous knowledge surfaced when subjects reported using old commands, searching for old commands and using old procedures.

4.3.3.1 Using old commands

Most commonly, subjects experienced problems as they tried to use well learned commands in a new environment where these commands were no longer available. For example, short-cut keys will trigger functions other than the intended ones. In previous versions of PowerPoint (a presentation package), the command Ctrl+N added a new, blank slide to a presentation. In the current version of this software, Ctrl+N opens a blank presentation and Ctrl+M adds a new slide to the active presentation. Users found this very annoying as in some cases it will perform unwanted functions.

4.3.3.2 Searching for old commands

In other cases, subjects reported spending too much time looking for commands that they knew existed but had been placed under a different menu structure from the one they were used to or their name had been changed (e.g. preferences in Word for Windows 5.1 has been renamed as options in version 6).
4.3.3.3 Using old procedures

This is different to the previous problem as is much less visible. Users, in this case, are not looking for a specific command but trying to apply a procedure which is no longer available. For example this was found when subjects complained about the Mail Merge command in Word 6 for Windows.

Compared to previous versions of the software, the objects that need to be identified are different. This challenges the users' understanding of the function and not just the labels that it uses, causing frustrations. In some cases this "old understanding" of the function persisted months after the upgrade.

Some users reported going back to the previous version of the tool, so they could perform the task without problems. It is important to say that re-installing the old version is in itself a very time consuming task.

4.3.4 Upwards / Downwards Compatibility problems

Another problem found in 58% of the respondents is the one of compatibility. This was found not only between users using different versions of the same tool but also encountered within user's own environments. Upgrading one tool, in some cases, caused incompatibility when sharing data with another tool, in the same machine.
An example of this problem was reported by users that had upgraded to Word 6. When sharing documents with other members of the department, who were still using a previous version of the tool, they had to remember to <Save as...> a lower version of the software. However, as they had to work using a lower version, they could not import graphics into those documents using other components of the suit they had installed.

4.3.5 Response time problems

24.2% of the subjects reported being slowed down by the upgrade as a consequence of hardware-related issues. As new versions of the tools add more functionality, they utilise more memory and require more processing resources. In many cases this meant slower response time and frequent freezing of the applications.

The slowness in the performance of new versions of software has been technically demonstrated many times. For instance, in a study reported in the MacUser magazine (MacUser, 1995), three versions of Microsoft Word were tested for speed. Two measurements were taken: The time (in seconds) that each application will take to launch and the time it would take to perform a word count of 7000 words (see Figure 4-9 and Figure 4-10, below).
Both measurements showed that version 6.0 proved to be much slower than version 5.1a. Microsoft released a new version in order to minimise this effect. Although faster than 6.0, version 6.0.1 is still slower than the first one, 5.1a.
4.3.6 Loss of configuration problems

Upgrading versions of the same software caused 6.1% of the subjects to lose their existing customisation of the tool. Some users adapt the toolbars or create macros that help them in their performance. It was observed that in all the cases, this customisation was lost, creating frustrations and forcing users in some cases to re-configure the tool. This is a very time-consuming task for most of the users.

In other cases, when losing the existing configurations, users decide to use the standard configuration (even if it is more confusing and has functions they will never use). This decision was explained as a conservative approach to prevent configuration problems from happening again. This was an important finding in the interviews. Most of the users decided not to invest too much time learning or adapting the tool as they "know" it will soon be upgraded again.

Those users that decided to re-customise their tools, reported spending long hours in this task. In some cases this meant extended working hours for a week. Some subjects reported that the new version did not allow for certain customisations they were already used to.
4.4 Correlation Analysis

An analysis of the relationships between variables was undertaken. In this case, all 43 respondents were considered and an extra variable was added to categorise the type of change (see Table 4-1, below).

<table>
<thead>
<tr>
<th>Severity of the change</th>
<th>means</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (minimum)</td>
<td>same SW, same HW</td>
</tr>
<tr>
<td>2</td>
<td>same SW, different HW</td>
</tr>
<tr>
<td>3</td>
<td>different SW, same HW</td>
</tr>
<tr>
<td>4 (maximum)</td>
<td>different SW, different HW</td>
</tr>
</tbody>
</table>

Table 4-1, Codes used to identify the severity of the change

Using all responses (n=43) for this analysis, including those that reported on upgrades that included the change of hardware and even the change of software, supported critical findings described in the following sections. The more subtle type of upgrades where hardware and software remain unchanged while a new release is installed (i.e. Word 6 to Word 7) were rated with a 1; at the other extreme, changes that involved new hardware and software were considered a 4 in the category “type of upgrade”.

Significance was calculated using Contingency Tables with \( \chi^2 \) (Chi Square). The values reported include \( \chi^2 \), degrees of freedom and significance. The most significant findings are presented below.
4.4.1 Time to master

The time it took respondents to master the new version of software showed some correlation with the usage of commands from previous versions ($\chi^2 = 6.28$, df=4, $p<0.250$). The problem of interference in transfer (described in 4.3.3-Skill transfer problems, page 81) is of primary importance in this thesis and will be addressed in depth in the following two chapters.

Respondents adapted to the new version faster when their department or group upgraded at the same time ($\chi^2 = 8.46$, df=4, $p<0.1$). This finding could be accounting for the informal support users received from colleagues from the same department or for a more formal support from the IT installation crew as they spend more time in a particular department.

4.4.2 Type of upgrade

Responses were categorised in four groups according to the severity of the change (see Table 4-1, above). In this case, a change of version (i.e. Word 6 to Word 7 - category 1) represents a more subtle change than changing software and hardware altogether (i.e. from Mosaic in a Mac, to Netscape running on a PC - category 4).

A significant negative correlation ($\chi^2 = 9.76$, df=2, $p<0.01$) was found between the severity of the change (or amount of change) and the problems caused by the transfer of skills from previous knowledge. In other words, the more exaggerated the change, the fewer transfer problems appeared to be.
Another consequence of experiencing a more visible change was the reduction of the compatibility problems experienced by subjects ($\chi^2=8.26$, df=2, p<0.025). Both these counter-intuitive findings are discussed in detail in 4.5-Discussion, page 89.

4.4.3 Experience

No significant correlation was found between user's experience with computers and all other variables. More experienced users (users that had used IT tools for more than 10 years and therefore underwent many upgrades) reported similar experiences with recent upgrades as those with little IT experience.

4.4.4 User Satisfaction

User satisfaction levels increased when they could work faster with the new tool ($\chi^2=119.49$, p<0.005, df=81), the tool was easier to use ($\chi^2=85.37$, p<0.5, df=81) and when they make less errors ($\chi^2=52.29$, p<0.1, df=42).

4.4.5 Lack of support

Users that experienced little or no support at all, reported having more Transfer of Skills problems ($\chi^2=,$, p<0.01, n=43) than those that were formally supported through training.
4.5 Discussion

The survey and interviews described in this chapter helped identify that most users experience some type of problem when upgrading their software, problems that can be grouped in the following categories: Transfer problems, Compatibility problems, Response Time problems and Configuration problems (Table 4-2).

<table>
<thead>
<tr>
<th>Transfer</th>
<th>Configuration</th>
<th>Response time</th>
<th>Compatibility</th>
<th>Lack of Support</th>
<th>n^4</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 (51.2%)</td>
<td>2 (4.65%)</td>
<td>9 (20.9%)</td>
<td>20 (47%)</td>
<td>31 (72.7%)</td>
<td>43</td>
</tr>
<tr>
<td>20 (60.6%)</td>
<td>2 (6.06%)</td>
<td>8 (24.2%)</td>
<td>19 (58%)</td>
<td>24 (72.1%)</td>
<td>33</td>
</tr>
<tr>
<td>2 (20%)</td>
<td>0 (0%)</td>
<td>1 (10%)</td>
<td>1 (10%)</td>
<td>7 (70%)</td>
<td>10</td>
</tr>
</tbody>
</table>

Table 4-2, A comparison of findings between total and restricted sample

Of these four categories, the last three relate to technology and only the first one, to people. Although the technical findings are important and need to be considered to minimise the human costs of software upgrades in organisations, we will put them aside for now and return to them by the end of this thesis. The issue of Transfer will be the main focus of this thesis from now on, for two different reasons:

1. It is the category in which most of the problems where found
2. It is the only category that relates directly to people, falling under the umbrella of human factors

^4 n=43 represents the totality of respondents, including the 10 that reported "bigger" changes; n=33 reflects the restricted definition of upgrades found in 2.3-Scope, page 13.
4.5.1 Transfer of Skills

Subjects reported problems when they tried to use existing skills with the new version of the software. In many cases, the new version would not provide the same short-cut key for a particular function as the users knew it. In other cases, different dialogue windows replaced old ones, functions were moved to different locations within the menus and procedures were modified in order to add new functionality or maintain consistency. Subjects experienced frustration, as previous proficiency levels decayed when using the new tool. The previous investment in the development of skills for a particular tool proved, in many cases, to be of little or no use at all.

It is worth distinguishing between two situations when transfer of skills occurs. One, in which skills learned through training get “transferred” into using the tool, in the workplace. In this situation the learner applies skills that acquired during a training session.

Another situation is when subjects apply skills that, although originally learned through training, have been practised for a long time while using a version of a particular software tool. In this case, skills that already exist are applied using a new version of the tool.

As was concluded in Chapter 3, the organisational survey of upgrades, most organisations do not support “simple” upgrades with training. It is the belief that users will easily transfer existing sets of skills to the new version. However, it seems to be in this situation when users experience most of the Transfer problems reported in this study.
4.5.1.1 Transfer of Skills and the visibility of the upgrade

It would be expected that transfer problems occur when the new environment is different to the old one, and subjects can not apply existing skills. The study showed the opposite: Users who have been subjected to less serious changes appear to have experienced more problems than those that were exposed to a "bigger" upgrade.

A possible explanation for this counter-intuitive effect is that when changes are more visible (i.e. change of hardware), users get more attention from implementers, and training is provided.

The study showed that only 27,3% of the users were formally supported through training courses in the upgrade of their software. This indicates that the majority of respondents had to use alternative methods (i.e. colleagues, manuals, etc). Situations in which software was installed in a different technical platform, or a new product was installed showed higher levels of support from implementers.

Another possible explanation for this effect is that when users are more aware of the change, they recruit different cognitive resources, using a different strategy in the interaction with the new tool. Thus, they pay more attention to the transfer of skills.

When the change is subtle, users experience more problems, as they "automatically" try to apply old skills, learned from previous versions of the tool. This is not the case for users that experienced a bigger change.
In conclusion, when the change was more obvious (either because the users were supported through specific actions or because the tools were different enough) users experienced fewer problems transferring existing skills.

4.5.1.2 Transfer of Skills and Stress levels

In the interviews, it was found that transfer problems are exaggerated in two situations:

- When working under stress. A subject reported that her fingers would automatically jump onto the F10 key (instead of Alt+S) when wanting to save a particular report. This was exaggerated on Friday afternoons, the deadline for that particular report. She needed to frequently save her document, for fear of losing information, but would get frustrated when, instead of saving, the spell-checking window appeared.

- When being relaxed, in "automatic pilot": Some subjects reported an increase in the number of problems as they relaxed, thinking they are the same tools. Some of them mentioned that in order to avoid errors they have to be in "non-automatic pilot".
These findings, in which Transfer problems appear to be exaggerated when users experience little or too much stress, reinforce Yerkes and Dodson ‘inverted U-shaped curve’ (Yerkes and Dodson, 1908; Locke, 1976). The Yerkes-Dodson Law, developed around the eye-witness testimony field, states that performance is related to stress in a curvilinear fashion. At low stress levels, low arousal, subjects perform poorly. As stress increases, performance reaches its peak, diminishing again under higher levels of stress.

4.5.2 Preliminary Hypothesis

As a result of the survey we can now offer a tentative hypothesis about the circumstances in which people experience skill transfer problems. It appears to be a function of the visibility of that change, either characteristic of the software itself, or triggered by implementation actions. Before exploring this preliminary hypothesis, it is important that we examine what is already known about the issue of Transfer of Skills, particularly in the IT arena. That is the purpose of the next chapter.
4.6 Chapter Conclusions

The study described in this chapter showed that most users responding to the questionnaire\textsuperscript{5} experienced software upgrades as a far from transparent processes and in most cases very disruptive ones. Users reported little participation on the decision for upgrades and the feeling of not being supported during their implementation.

Some of the subjects studied returned to the previous version of the tool explaining that they “did not have time to learn the new one”. Some others reported having to spend long hours “figuring it out”, abandoning certain functionality and customisations and even feeling inappropriate for the job. A job that they had already mastered and it now has changed.

Most respondents experienced problems when upgrading software. The most frequent problem occurs when transferring existing skills to the new tools. Before testing the counter-intuitive preliminary hypothesis, in which transfer problems decrease as a function of their visibility, the next chapter will perform a review of the literature in the area of transfer, attempting to create a model that could explain the effects software upgrades have on end-users.

\textsuperscript{5} As it will be further described in Chapter 8, a weakness of this study is that, at the moment the University holds no information on the total number of clerical IT users nor the type and number of software upgrades. Then, the sample obtained could have been a self selected sample, influencing the findings reported.
5. Towards a Model of Transfer of Skills for Upgrades

"What you do to the mind knows its place; it never spreads. You train what you train"
Stratton, 1922

5.1 Chapter Summary

Chapter 4 discussed the effect that upgrades of software tools have on individuals working in an organisational context. From the different types of problems found, the issue of skills transfer was the one most frequently reported.

This chapter will explore the theories behind skills transfer, particularly those related to the interference effect that previous skills have over the acquisition of new ones.

At some points in the review we will re-examine the data from Chapter 4 to test whether the theoretical propositions are helping to understand the phenomena.
5.2 Introduction

Many researchers and professionals in psychology would agree that the problem of transfer is at the core of cognitive psychology, particularly at the centre of learning. Singley and Anderson (1989) stated that "the problem of transfer if perhaps the fundamental educational question".

When subjects were asked about their problems with upgrades in the previous chapter, they complained about the transfer of skills by mentioning that old skills were preventing them from easily learning and applying new ones. So, why are users having problems "upgrading" their skills to newer versions of software they already were familiar with?

In order to answer this question, we need to review what the cognitive psychology field knows about transfer. It is important to understand the process through which humans learn, associate and forget skills, as they continue to peregrinate along the survival road.

Once these theories are reviewed, we can identify possible explanations of the problems of transfer as reported in the study described in the previous chapter.

5.3 Transfer: A Definition

There is no one formal definition of transfer. In this thesis we will use Ellis’s definition: “Transfer is the application of knowledge and skills acquired in one setting to other situations" (Ellis, 1965; Singley and Anderson, 1989).
Many researchers use the names transfer of learning, transfer of training or skills transfer almost interchangeably. For the purposes of this thesis, transfer of learning and transfer of training will refer to the situations in which skills are transferred between a training event and the application of those skills.

Most studies of transfer concentrate in this area, looking at how much learning occurs under different training conditions, testing those skills as they are applied. For example, initial research on transfer asked subjects to learn items from pair associate lists, testing the amount of transfer (or interference) at later stages (i.e. McGeoch and McDonald's studies of the role of interference in 1931).

We will consider skill transfer in this thesis as a different category, as the transfer between current skills and the application of those skills to a new situation.

5.4 A Classification Of Transfer

Authors have categorised transfer according to its direction (Ellis, 1965; Hall, 1971) and dimension (Burton & Magliaro, 1987; Gagne, 1966). The following section will explore these categorisations, before going into an analysis of the theories behind transfer.

5.4.1 Direction

The direction of transfer is observed through the impact that original learning has over subsequent learning or vice versa.
5.4.1.1 Positive transfer

Positive transfer occurs when the knowledge from a previous task (a software tool, in this case) helps in the performance of a subsequent one. For example, when presented with a new piece of software, a user will know that "double-clicking" on the icon in the corner of a window, will close that window.

5.4.1.2 Negative transfer

This is the case in which previous knowledge has a detrimental effect over the acquisition (and usage) of new knowledge. This is a very important category for this thesis, as it can explain the problems users have when using new software. The case of negative transfer, also called interference, will be studied in more detail later in this chapter (see 5.6-Interference Theory, on page 105).

5.4.1.3 Non-existent transfer

In this case, previous learning has no observable effect on the performance of a new task. It neither interferes with the learning of the novel situation nor supports it.

Consider a case in which transfer is non-existent, where nothing that has been learned before applies or interferes with the new situation. This can only be described conceptually, as it would be very difficult (almost impossible) to assert that nothing that a person knows is being re-applied in a novel situation.

5.4.2 Dimension

The classification near-distant relates to the environment in which the skills are learned originally and applied later. The specific-general categories relate to the specificity of the content being transferred.
5.4.2.1 Near transfer

This type of transfer can be observed when the new context has similar features to those in which proficiency was achieved. Therefore we can say that there is an almost "direct application" of the learned skills to the new situation.

5.4.2.2 Distant transfer

Knowledge is transferred even when environments are different. An example of a positive distant transfer can be seen when people transfer knowledge from the real world to their understanding of software. This explains the success that GUI technologies (Graphical User Interface) have had. Users could use well learned skills of "pressing on a doorbell button" to clicking on a pushbutton on a computer screen.

5.4.2.3 Specific transfer

In this category, well-defined particular skills or concepts are transferred to a novel situation. For example, this is the situation in which a particular shortcut key to operate the <save> command in a wordprocessor is learned (i.e. Ctrl+S) in classroom training and later applied as-is in the real world.

5.4.2.4 General transfer

Different to the specific transfer category, in this case strategies or heuristics are transferred from one situation to another. To continue with the same example, it is the concept that documents can be saved, and that there are shortcuts that can be used to operate the <save> command, what is learned and later applied.
The differentiation between near and distant, specific and general should not be considered a binary categorisation of the types of transfer. Research in transfer of computer skills has looked into all these dimensions, finding it difficult to isolate one from another. Transfer of computing skills in the real world is a polymorphic event that would occur in more than a single dimension.

5.5 Theories of Transfer: Two Lines of Thought?

The subject of transfer has been widely researched, particularly over the last century. Many theories attempted explanations about how humans transfer skills to novel situations (Appendix 3 presents an account of many of this theories, grouped by underlying principles). In spite of having a large number of theories, details on how transfer occurs are today still somewhat obscure.

A historic view of the research performed in this area reveals two very different directions. Their main difference is whether skills are specific to particular situations in which common elements are found, or skill development which is about exercising the mind, like a muscle.

It was at the beginning of this century that Thorndike and Woodworth (Thorndike and Woodworth, 1901) demonstrated that there was always an element of commonality when transfer of skills was observed. Their theory of identical elements became a bedrock for significant further research although theories that supported transfer as non-specific prevailed until today. A review of their theory and a counter-argument follows.
5.5.1 Theory of Identical Elements

In their work Thorndike and Woodworth support the idea that mental functions are quite independent from one another and that the development of one type of cognitive skill will only be limited to that particular skill. They argued that “it is misleading to speak of sense discrimination, attention discrimination, attention, memory, observation, accuracy, quickness, etc., as multitudinous separate individual functions are referred to by any one of these words. These functions may have little in common” (p.p. 249).

They believed that skills were not of the general type but very specific to the material (and context) where the skill has been learned. The only kind of transfer, they reported, is the one in which the same skill is performed in a slightly different environment.

As an example of their work, Thorndike and Woodworth describe an experiment in which they presented subjects with 125 pieces of paper cut in various shapes (i.e. rectangles, triangles, circles, irregular figures). Subjects were self trained in the mental function of estimating areas from 10 to 100 cm² using the rectangles. When sufficient proficiency was reached, they were tested using other figures. The subjects showed very poor transfer skills for estimating areas.
The Theory of Identical Elements asserted that as long as elements were identical in two situations, transfer could occur. As the "distance" between these two situations increased, the level of transfer would decay rapidly. Thorndike and Woodworth believed that developing a particular skill such as "attention to the meaning of words", would not develop the skill of "attention to the length of words" unless there was strong element of commonality between the two skills. Stratton, a supporter of this theory, in relation to transfer wrote: "What you do to the mind knows its place; it never spreads. You train what you train" (Stratton, 1922).

5.5.2 A critique of the theory of identical elements

Thorndike and Woodworth's theory of identical elements drew major criticism from some researchers in the same field (Orata, 1928; Guthrie, 1935; Cook, 1936; Allport, 1937). The major controversy was related to the loose definition of elements in their theory, which varied from great specificity to great generality.

In a counter-argument to Thorndike and Woodworth, Pedro Orata argued that the mechanistic interpretation that Thorndike proposed in which transfer occurs as a result of common elements was, in fact, not a definition of transfer, but a mere repetition of a learned skill.

Orata stated that "doing over again what we have done before in another situation is not transfer, since it is nonsense to say that training to add 2+3 is transferred to the ability to add 3+2" (p.p. 158). Deviating from the repetition of elements as a means to understand transfer, Orata suggested transfer to be "...the extension and application of meanings to new problems or situations in such that we can deal with them effectively" (pp 176).
5.5.3 A modern perspective on the theory of identical elements

The discussion of whether transfer is specific or general has been the leitmotif in the history of the theories of transfer. However, a modern perspective suggests that specific and general components can be found in transfer under certain conditions (Gagne, 1985) and that specific and general theories are "...separate components and dimensions of a multifaceted process" (Gray and Orasanu, 1987, pp 183).

Current models of transfer are becoming more holistic, as they include motivational and environmental variables as components that facilitate or impede the transfer of skills. For example, Baldwin and Ford (1988) propose a complex model in which transfer is not just seen as input and output factors but a series of transfer conditions. Their model accounts for not only the type of material being learned but variables such as student characteristics, instructional design, work environment, degree of original learning and retention, etc.

Their model could be represented by the following function:

\[ TR = f \left( SC, ID, WE, DL \right) \]

where

- \( TR \) = Transfer
- \( SC \) = Student Characteristics
- \( ID \) = Instructional Design
- \( WE \) = Work Environment
- \( DL \) = Degree of Original Learning
In this case, for transfer to occur, all variables should have a positive value higher than zero.

5.5.4 A re-examination of the findings from the survey

The findings from the survey described in the previous chapter (see 4-A Survey of Problems Caused by Software Upgrades, page 57) suggest that users experience problems as they transfer skills from previous versions of the software they use, to the upgraded ones. Furthermore, these problems are exaggerated as the change users experience becomes minimal. This counter-intuitive effect needs to be explained by the models of transfer reviewed.

The theories and models analysed do not propose clear explanations for the findings from the survey. Data shows negative transfer from the previous version of the software towards the acquisition of the new one; negative transfer that increases as the two versions become similar.

The Theory of Identical Elements would suggest that positive transfer occurs in the case of maximum similarity (i.e. the case of a simple upgrade). However, this theory does not account for negative transfer. That is the main problem here.

We will now divert from the general theories of transfer and explore the Interference Theory as it could provide a sensible explanation for the effects reported by subjects in Chapter 4.
5.6 Interference Theory

The classical view that memory is based on the association of ideas is the basis for the Interference theory. Born in the times of British philosophers Hobbes and Locke, in the seventeenth century, the theory that learning occurs when previously unrelated events are associated is still considered today a major direction for the cognitive sciences.

Even though research in this area has undergone a rapid decline over the last decade, contemporary researchers such as Alan Baddeley consider that "...it would be a great pity if the awareness of the very powerful effects of interference were to be lost simply because they tended to be associated with a theoretical approach that became regarded as outmoded and somewhat sterile" (Baddeley, 1993, pp 248).

The theory of Interference was originally used in an attempt to describe the effects of short-term forgetting (Keppel and Underwood, 1962; Loess, 1968), competing with trace decay theories (Peterson & Peterson, 1959; Brown, 1958). In the case of this thesis, a long-term memory perspective is more appropriate, as the skills users learn are practised for several months.

Interference can be defined as proactive or retroactive depending on its direction. This categorisation is described below.
5.6.1.1 Retroactive interference - RI

RI refers to the effects that the learning of new skills has over the recall of previous ones (Figure 5-1). For example, if a user had to use an older version of a word-processor after having used the newer one for a while, he or she would experience interference from new skills, having difficulties remembering old commands. Attempting to use newer commands on the old software would be expected.

5.6.1.2 Proactive Interference - PI

This is the case in which older skills affect the development of newer ones (Figure 5-1). This type of interference is the one that supports the data from the survey (Chapter 4, page 57). The results showed that users experience interference from previously learned versions while learning new versions of the software.
Data from the survey showed that *interference* increased as a function of the similarity between versions (see 4.4.2-Type of upgrade, page 87). Many authors have identified *interference* to be a function of the *similarity* between the material learnt and the new one (Underwood, 1964; Underwood and Postman, 1960; McGeoch, 1942) and not a function of the *type of material* (Underwood and Postman, 1960; Underwood, 1964) or of the level of initial learning (Slamecka & McElree, 1983; Baddeley, 1993).

The theories reviewed only explain part of the problem. It is true that there are high degrees of similarity between software versions, and therefore interference can be found when transferring skills to newer versions.

If increasing the similarity between versions result in increasing the interference, what happens when similarity is at its maximum, namely, when each day users use the software they already know? Their existing skills and those required to operate the software are identical, and therefore interference should reach its peak. However positive transfer can then be observed.

The formal theories of interference don't explain this effect, however there is a model of interference that does take similarity of elements into account and proposes an explanation for the findings: Osgood's "Similarity Paradox".
5.6.2 Osgood - The Similarity Paradox

Osgood (1949) argued that the well accepted principle that "the greater the similarity, the greater the interference" did not apply to all the spectrum of transfer, but to particular instances of it. Maximum similarity would not only not create interference, but would be the case of maximal facilitation of learning, as skills that are practised repetitively are learned better. Although each repetition of the skills would be a unique event in which the experience in slightly different each time, ordinary learning would occur.

In order to create a model that would explain the paradox, Charles Osgood (1949) performed an integration of the data from several studies conducted on the issues of transfer and retroaction. Although his focus was on retroactive interference, it will help explain proactive interference findings from this thesis. Similar to the work done by Gibson (1940), Osgood's integration of data included resources recently made available, that helped him complete a picture that Gibson had left inconclusive. However, Osgood in his analysis makes full use of Gibson's findings.

Osgood also based the new model on the Skaggs-Robinson Hypothesis, in which Robinson (1927) identified a different non-linear function between similarity and transfer.
The Robinson hypotheses is represented in Figure 5-2. The efficiency of recall (Y-axis) is plotted against similarity (X-axis). Facilitation is greatest in point A which indicates maximum similarity (i.e. identity); point B represents a moderate degree of similarity, with an effect of maximum interference and low facilitation; point C defines minimum similarity (neutrality), causing an increase on facilitation that does not reach original levels.

Robinson's definition of similarity was further developed by Osgood, by considering *stimulus* and *response* as the two main components of similarity. Maintaining the same stimulus and learning a different response would have higher interference effects than maintaining the response and changing the stimulus. Adding this new dimension to the model, Osgood proposed the "Transfer and Retroaction Surface" to represent the problem of transfer (see Figure 5-3).
Figure 5-3: The retroactive interference surface as described by Osgood

In this model transfer (Y-axis) is plotted as a surface across stimulus (Z-axis) and response (X-axis) similarities. Transfer reaches its positive zenith when both stimulus (S_i) and response (R_i) are identical (the case of ordinary learning). As the stimulus changes (S_s → S_n) and response remains identical (R_i), positive transfer diminishes until reaching the neutral stage. Transfer is therefore neutral up to the point in which, identical stimulus (S_i) is associated with a less identical response. As response drives away from similarity (R_o) transfer becomes negative, reaching its maximum in the case when identical stimulus (R_i) is linked to an antagonistic response (R_a).

This model in which there is a distinction between stimulus and response, and a non-linear relationship between similarity and interference (or transfer) seems to explain the findings from the survey.
5.7 A re-examination of the findings from the survey II

Although the GOMS model developed by Newell, Card and Moran (Card et al, 1983) is based upon a different conceptual framework than the stimulus-response theory, for our purposes we will substitute stimulus and response with components of GOMS.

GOMS identifies four components: Goals, Operators, Methods and Selection Rules. The Goals users have when attempting to perform a task on a computer could be placed along the Z-axis, as the stimulus in the Osgood model. These Goals would trigger some response, represented along the X-axis by the other components of the model: Operators, Method and Selection Rules (Figure 5-4). For example, the Goal of "underlining a word" would be the stimulus that trigger the Operators, Methods and Selection Rules required to underline that word in those particular circumstances.

Re-interpreting Osgood's model to explain the problem of transfer of skills in software upgrades, the following can be observed:

• When there is no change, neither in the Goals nor in the resolution methods for those Goals, a positive transfer could be expected (point A in the graph). This is the case in which users continue to use their software, without upgrading to new versions.

• When new functions are introduced (point B in the graph) there is a change in the Goal (stimulus) AND in the Methods (response). In this case there is neutral transfer without interference.
- The situation where maximum interference (negative transfer) can be observed is the one in which the Goal remains identical, and the Methods change (point C in the graph). Any of the problems found in the survey could function as an example here, for instance, changing the short-cut key or the location of a known function.

Figure 5-4, Using GOMS with Osgood's similarity paradox model

5.7.1 Implications

The previous discussion used a model of transfer to explain some of the findings from the survey in Chapter 4. However, the survey indicated that when users experienced a "more visible" upgrade, the effects of interference diminished. Osgood's model does not account for this issue.
There is a possible route of research, if we consider context in this equation. The role of context in recall has been widely accepted and used in HCI research (Mayes et al, 1988). If context can aid recall, we could assume that it could be reinforcing interference as well, in the case of transfer. By changing the context in which recall takes place, we could minimise the effects of interference (this is represented over Osgood’s model by the blue arrow in Figure 5-5). Adding context to Osgood’s model, might explain the findings from the survey.

Figure 5-5, The addition of context to Osgood's model of transfer
Context can be established in different ways, although the only contextual variable we could operate on in the scope of this thesis is the visual cues within the tool. Modifying visual cues could minimise the interference effects caused from already learned skills that apply to a previous version of a software package. In a way, this would imply making the versions less "similar" to each other. Before exploring this issue through an experimental setting, a hypothesis needs to be established.

### 5.8 Hypothesis

The hypothesis is then defined as follows:

"When upgrading to a new version of a software package, simple environmental visual cues reduce the interference effects produced by previous skills in the acquisition of new ones."

This suggests that altering some of the environmental variables could help in minimising the negative transfer effects when upgrading software tools. However, before testing this hypothesis, we should consider that these alterations go against the consistency principle.
5.9 Change contextual variables or maintain consistency

The decision to operate on the contextual variable and not other variables for this study is based on the real-life application of the study. In most cases, with off-the-shelf packages, altering the design (i.e. modifying the short cut keys so they become consistent with the previous version) would not prove feasible (i.e. expensive and difficult to maintain). However, there is some flexibility in modifying the visual context of use of a new version of the software.

5.9.1 The Importance of consistency

Today's organisations are most concerned with maintaining consistency in the design of their software tools (European Commission, 1996). Standardisation groups exist inside organisations and there are several national and international bodies that look after standards for software development (i.e. ISO).

Consistency is undoubtedly imperative in design as a means of guaranteeing positive transfer of skills. It is sensible to say that workers that have to learn new skills want to re-use previous skills as much as possible, minimising the needs for training or support.

Consistency is a must when designing an application: Controls should be placed consistently around the interface, dialogues should behave in similar manner (even when dealing with different data) and the aesthetic appearance of the product should seamlessly bundle the functionality together. In fact, large quantities of money are spent in the creation, distribution and maintenance of standards for the creation of software.
The role of the standards group is defined early in Information Technology projects and their well documented (in most cases) sets of rules and guidelines are distributed and even enforced across design teams.

A good example of the benefits from interface consistency can be experienced when using most of the off-the-shelf Microsoft products. Microsoft's design strategy is to maintain consistency of the main menus of their products. In this case, when searching for a "standard" command, users will be able to transfer skills from previous experiences with other products.

For instance, most users with minimal experience on popular software will know how to open a file, regardless of the type of file being opened (e.g. document, spreadsheet, graph, etc.). This type of consistency supports a more transparent interaction in many cases.

5.9.2 The problem with consistency

The importance of consistent interfaces and its positive impact on skill transfer has been widely discussed in the HCI literature (Polson, 1988; Polson et al., 1987; Karat et al., 1986). There is also a counter opinion, that argues that consistency prevents products from being tailored to the real needs of the users (Grudin, 1989).
However, none of these studies looked at the transfer of skills that exist between versions of the same off-the-shelf software tool, where changes are normally disguised in order to maintain the consistency of the product. As discussed in Chapters 3 and 4, when people learn skills that are somewhat similar to the existing ones, difficulties appear. These difficulties seem to be related to the perceived "distance" or similarity between old and new versions of the software.

Therefore, the questions to be asked here are twofold:

- Should changes in a product be made so they unnoticeably integrate into the existing tool, maintaining consistency and therefore maximising positive transfer of skills?

- Or could changes be exaggerated through visual cues so as to minimise the interference effects that previous skills might have over the acquisition and usage of new ones?

Answering these questions is the purpose of the next chapter, investigating the impact that context has when transferring skills between two similar versions of software products.
5.10 Chapter Summary

This chapter explored different transfer theories, particularly those that explain the interference effects found in Chapter 4. The "similarity paradox" model created by Charles Osgood was combined with the GOMS model to propose an alternative to minimise the interference effects on software upgrades: the use of visual context. An experiment to test this hypothesis is described in the next chapter.
6. Upgrading to Word 2000: The role of context similarity in positive transfer facilitation

"I used to be quite good with the previous version"

A surveyed user

6.1 Chapter Summary

In a previous chapter we explored the interference effects that occur when transferring skills between versions of the same software. This chapter will further investigate the role that context similarity has over those effects.

This chapter describes an experiment to study the effects that similarity of visual context has over people's learning and remembering of changes in the software they use. Paradoxically, this study shows how maintaining consistency when upgrading software tools could be the wrong strategy for the innovation of technology.
6.2 Method

6.2.1 Experiment design

The experiment described in this chapter simulates an upgrade of software. In order to reproduce the effects that would normally occur in a real environment, it was important to first understand the "typical" changes users find when they are exposed to a new version of software. Creating the same type and amount of change on a new (hypothetical) version of the software would allow us to test the impact that our intervention (visual cues) could have on the real context.

We needed a method that would help us represent knowledge (or skills)\(^6\) required to operate each version of the software. The amount of identical skills found in both versions could be a good indicator of the similarity and differences between them and therefore illustrate the creation of the new version of the software for the experiment.

6.2.2 Different methods explored

There are several methods used in the HCI arena created in an attempt to qualify and even quantify the knowledge that is required to successfully operate a software tool. Some of these methods were reviewed in order to identify the one(s) that better serve the purposes of this study.

---

\(^6\) "Knowledge" and "skills" are used as interchangeable terms in this thesis.
Cognitive modelling tools provide an explicit analysis of the knowledge people need in order to perform a task. In some cases they allow the prediction of learning times and amount of transfer from previous skills. Most recently, these methods have attempted to account for error production. Early versions were criticised because they did not allow for human error (Olson, J. and Olson, G., 1990).

Several methods were analysed for this study: GOMS (Card et al, 1983), ERMIA (Green and Benyon, 1995), Cognitive Complexity Model (Bovair, Kieras and Poitson, 1990), Interacting Subsystems Model (Barnard, 1987), ACT (Anderson, 1983), Task Action Grammar (Green and Payne, 1984; Green and Payne, 1986) and the Keystroke-Level Model (Card, Moran and Newel, 1980).

Most methods were discarded because of being cumbersome and time consuming for using in a real environment. It would prove unfeasible to perform any of the methods on a full-size off-the-shelf product, as these products contain a very large amount of functions and normally more than one way to access each of those functions.

A sense for this complexity can be demonstrated through the following two examples. Both cases only describe one of the eight different methods\(^7\) that exist for opening an existing document\(^8\) using Word for Windows (Table 6-1).

---

\(^7\) This list of methods was obtained through interviews with users. Some of them reported not knowing some of the methods, showing that they not always choose the shortest, most economical option.

\(^8\) This function was randomly chosen to illustrate the point in the section.
Table 6-1, Different methods for opening an existing document using Word for Windows

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Retrieving a document from the list of the last four edited documents</td>
</tr>
<tr>
<td>2.</td>
<td>Retrieving a document that is already opened in another window</td>
</tr>
<tr>
<td>3.</td>
<td>Opening a document by sequentially verifying the contents of all documents</td>
</tr>
<tr>
<td>4.</td>
<td>Retrieving a document by using its location and name</td>
</tr>
<tr>
<td>5.</td>
<td>Finding a document by using its location (sometimes used in conjunction with 3)</td>
</tr>
<tr>
<td>6.</td>
<td>Searching for a document using its name</td>
</tr>
<tr>
<td>7.</td>
<td>Searching for a document using part of its content</td>
</tr>
<tr>
<td>8.</td>
<td>Searching for a document using date, size, etc.</td>
</tr>
</tbody>
</table>

In the first case, ERMIA was used to represent how to open an existing document in Word for Windows, using only the fourth method from the table above. The researcher spent approximately 5 hours to perform this task (Figure 6-1).
Another example of how complex the use of formal methods could be, was observed when a GOMS approach was used to describe the same function. The results follow:

**GOAL: Retrieve document (knowing location and name of document)**

This high-level goal can be broken into smaller tasks or goals:

1. Accomplish goal of initiating the OPEN command.
2. Accomplish goal of identifying document.
3. Accomplish goal of selecting document.

Each of these sub-goals can also be broken further:
Method to accomplish goal of initiating the OPEN command

1. Recall command name OPEN... and retrieve from LTM (Long Term Memory) the menu name for it FILE, and retain the menu name FILE.
2. Recall the menu name FILE, and move cursor to it on Menu Bar.
3. Press mouse button down.
4. Recall command name OPEN..., and move cursor to it.
5. Recall command name OPEN..., and verify it is selected.
6. Release mouse button.
7. Forget menu name FILE, forget command name OPEN..., and report goal accomplished.

Method to accomplish the goal of identifying document

1. Decide: If document is on screen, then report goal accomplished.
2. Use scroll bar to advance text.
3. Decide: If no more documents in directory, accomplish goal of selecting next directory (note: for simplicity purposes this study will not develop this branch of the GOMS analysis further)
4. Decide: If no more documents in directory and no more directories in list, report document not found (note as above)
5. Goto 1.

Method to accomplish the goal of selecting document

1. Determine position of name.
2. Move cursor to any part of the name.
3. Double-click mouse button (note: this action can also be performed through other methods, but this one was chosen for being a popular one)
4. Report goal accomplished.
And these goals could be further broken into sub-goals.

<table>
<thead>
<tr>
<th>GOAL: Retrieve document (knowing location and name of document)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Accomplish goal of initiating the OPEN command.</td>
</tr>
<tr>
<td>2. Accomplish goal of identifying document.</td>
</tr>
<tr>
<td>3. Accomplish goal of selecting document.</td>
</tr>
</tbody>
</table>

Each of these sub-goals can also be broken further:

<table>
<thead>
<tr>
<th>Method to accomplish goal of initiating the OPEN command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Recall command name OPEN... and retrieve from LTM (Long Term Memory) the menu name for it FILE, and retain the menu name FILE.</td>
</tr>
<tr>
<td>2. Recall the menu name FILE, and move cursor to it on Menu Bar.</td>
</tr>
<tr>
<td>3. Press mouse button down.</td>
</tr>
<tr>
<td>4. Recall command name OPEN..., and move cursor to it.</td>
</tr>
<tr>
<td>5. Recall command name OPEN..., and verify it is selected.</td>
</tr>
<tr>
<td>6. Release mouse button.</td>
</tr>
<tr>
<td>7. Forget menu name FILE, forget command name OPEN..., and report goal accomplished.</td>
</tr>
</tbody>
</table>
None of the methods would prove an efficient tool to measure similarity between versions of the same software. It would not be cost effective to perform any formal analysis on the totality of a product.
However, a reduced version of GOMS was used for this study, to assist a visual inspection of the products. Using only the Goal and Method components of the GOMS methodology (approach introduced in Chapter 5) proved to be a feasible way to understand the changes in the new version of the software. In brief, the method used consisted in identifying major goals (i.e. opening a document, changing the size of the page, etc.) and visually inspecting the method needed in each version of the software. Although not as thorough as a complete GOMS analysis would have been, this approach proved to be a feasible way of identifying changes. There is a structured component to it (Goal) followed by a subjective analysis (Method).

The output of the analysis using the previous two commercial versions of Word for Windows (6 and 7) indicated that the most common changes were the re-location of options of the menu (the windows used for the visual inspection of the methods used for different goals are presented in Appendix 4).

6.2.3 Subjects

Based on the need for a set of subjects with similar pre-existing skills, forty-eight consultants employed by Andersen Consulting and Perot Systems Europe were chosen for this experiment. The main reason for choosing this user group is that is a very homogeneous population, both in terms of background and type of usage of their computing tools. This would minimise the variability among subjects, a known problem when performing experiments in the real world.
The subjects age ranged between 23 and 34 and they were all Change Management consultants. Many countries were represented in the sample, however not much variation was found amongst them.

The experiments were conducted in their work environment. Such a context-rich environment had both advantages and disadvantages. First, it provided the realism and richness of the real environment, serving the need for a true observation of the problem. Second, it required a much larger number of tests to finally obtain forty-eight valid ones.

It is important to mention here that the runs of the experiment that were discarded were due to uncontrolled disruptions of the test with activities that are part of their normal work. The most common reasons for having to discard a subject from the experiment were receiving a phone call or being interrupted by a colleague.

As their performance was automatically timed and they were not observed while performing the tasks, it was necessary to perform a short interview with each subject after the experiment to judge the validity of the test (e.g. "Were you interrupted during the performance of these tasks?").

Subjects were randomly divided in two groups: Experimental and Control, and were only exposed to one of two conditions. Subjects would only perform the experiment once.
6.2.4 Tool and Tasks

Two simulated upgrade scenarios were created for the experiment. The first scenario presented the experimental group with a new version of the word-processing tool named Word 2000, while the control group was exposed to version 8 of the same software.

But why Word for Windows? The reasons for choosing this word-processing package for this experiment are several:

- **It is widely used:** Word for Windows is the most popular word-processing tool these days and the standard in most organisations, even under Macintosh platforms.

- **The tool has gone through many versions already:** As presented in the introduction of this thesis, Word has undergone almost an upgrade per year, since its creation in 1990. By revising the type of changes made in previous versions of the tool, it was possible to create a credible “new version”.

- **It was possible to simulate an upgrade using its own programming language:** Word for Windows utilises Word Basic, a programming language based on macros that allowed to customise the visual appearance of the tool, alter the menu contents and log the actions performed by the users.
• It allowed the creation of realistic tasks for the tests scenarios: Word-processing editing tasks are very common among computer users, and widely used in psychological experimentation (for an example see Carroll, J., 1982).

To better describe the experiment we should address two areas: First, the tasks that had to be performed and second, the external appearance of the tool. The only differences between the two versions (8 and 2000) could be observed in the external appearance of the interfaces. The tasks subjects had to perform were identical in both cases.

Users were guided by the tool, which through a series of macros displayed instructions on the tasks users needed to perform. These macros logged the user's sequence of commands and time stamped each individual task (an example of the internal log file is shown in Appendix 4). Macros were automatically triggered as tasks got completed.

The experiment comprised three sequential tasks: First, a welcome box presented the experiment and gave general directions (Screen 6-1 - Welcome window for Word 2000) on the experiment. A document opened automatically, over which the tasks were going to be performed. This document was, at the same time, a reinforcement for the type of changes the new version contained.
6.2.4.1 First task

Consecutively, directions for the first task were displayed. In this case, the objective was to change the size of the page from letter to A4. In the current version of the tool (Word 7) this task is performed through the path <File><Page Setup...>. This would display a dialogue box with several tabs, one of them being <Page Size>. Once this one is selected, there is a combo-box displaying the possible page sizes, one of them being A4.
For the experiment, the location of the <Page Setup...> function was changed. It was placed under the <Format> option of the main menu (Screen 6-3). As described earlier, this type of change is a very common one with software upgrades. The reason behind a change like this could be that the makers of the software saw that that function was, in fact, a formatting function and not a file command.

6.2.4.2 Second task

After this first task was performed, a second window appeared with directions for the next task. Subjects were asked to italicise the first three occurrences of “Word 2000” (or “Word 8”, depending on the group). All italicising functions were identical to those in the current version of the software. The reason for this task was to act solely as a distracter between tasks one and three.

6.2.4.3 Third task

Once task 2 was completed, a third window appeared, displaying instructions for task three. Now, the subjects were asked to set the left margins of the page to two centimetres — a function that is performed using the same option they used for the first task, <Page Setup...>. The reason for this part of the experiment was to observe time to recall for the location learned during task 1.
Screen 6-2 - Beginning of task 1

Screen 6-3 - User looks for Page Setup under the File menu
Screen 6-4 - Finally the user finds the option under the Format menu

Screen 6-5 - The standard Page Setup dialogue is displayed
6.2.4.4 Visual cues

In terms of external appearance there was quite a difference between versions. Word 8 was identical to version 7.0 (the one subjects currently use) and Word 2000 was significantly different: Background and font colours were altered, new icons were added and sound and animation was used when the tool was loaded (see Screen 6-6 and Screen 6-7). However, the menu structure and the behaviour of the functions involved in the experimental tasks were identical in both experimental and control conditions.

Screen 6-6, Word 2000 version

Screen 6-7, Word 8 version
6.2.5 Pilot study

A pilot study was conducted with three users taken from the same user population described above.

As mentioned before, both "upgrades" were presented as real upgrades and not as simulations for an experiment. The fact that they were created based on typical changes, made them realistically look like a possible upgrade.

6.2.6 Procedure

Experimental conditions and control conditions were treated in a similar way, varying only in the tool to which subjects were exposed.

Users were briefed on the experiment and a simple identification of their profile was performed, confirming that they used Word 7 as their current word-processing tool. Users were then introduced to the upgraded tool (either Word 8 or Word 2000) and taken to the first automatic task window. From that moment they were encouraged to read the instructions and perform accordingly.

Although the experimenter was not physically with the users during the experiment, subjects were told how to reach him in case they got problems with the tool or the tasks they were to perform. No subjects abandoned the experiment as a result of need for help from the experimenter.
6.2.7 Questionnaire

After the experiment, subjects were asked to fill out a questionnaire (see Appendix 4) about their experience with the new version. The questionnaire was designed to elicit their immediate subjective opinion in regards to the upgrade, as well as some user profile information (i.e. time with previous version, depth of usage of previous version, number of previous upgrades, hours per week spent on the tool, level of automation, etc.).

This questionnaire was first piloted with three users and corrected for readability and interpretation. The questions or statements were placed in no particular order, collecting responses on a six-point Likert scale.

As can be seen in the table below (see Table 6-2), there are some negative\(^9\) statements embedded in the questionnaire. There were two reasons for having this type of statements in the questionnaire: To allow a check on the validity of the answers within each subject and to encourage subjects to read the statements carefully before placing a mark in the scale, preventing “automatic” responses.

---

\(^9\) Agreeing with this statement would have meant a negative opinion about the change or about the new version.
<table>
<thead>
<tr>
<th>Type</th>
<th>Statement presented to subjects in the survey</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceived learnability</td>
<td>I feel comfortable that I will soon master this new version of Word</td>
</tr>
<tr>
<td></td>
<td>I would like to spend more time learning this new version</td>
</tr>
<tr>
<td></td>
<td>Other people will have no problems using this new version</td>
</tr>
<tr>
<td>Perceived similarity</td>
<td>This new version and the one I’m used to are almost identical</td>
</tr>
<tr>
<td></td>
<td>I see the reasons behind the upgrade</td>
</tr>
<tr>
<td>Readiness for change</td>
<td>I see the need to upgrade to the new version</td>
</tr>
<tr>
<td></td>
<td>I am very confident with the version of Word I normally use</td>
</tr>
<tr>
<td></td>
<td>I would welcome a new version of Word</td>
</tr>
<tr>
<td></td>
<td>This new version will help me do my work better</td>
</tr>
<tr>
<td></td>
<td>Changing to this new version will be transparent for me</td>
</tr>
<tr>
<td></td>
<td>The rest of the organisation will welcome this new version</td>
</tr>
<tr>
<td>Perceived aesthetics</td>
<td>The new version looks very good</td>
</tr>
</tbody>
</table>

Table 6-2: Questions used in the questionnaire

6.2.8 Interviews

Once the subjects finished with both the experiment and the questionnaire, a short interview was conducted. As the experiments were performed in the subject's work environment and their performance was automatically timed and because they were not being observed while performing the tasks, it was necessary to judge the validity of the test. In those few events in which subjects were interrupted from the experiment (i.e. telephone calls), the data from that run was discarded.
6.3 Results and Analysis

The automatic logs generated by the macros were analysed and grouped with the subjective data provided by the users after the experiment. The results of the experiment are described below.

The graph below illustrates the findings obtained in the experiment (Figure 6-2). The X-axis represents each one of the three tasks performed, the Y-axis shows both Word 2000 (experimental condition) and Word 8 (control group) versions and finally the Z-axis shows the average time (in seconds) that it took subjects to successfully perform each task.
As expected, there was no significant difference on the performance times from task 2 between both groups (54 sec and 57 sec). This task had the objective of acting as a distracter, separating tasks 1 and 3. There was no difference in the visual appearance of the icon that had to be selected nor in the behaviour of the function.
The most important finding from this experiment can be observed by studying the average times for both control and experimental groups (Table 6-3). Consider the difference between task 1 and 3. On task 1, subjects had to re-learn the location of a function they were already familiar with, <Page Setup> (now under the Format menu) while task 3 was testing that learning\(^10\).

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AVG</td>
<td>STD</td>
<td>AVG</td>
</tr>
<tr>
<td>Control Group</td>
<td>137</td>
<td>130</td>
<td>54</td>
</tr>
<tr>
<td>Experimental</td>
<td>151</td>
<td>174</td>
<td>57</td>
</tr>
<tr>
<td>t-test (p)</td>
<td>0.74</td>
<td>0.86</td>
<td>.006</td>
</tr>
<tr>
<td>t_{00}(46)=\text{2.04}</td>
<td>0.34</td>
<td>0.53</td>
<td>-2.81</td>
</tr>
</tbody>
</table>

Table 6-3: Average times taken to perform tasks one to three and standard deviations

In task one, the control group, using the visually unmodified version (Word 8), took a mean time of 137 seconds to find the new location of the <Page Setup> function. The experimental group, however, took longer than that (151 seconds).

\(^{10}\) We will not attempt to tackle here the different mechanisms subjects used in order to find the function. The exploratory strategies were various but they do not add to the subject of this research as each group used a combination of the same strategies.
No mayor effect was expected for task 2, as the task and type of changes were identical. A t-test over the results from experimental and control condition showed the lowest distance between these two groups (t(p)=0.86).

In order to see the effects that this experimental manipulation had on subjects performance, we should see what happened with task three. The subjects exposed to the experimental version Word 2000 showed faster performance (20.02 sec) than those in the control group (45.97 sec).

These results were significant at the p=0.005 level, using the unpaired student t-test (two-tails; t=0.006, df=46).

6.3.1 Interpretation

The reduction in the time (task 3) as a function of the visual cues provided, could be interpreted in various ways:

1. Encoding Specificity. The principle researched by Tulving (1983) sustains that cues are stored together with the element being learned at the moment of encoding. In the experiment, giving the visual cues that belonged to a previous version of the software would have increased the interference levels.

By providing the subject with different cues (experimental condition), already learned skills were not being automatically retrieved.
2. **Depth of processing.** Subjects explored a "different" version of the tool spending more time manipulating options and trying to understand the changes during task 1. This deeper level of cognitive processing (instead of an automatic performance) supported their learning of the changes and increased performance for task 3.

3. **Learning strategy.** Other possible explanation for this finding is of the meta-cognitive type. Subjects, as being presented with a "new" tool, would switch automatically to "learning mode", increasing the time dedicated to the first task as they were building a new schemata or model for the tool. Consequently, this allowed for an increase of the performance times observed in task 3.

An increase in the number of alternatives or options could have increased the time required to cognitively process them. Subject's expectations, in the case of Word 2000, could have placed the "imaginary" addition of options as they were supposedly dealing with an important software upgrade.
6.3.2 The role of perceived change

As discussed earlier in this chapter, the main purpose for adding visual cues to the experimental condition was to increase the "distance" between current version of software and the upgrade. However, this "distance" is of subjective quality, as different people would perceive change in different ways.

At this stage, a different analysis of the data was performed exploring the role of perceived change on the time to perform each task. In this case, the entire sample (n=48) was re-distributed based on question number 2 from the questionnaire. The question that identified the subject's perception of the "distance" between versions was "this new version and the one I'm used to are almost identical".

The sample was then divided in two new groups, ignoring the original separation between experimental condition and control group. In this case, those subjects that perceived the change as being high (or large, or important - with answers 1, 2 and 3) were now grouped together into a new category. We will define this category with the Greek letter $\alpha$ (alpha, N=11). The subjects that rated the change as not being that important (answers 4, 5 and 6) will be considered group $\beta$ (beta, N=37).
This regrouping caused 17 subjects from the experimental condition to move to group β (smaller change) and only 4 subjects from the control group to move to group α (bigger change). This could indicate that either the changes built into Word2000 were not different enough to the current version of the tool or that people have a tendency to minimise perceived differences in order to apply existing skills. We will not attempt to answer this particular question in this thesis, however regrouping the data gave the thesis an important new direction.

If the times for the tasks are now considered (Figure 6-3), an interesting effect can be observed. The group that reported perceiving a bigger change (α), took significantly less time than the group perceiving little or no change at all (β) in the performance of tasks 1 (α=89 sec., β=160 sec., at t(9)=0.0457), and 2 (α=41 sec., β=60 sec., at t(9)=0.2702). Only a non-significant change was observed between the new groups for task 3 (α=30 sec., β=34 sec., at t(9)=0.707).
A possible explanation for this significant difference is that perceived similarity could have an important negative effect on the interference caused by previous knowledge. Subjects that saw the tool as being different from the one they were used to, probably approached the task in a different way, prioritising exploration skills over the automatic re-utilisation of old ones.

This approach prevented them from "blindly" operating the interface and getting lost when observing a different behaviour in response to their commands.
6.4 Chapter Conclusions

A hypothesis was proposed at the end of Chapter 4, creating a model in Chapter 5 that was eventually tested in the present chapter.

There is enough evidence from the experiment described in the present chapter to support the hypothesis, as the experimental condition showed less interference than the control one. In other words, we can conclude that visual cues do have a positive impact in the assimilation of a new version of the software.

However, the most important findings from this chapter suggest that perceived similarity between versions of software tools could have a stronger effect than the visual cues artificially built into the experiment, on the way in which people cognitively approach the change.

This last point requires us to revise the model proposed in Chapter 5. As discussed in section 5.7.1, page 112, the contextual variable was added to the Goal along the z-axis of the model. In this case, perception of similarity should be added to the contextual variable, as it was shown to have an even higher impact on the findings.

This thesis would not be complete without the translation of findings back to the applied world. The next chapter will present a suggested strategy to aid organisations in minimising the costs of software upgrades. The recommendations are based on the findings from all previous chapters.
7. An Organisational Strategy to Minimise the Human Costs of Upgrades

7.1 Chapter Summary

The present chapter proposes a strategy that organisations could apply to reduce some of the human costs of software upgrades.

7.2 Introduction

Previous chapters showed that the human costs of software upgrades in organisations are of significant importance (Chapter 4), findings that are not recognised by the organisational understanding of the costs of upgrades or by the strategies used to manage upgrades (Chapter 3).

The problem of transfer was explored (see Chapter 5) as it was the problem most frequently reported (Chapter 4); a strategy for minimising the problems of transfer was explored in Chapter 6.

At this point in the thesis both organisational (Chapter 3) and individual (Chapters 4, 5 and 6) threads come together in order to propose an overall strategy for the management of upgrades, with the intention of minimising the human costs and ultimately minimise the Total Cost of Ownership for off-the-shelf products (Chapter 1). The recommended strategy which follows is based on the findings obtained in the preceding chapters as well as the methods, theories and studies reviewed.
Research implications for this thesis will be discussed in the next chapter (Chapter 8). However, proposing a strategy at this point in this thesis intends to extrapolate some of the empirical findings into a method that organisations can apply for the minimisation of human costs of software upgrades. As a possibly first study in this direction, the recommendations in this chapter should not be considered definitive but starting points for further applied investigation.

Although software vendors have an important influence on the costs of software upgrades, the model will only consider what user organisations can do to minimise those costs, particularly what management, designers and implementers can do.

The model proposed (see Figure 7-1) aligns with the four stages usually found in most Information Technology methodologies: Planning, Design, Implementation and Support. The tasks represented in the model and the discussion that follows are specific to the human aspects of software upgrades; the model does is not intended to be a comprehensive methodology for systems installation. The tasks described here should be considered in addition to the many tasks found in IT methodologies.

The strategy and methods proposed in this chapter are targeted to:

1. *Decision makers*: Those that decide on the appropriateness and timing of an upgrade. Providing them with information on the vast array of direct and indirect drivers, and the effect of upgrades on people, could aid their decision making process. A checklist of the drivers is provided in Appendix V.
2. Designers: They are not the designers of the software itself (as it is an off-the-shelf package), but they design the configuration and appearance of the interface. They are named designers here because their job should not just be that of customisation of the tool. Instead, they would have to understand tasks and identify changes in goals and methods, so as to decide on the appropriate action to follow. Providing them with a quick job-aid, in which types of changes are followed by recommended actions could help them gain awareness on the complexity of the problem and support their decisions on when to hide or exaggerate a change in a new version of the software (this job-aid is provided in Appendix V).

3. Implementers: Communication, training and job-aids fall under the responsibilities of this particular group. They should work together with the designers of the upgrade (described above), supporting those changes that need to gain visibility. Presenting them with a checklist of possible support strategies could help them gain awareness on the complexity of the problem, at the same time as supporting in their job.

7.3 An initial big picture of the model

A brief statement of the overall rationale for the model is presented here, before each stage is examined in greater detail.

The method begins with the planning of the upgrade. Decisions about the timing of an upgrade should be made in an informed way, gathering data from internal and external drivers, as well as the continuous monitoring of the performance of the workers.
Once the upgrade is decided, designers should evaluate the current and future scenarios, deciding on whether the change should be hidden or exaggerated. A hidden upgrade will continue with the technical distribution and installation of the software, avoiding all disruptions. On the other hand, if designers decide to exaggerate the change, making the changes explicit through design. In that case, communication and training activities should be performed.

Finally, the help desk personnel are trained in the changes and resolution strategies to support workers with the new version. Information on performance problems is collected to inform management, assisting them in the decision efforts on the appropriateness of further upgrades.

The tasks presented in the model will now be described in detail.
Figure 7-1, A method for the minimisation of human costs of software upgrades
7.4 Planning an Upgrade

Deciding on the appropriateness of an upgrade is a key task that has to be performed. This thesis showed how most upgrades are triggered by indirect drivers in organisations. If we consider the human costs that are consequence of the upgrade of software, we would understand how important it is to make the right decision. Embarking on an upgrade should be justified with tangible benefits.

Upgrades can be disruptive for most users: Configurations can be lost, compatibility problems can appear and most critically, users can return to the feared learning curve. If the upgrade is founded on indirect drivers (i.e. "new software comes with the new computers") then human costs are added to the already expensive software and hardware costs, failing to counter-balance the cost-benefit equation.

7.4.1 Inform Decisions

The first stage in the proposed method is to inform the upgrade decision. Normally the strongest input decision makers have are the indirect drivers (discussed in section 3.5.3, page 41). Indirect drivers for upgrades not only will persist in the future but they might increase, as a consequence of the pressure from the IT market. Indirect drivers from within the organisation will continue to exist.

To support the decision on whether the upgrade is appropriate or not at a specific time in an organisation, it is important that information is available to decision makers. This information should include:
• Benchmarked data on the Total Cost of Ownership for a similar situation
• Measurable performance objectives
• Current levels of performance
• Direct and indirect drivers
• Cost-benefit models, etc.

Organisations such as Gartner Group have identified key aspects of successful migrations in organisations. Making the decision to upgrade (or not) is one of the main drivers for success (a detailed analysis on this issue can be found in Preparing for the Electronic Workplace: An OIS Migration Toolkit, Gartner Group, 1996).

Decision makers should recognise the hidden costs of software upgrades. Although these costs are yet to be quantified, findings indicate that they are quite significant and therefore should not be ignored.

7.4.2 The decision to upgrade
Once all information is analysed and benefits for the workers and for the organisation are identified, an upgrade can be justified (or not). If there are not enough tangible benefits upgrades should be postponed until needed.

7.4.2.1 Different upgrade strategies
There are essentially three different ways in which organisations upgrade software. These are illustrated in Figure 7-2.
Some organisations consider upgrades as a continuum process (red curve in Figure 7-2). In this case, upgrades efforts do not have a clear start or end, decisions are not made explicitly regarding the appropriateness of an upgrade. There is generally a team devoted to continuously upgrade workstations as soon as new versions are released to the market (Gartner Group, 1996a).

This is a somewhat disorganised approach, because while some groups are still using Version a, others could be already using Version b, causing incompatibilities of both data and skills. Upgrading in this way, as discussed in earlier chapters, will not only increase software and hardware costs, but will considerably increase the human costs of upgrades, resulting in poor performance.
A different approach is to clearly delimit each upgrade with a starting and ending point - the "big bang" approach (see blue line in Figure 7-2). In this case, all new versions are installed, but upgrades are implemented at the same time for the whole organisation. We have seen that subjects adapted to new versions faster when departments upgraded at the same time. (see 4.4.1-Time to master, page 87).

Big-bang implementations have two major advantages: First, they minimise software and skills compatibility problems within the organisation and second, they increase the visibility of the change, helping users recruit cognitive resources that will help them to better assimilate new skills. However, interference from previous skills could still affect performance in this case.

A more beneficial approach from the user perspective is when organisations decide to "skip" versions of a particular software, minimising the amount of interventions and disruptions to the workers (see green line in Figure 7-2).

From a skill development perspective, as demonstrated through this thesis, more clear and bigger changes are easier to deal with than minimal ones. Late upgrades have also the benefit of being more technically stable. This is a well known problem software vendors have caused by the urge for delivery of new versions. An example of this is Microsoft's Word 97, released as part of the Office 97 package during the beginning of 1997. Soon after the release, this version of Word had to be removed from the shelves of the stores as clients were experiencing major problems with the software.
7.5 Designing an upgrade

Current and new interfaces should be compared, evaluating the similarities between them. The method used in this thesis for the evaluation of the similarities between versions was based in the visual inspection of the interfaces and a part of the GOMS methodology (see Table 7-1).

<table>
<thead>
<tr>
<th>Goal (old  new)</th>
<th>Method</th>
<th>Perceived Similarity</th>
<th>Need for Intervention</th>
</tr>
</thead>
<tbody>
<tr>
<td>old</td>
<td>old</td>
<td>low</td>
<td>hide</td>
</tr>
<tr>
<td>old</td>
<td>old</td>
<td>high</td>
<td>none</td>
</tr>
<tr>
<td>old</td>
<td>new</td>
<td>low</td>
<td>none</td>
</tr>
<tr>
<td>old</td>
<td>new</td>
<td>high</td>
<td>exaggerate</td>
</tr>
<tr>
<td>new</td>
<td>old</td>
<td>low</td>
<td>exaggerate</td>
</tr>
<tr>
<td>new</td>
<td>old</td>
<td>high</td>
<td>exaggerate</td>
</tr>
<tr>
<td>new</td>
<td>new</td>
<td>low</td>
<td>none</td>
</tr>
<tr>
<td>new</td>
<td>new</td>
<td>high</td>
<td>none</td>
</tr>
</tbody>
</table>

Table 7-1, Similarity between versions

As described in section 6.2.2, page 120, performing a full-size GOMS analysis would be unfeasible. However, a shorter version in which only main Goals and their Methods are used, could be appropriate to perform in an organisational setting. The Goals over which the inspection will be performed, could be those that are of higher criticality for the particular user group, or for the business itself.
Table 7-1 represents eight situations designers might encounter when analysing similarity between two versions of the same software.

The first two columns result from a GOMS analysis of the versions, performed by the designers of the upgrade. Since analysing all functions of a software package is entirely unpractical, a realistic set of Goals has to be identified in order to perform the comparisons.

The third column lists the perceived similarity of versions from an end-user perspective (note that neither technical nor functional similarities are considered here). Users can rate their perception of similarity on a low-high scale, before the upgrade is implemented. As a result, recommended actions for design interventions appear on the last column of the table.

For example, the case in which same Goal is achieved using the same Operators and Methods is a case of maximum similarity or identity. However, if the perception of the change is high, design efforts should be made to hide all visual cues for the change. If users' perception of the change is low, it does not require any intervention from designers, as it will not represent a problem for the effective transfer of skills.

At the other extreme, if different Goals are achieved through the same Operators / Methods, exaggerating the change should be considered. Visual cues (i.e. changing the background colour of the application, or the font set) could help users identify new sets of skills, minimising the effects of interference.
A tool like the one suggested in Table 7-1 could help designers make the right decision on whether a particular upgrade should be exaggerated or camouflaged, facilitating users' assimilation of it.

7.6 Implementing an upgrade

It was observed in the studies for this thesis that most organisations only consider technical issues when implementing upgrades. This is still valid in the case a decision was made to hide the upgrade, in which there would be no engagement with the users. On the other hand, if the upgrade has to be made visible, the following tasks should be considered.

7.6.1 Communication

Early awareness of the launching of a new version of the tool can be beneficial for the individual management of skills. End-users will anticipate changes, increasing their potential in dealing with them. Communication messages should include timing and duration of the upgrade, reasons behind it, personal benefits (if any) and changes in the tool. The benefits of the latter should not be underestimated, as knowing what is different can help individuals cope with change in a better way.

7.6.2 Training

Advice here is based on the work of Charles Beard (1993) and Clare Pollock (1988), as they investigated in depth, the role of training in the development and transfer of skills.
Developing skills that “already exist” requires specific strategies. A standard course to teach the new version can frustrate users mainly because they already mastered the current version and will feel they are wasting their time reviewing existing knowledge. However, as observed in this thesis, ignoring the need to upgrade current skills could result in major costs for the users and therefore for the organisation.

7.6.2.1 Managing skills

An important first stage is to identify who requires what type of training in the organisation. Not all users could be transferring skills from a previous version of the tool. It is critical to have a means to allow the identification of skills that need “upgrading” and skills that need development.

Having a repository of skills could prove useful for the planning of training and other actions towards the development of skills in organisations. Identifying who has mastered a particular software version, would certainly support management decision on upgrades.

7.6.2.2 Levels of training

Clare Pollock (1988) analysed the transfer of skills from WordStar to WordPerfect®, using a combination of training methods. She experimented using low, high, related and unrelated training finding best results in the low-related group.
Her model is based on the concept of frames. The frames are organised in a hierarchical way, having low and high level frames. Low level frames can be constructed from low level training, although it is not clear if this type of training will enable learners develop high level frames through the low level ones.

High related training involved the generalisation of frames in order to produce a high-level frame. For example (from Pollock), "You delete a character to the left of the cursor in WordStar, by pressing the 'Del' key. You can delete a character to the left of the cursor on most word-processors by pressing a key or a combination of keys. In WordPerfect®, you delete a character to the left of the cursor by pressing the 'Backspace' key. (Instantiation)

Training at a low level involves copying the existing frame and adding and deleting nodes to it. For example (from Pollock) "deleting a character on WordPerfect® is like deleting a character on WordStar (copied frame), except that you use the 'Backspace' key (added node) and not the 'Del' key (removed node)".

Although Pollock's research involved the transfer of skills between two different software tools and not two versions of the same tool, her findings could still be applied in this context. However, further research will have to be done in this area.
7.6.2.3 Meta-cognitive training

This can be a key aspect of the development of skills. As upgrades will continue to happen, even with greater frequency, learning to learn can be a strategic skill. Users should be made aware of the interference effects they can experience and the mechanisms they can use to minimise those effects. Understanding how learning occurs, end-users can decide on particular strategies that will help their learning style. A by-product of this training can be the increase of the tolerance levels for the assimilation of further changes.

7.6.2.4 Differential training

Training only on what has changed can prove more effective and efficient for the users and the organisation. This approach can largely reduce training times, maintain the user's attention throughout the training and minimise downtimes.

This is a similar approach to the one described in 7.6.2.2-Levels of training, helping the learners associate low-level frames, adding or deleting frames to them.

7.6.2.5 Distributed practice

When new material is learnt under distributed practice, there is a reduction on the interference effects (Underwood and Ekstrand, 1966). Although we are not aware of research done using distributed practice on the subject of this thesis, it can prove beneficial as a training strategy. In this case, computer-based training could be used to support small amounts of practice during a period of time.
7.6.3 External cues

This refers to all visual cues, other than the ones designed into the application. Supporting a change with visual reminders will aid users in the isolation of new skills, from the interference effects caused by previous ones. Examples of external cues are job-aids, posters, workplace layout, etc. These visual cues can be part of the communication and training activities. They will remain in the workplace, reminding users of the changes and minimising the interference effects.

7.7 Supporting an upgrade

Help desks are critical in the support for the installation of new software. In the case of upgrades, help desk personnel could use specific strategies to assist users through the changes. If help desks understand what the characteristics of skill transfer are and they are made part of the overall implementation strategy, they will be able to help users in the faster assimilation of the change. For example, they could point out the relation Goal / Method to the end-user, facilitating the transfer of skills.

Help desks can also gather data from the end-users in regards to when new functionality, speed or other resources are needed to support performance and inform management's further upgrade decisions.

7.8 Chapter Conclusions

This chapter explored the implications that empirical findings from previous chapters can have in an organisational context, proposing an explicit strategy for the management of upgrades.
Although this strategy still needs to be tested in the real world, the intention here is to propose tasks that different stakeholders can perform in order to support users in the assimilation of software upgrades. The validity of each of these tasks will be determined by further investigation done in this arena.

Research implications and final conclusions will be covered in the next chapter.
8. Conclusions

8.1 Introduction

Many lessons were learned over these past years, in the course of this research into the effects that software upgrades have for organisations and their workers. The following is an account of the major lessons, both in terms of scientific findings and personal growth as a researcher.

8.2 Lessons learned

To tackle the problem of upgrades in organisations a multidisciplinary framework was needed. It would not have been possible to address the issue of explaining the phenomena of human costs of software upgrades without gathering the perspectives from Organisational Psychology, Cognitive Psychology and Technology together.

An organisational perspective provided information on the drivers for software upgrades, identifying that a great proportion of what triggers upgrades in organisations involve reasons not directly related to the end-user need for better functionality or improvement in current processes. Furthermore, what we called indirect drivers in this thesis, create a situation in which upgrades are decided based on market or vendor pressure. In consequence, most managerial attention is focused on the state of the market and vendor promises, overlooking the needs of end-users. This finding, in itself, would have not represented a problem if it did not have detrimental consequences for the performance of end-users.
Exp 46.

### Unforced

<table>
<thead>
<tr>
<th>Value</th>
<th>Metal</th>
<th>Height</th>
<th>Ball-Side</th>
<th>Value</th>
<th>B unin</th>
<th>Bno/HE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Availability</td>
<td>0</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Package-Side</td>
<td>0</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
</tbody>
</table>

### Forced

<table>
<thead>
<tr>
<th>Value</th>
<th>Metal</th>
<th>Fullness</th>
<th>Pack, Snr</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Availability</td>
<td>1</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>Package-Side</td>
<td>2</td>
<td>3</td>
<td>13</td>
</tr>
</tbody>
</table>

D Neither Ordered

<table>
<thead>
<tr>
<th>Metal</th>
<th>Fullness</th>
<th>Ball Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
In seeking to understand the user impact, it was the cognitive psychology field that helped in the understanding of user problems and the identification of the root causes of those problems. An end-user perspective verified the initial finding of the theses, in that upgrades are generally decided based on indirect pressures and not based on the needs of the users.

At this stage in the thesis we learned that smaller changes tended to cause more problems than large and more visible changes. This counterintuitive finding became the essence of the thesis, driving the need for further investigation on the issue. The lack of support in the management of upgrades coincides with the smaller changes, creating a bigger problem for end-users.

Understanding technology, allowed the identification of changes between versions of the same software, informing the design of an experiment to test the findings and a possible solution.

Another important finding came through the field experiment, in which subjects were asked to perform some tasks using an upgraded version of the tool they normally use. Subjects that used the version which looked more different from the one they were used to, performed better and reported more satisfaction with the change than those that experienced very little change. The use of visual cues could be a possible direction for the minimisation of transfer problems.
8.3 Reflection on the research methods used in the thesis

In order to conduct research in this area, I started with a macro field-based type of setting, "casting a wide net" on the possible problems, and moved towards a more micro approach, when those problems were identified.

The strategy of the thesis proved beneficial for the investigation of the problems; its strength was probably the usage of field-work methods for the initial identification of problems in combination with laboratory-style experimentation that allowed for a better control of the variables, as research progressed.

As described earlier, this thesis crossed the boundaries between different fields as required by the topic of research. Methods were combined and in some cases novel approaches were taken. The following is an account of the lessons learned, both in terms of what worked and what did not work, from a methodological perspective.

8.3.1 First study: Organisational survey (Chapter 3)

8.3.1.1 Strengths

The method used for this part of the study used novel techniques, not yet widespread in the academic research world. First, the use of an electronic media such as e-mail, to distribute a questionnaire, second, the request to respondents to forward the original questionnaire to others and third, to use intermediaries (consultants) to report on the target group (decision makers in organisations).
The use of e-mail as a distribution technique proved successful, as indicated by the response rate, the amount of information in the responses and the promptness with which answers were received. Asking subjects to redistribute the questionnaire to others enlarged the sample size, maintaining the core characteristics of the sample.

Another important gain was the use of consultants as intermediaries to report on organisational issues. Although closely involved with their client organisations, they reported with no constraints on the “negative” (as well as the “positive”) aspects of how upgrades are handled in organisations.

8.3.1.2 Weaknesses

The survey asked respondents to “summarise” their experiences in answer to three questions. Unfortunately, this did disguise information on the frequency of each problem that was reported on. For that reason, this study does not show any frequency of answers. For example, in 3.5-Question 1: “What drives organisations to upgrade their software?”, it would have been methodologically incorrect to report that one particular driver is more important than others, or has been reported with more frequency than others. This would not have been an indicator of the real frequency of that driver in organisations.

The questionnaire could have also benefited from a better identification of the profile of the respondents. Having demographic data would have allowed us to perform a categorisation of the responses.
8.3.2 Second study: End-user survey (Chapter 4)

8.3.2.1 Strengths
Finding users that have recently upgraded and can recall the details of the upgrade can be a difficult task. In this thesis, a very large number of possible targets were initially contacted through a questionnaire, followed by an identification of the cases that represented upgrades as defined in the thesis and concluding with interviews with those that could report on the subject. Finding a way to locate those specific subjects was a key methodological problem that was successfully achieved.

8.3.2.2 Weaknesses
Because of the characteristics of the study, the size of the sample was not established by the research but was a consequence of the method used and the amount of secretarial personnel that had upgraded at the university. Certainly, limiting this study to university secretaries limits the generalisation of some of the findings.

8.3.3 Third study: Word 2000 experiment (Chapter 6)

8.3.3.1 Strengths
The field experiment was able to reach people in their real work environment, using their current skills in using a tool and applying them to a realistic upgrade of software.

If this study had been conducted in a laboratory environment it could have allowed the experimenter to have more control over the variables, but much of the ecological validity of the experiment could have been lost.
8.3.3.2 Weaknesses

This study looked only at the initial performance times, as users were presented with an upgrade condition. Skills develop over time and the initial reactions observed in the experiment could or could not indicate a persistent direction of the effect. Therefore a longitudinal study of the same sort would be recommended.

8.4 Future directions of software upgrades

There is no indication that the frequency of upgrades will decrease. In fact, there is evidence of the emergence of new indirect drivers that will force organisations to frequently upgrade their software over the next years. Some of these drivers are:

- The Year 2000 compliance
- The European Monetary Union's one currency
- The increase in the use of internet and intranets

These, together with the drivers identified in this research, could sustain, if not increase, the effects that the upgrades have on the user population.
Training could become an integral part of newly released products. Packages are already including standard computer-based training programs for the end-users and it will not be cost-effective for organisations to develop their own in-house training. This could have negative effects on user's performance, unless that packaged training takes into account the particularities that exist in the cases in which users already use a previous version of the package.

Further investigation and communication of costs of the Total Cost of Ownership model could help organisations in the minimisation of human costs, informing management upgrade decisions.

However, there are other initiatives that could hide these costs. By automating the technical side of the upgrades, the Zero Administration initiative that Microsoft will release in the near future, could give management the impression that costs are being reduced. The reduction of the installation costs of upgrades will not automatically reduce human costs and could increase the frequency with which organisations upgrade software as it will be apparently cheaper to embark on software upgrades.

8.5 Implications for further research

As a result of this research, a number of related issues come to the fore for further investigation. Because this thesis draws from both cognitive and organisational psychology fields, implications for further research have been identified for each of them.
Cognitive psychology:

(1) It could be important to evaluate the role of depth of processing in the minimisation of the interference effects observed in this study. Subjects that spent more time “learning” the new location of objects using the upgraded version of the software, showed effects of less interference. A study in which either depth of processing or visual cues are isolated and controlled is needed for a firm conclusion on how they affect interference in the users’ upgrades of software.

(2) In the experiment described in Chapter 6, learning was tested in just one of its forms, looking at immediate learning after a short interpolated distracter.

An investigation of how distributed practice can affect the interference effects of upgrades is needed. The effect found by Underwood and Ekstrand (1966) could prove beneficial for the training efforts of future upgrades.

(3) Users develop automatic skills as they become proficient with a software tool. During this study it was observed that successful users moved from automatic skills into meta-cognitive activities that allowed them to “survive” in a new environment. Further research on how people decide to switch from one mode to another could be beneficial for the HCI arena.
Organisational psychology:

(1) Chapter 7 proposed a strategy for the minimisation of human costs of software upgrades in organisations. An action-research practical evaluation of this strategy is needed in order to assert its validity.

This strategy could be tested by implementing it only within a specific group or department in an organisation, using the rest of the organisation as a control group (as they implement the upgrade in the traditional way). This sort of study would help to further develop the strategy proposed.

(2) This thesis identified that one of the problems of upgrades is that decision makers in organisations do not have the appropriate information on the real costs of upgrades. The creation of a cost-benefit model for organisations would be an invaluable tool to reduce the effects upgrades have over people.
This thesis was written using Microsoft Word® 7 and in no way would this work have been possible without it. I am very thankful!

However, three months towards the end of this thesis, I decided to upgrade the wordprocessor, ignoring many of the findings reported in this study. I installed Office 97, a very fancy package that replaced my current version of Word® as well.

After spending a few hours installing the new package, and as the many files of my PhD were opened in the new version, major problems occurred. For example, bullet points triggered irrecoverable errors in the new version.

After many days of conversations with poorly informed help desk personnel, Microsoft USA apologised for the problem, mentioning that they had withdrawn the release from the market, and that a fix to the problem would be available by the end of the year.

As a consequence of the upgrade, most of this thesis had to be re-typed in the previous version, using a set of paper backups. An entire week was devoted to this experience, and during that time only one new page was written. This one.
9. References


181


10. Appendices

Appendix I - Organisational Survey
Appendix II - End-User Survey
Appendix III - Transfer Theories
Appendix IV - Word 2000 Experiment
Appendix V - Job Aids for Decision Makers and Implementers in Organisations
10.1 Appendix I - Organisational Survey

- E-mail Questionnaire
- Results
10.2 Appendix II - End-User Survey

- The Questionnaire
- Interview Scripts
- Results
10.3 Appendix III - Transfer Theories

- Transfer Theories
10.4 Appendix IV - Word 200 Experiment

- Screen Shots Used for Visual Inspection of Similarities Between Versions
- Questionnaire Used After the Experiment
- Internal Log (tracing sequence followed and time spent)
- Results from Experimental Group
- Results from Control Group
10.5 Appendix V - Job Aids

- Checklist for Decision Makers
- Job Aid for Designers
Checklist for Decision Makers in Organisations

Instructions: Use the following checklist as a tool that could help you understand the reasons that are driving a software upgrade in your organisation. Rank each question in both columns (using a 1 to 10 scale, 10 being your complete agreement with the statement). You can add your own drivers in the empty spaces provided. Add the numbers in both columns and write them at the bottom of the checklist. If the result from the left column is higher than the one from the right, you are upgrading for the right reasons for the organisation. If the result is lower, there is a high probability that external drivers are forcing the upgrade. This could be a costly decision for the organisation and should be analysed carefully.

<table>
<thead>
<tr>
<th>We are upgrading software because...</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>...our business requires new functionality.</td>
<td>...the current version will not be supported by the hardware or software provider in the future.</td>
</tr>
<tr>
<td>...the cost of maintaining existing systems is higher than using new ones.</td>
<td>...we don't want to fall behind.</td>
</tr>
<tr>
<td>...it represents a competitive advantage.</td>
<td>...our people don't want to feel their skills are falling behind.</td>
</tr>
<tr>
<td>...it has to support organisational changes.</td>
<td>...we like using new technology.</td>
</tr>
<tr>
<td>...it has to support new job requirements.</td>
<td>...the old one will not run under the new hardware.</td>
</tr>
<tr>
<td>...it will improve productivity.</td>
<td>...it comes with the new package (Office, Smart Suite, etc.)</td>
</tr>
<tr>
<td>...the new version allows the handling of objects in a better way.</td>
<td></td>
</tr>
<tr>
<td>...the current version has too many technical problems.</td>
<td>...there is incompatibility with other users / departments / organisations / clients / suppliers.</td>
</tr>
<tr>
<td></td>
<td>...the current one is not compatible with new operating systems and tools.</td>
</tr>
<tr>
<td></td>
<td>...the upgrade is free.</td>
</tr>
</tbody>
</table>

**TOTAL** | **TOTAL**
Checklist for Designers of the Upgrade in Organisations

Instructions: Use the following table as a tool that could help you make decisions on whether to intervene in the design of an upgrade (customising the interface through colours, icons, etc.). You could follow these steps:

1. Identify critical parts (dialogues, tasks, etc.) within the software that is being upgraded. Criticality could be defined by risk of errors, frequency of use (very infrequent usage could increase usability problems), parts where actions cannot be undone, etc.

2. Identify user goals. These could be the same as with the previous tool, or could have changed. Locate these in the table below.

3. Identify the method that would be used to reach the goals (identified in point 2). As before, the methods could have remained unchanged or could be different from the previous version of the tool. Locate these in the table below.

4. Evaluate with users their perception of the change (use an appropriate sample, not your colleagues in the IT department). Show the new interface to the users, ask them to perform the critical tasks you have identified and ask them to rate the similarity between old and new versions (you can use a low / high scale here). Again, locate this in the table below.

5. This tool suggests different levels of intervention (none, hide or exaggerate) based on the goal, method and similarity levels you have identified. The recommended actions for each of them are:
   - None: You could leave the interface as is. There is no need to alter it.
   - Hide: You could reinforce consistency between the previous version and the new one, through modifying the new interface. Use same colours, icons and distribution of objects in the screen, if possible.
   - Exaggerate: In this case, you need to "flash out" the change, by adding colours, sounds (if appropriate), repositioning objects in the screen. Any alterations to the interface that would help users see that the new tool is NOT similar to the previous one.

<table>
<thead>
<tr>
<th>Goal (old ⇒ new)</th>
<th>Method</th>
<th>Perceived Similarity</th>
<th>Need for intervention</th>
</tr>
</thead>
<tbody>
<tr>
<td>old</td>
<td>old</td>
<td>low</td>
<td>hide</td>
</tr>
<tr>
<td>old</td>
<td>old</td>
<td>high</td>
<td>none</td>
</tr>
<tr>
<td>old</td>
<td>new</td>
<td>low</td>
<td>none</td>
</tr>
<tr>
<td>old</td>
<td>new</td>
<td>high</td>
<td>exaggerate</td>
</tr>
<tr>
<td>new</td>
<td>old</td>
<td>high</td>
<td>exaggerate</td>
</tr>
<tr>
<td>new</td>
<td>new</td>
<td>low</td>
<td>none</td>
</tr>
<tr>
<td>new</td>
<td>new</td>
<td>high</td>
<td>none</td>
</tr>
</tbody>
</table>