



Enabling Workspace Transference

A cross-environment framework for application and data portability

Matt Carter BSc, MIT

Thesis submitted for the degree of Doctor of Philosophy to the School of Information
Technology, Bond University, Australia

February 2012

Table of Contents

Abstract	6
Statement of Original Authorship	7
1 Introduction	1
1.1 Motivation and statement of problems	1
1.2 Research question	2
1.3 Research deliverables	3
2 Literature Review	5
2.1 History of software portability	5
2.2 Existing workspace transference methodologies.....	6
2.2.1 Application podding.....	7
2.2.2 Application stasis	8
2.2.3 Virtualisation.....	10
2.2.4 Teleportation	13
2.2.5 System emulation	14
2.2.6 System boot.....	15
2.2.7 Web-based systems	16
2.2.8 Summary of workspace transference methodologies.....	17
2.3 Other portability issues.....	17
2.3.1 File synchronisation	18
2.3.2 Security	19
2.4 Literature Summary.....	19
3 Workspace Transference Framework	22
3.1 Framework aims	22
3.1.1 Application portability	23
3.1.2 Data portability	25
3.1.3 Data sharing	26
3.1.4 Environmental interaction.....	27
3.1.5 Interface requirements.....	28
3.2 Portability infrastructure.....	29
3.2.1 Application portability	29
3.2.2 Data portability & sharing.....	31
3.2.3 Environmental Settings	31
3.2.4 Interface Requirements	32
3.3 Framework Summary	33
4 Evaluation Criteria	34
4.1 Persona use cases.....	34
4.1.1 Novice	36
4.1.2 Knowledge Worker	37
4.1.3 Developer	39
4.1.4 Rejected personas.....	40
4.2 Technical criteria	40
4.2.1 Distribution	40
4.2.2 Synchronisation.....	42
4.2.3 Package management.....	43
4.2.4 Error handling	44
4.2.5 Development environment.....	46
4.3 Rejected evaluation criteria	47
4.4 Evaluation Criteria Summary	48
5 Prototype Design	50

5.1 Framework integration	50
5.1.1 Application portability	50
5.1.2 Data portability & sharing	51
5.1.3 Environmental interaction	53
5.1.4 Interface requirements	55
5.1.5 Framework integration summary	55
5.2 Logical Architecture	56
5.2.1 Client-Side Components	57
5.2.2 Server-Side Components	62
5.3 Component interaction	63
5.3.1 Prototype installation	64
5.3.2 Synchronisation	64
5.3.3 Package installation	65
5.3.4 Package upgrade	65
5.3.5 Package removal	65
5.4 Prototype design summary	66
6 Initial Prototype	67
6.1 Technology selection	67
6.2 Evaluation methodology identification	69
6.3 Framework compliance	71
6.3.1 Application portability	71
6.3.2 Data portability & sharing	72
6.3.3 Environmental interaction	73
6.3.4 UI requirements	73
6.3.5 Framework compliance summary	74
6.4 Persona use cases	75
6.4.1 Novice Persona	75
6.4.2 Knowledge Worker Persona	82
6.4.3 Developer Persona	87
6.4.4 Persona use case summary	92
6.5 Technical criteria	95
6.5.1 Distribution	95
6.5.2 Synchronisation	96
6.5.3 Package management	97
6.5.4 Error handling	98
6.5.5 Development environment	98
6.6 Prototype evaluation and weaknesses	100
6.6.1 Prototype portability	102
6.6.2 Slow bootstrap	103
6.6.3 Community support	103
6.6.4 GUI integration	103
6.6.5 Maintenance	104
6.6.6 Revised action plan	105
7 Revised Prototype	106
7.1 Framework compliance	106
7.1.1 Application portability	106
7.1.2 Data portability & sharing	108
7.1.3 Environmental Interaction	110
7.1.4 Interface requirements	112
7.1.5 Framework compliance summary	114
7.2 Persona use cases	114
7.2.1 Bootstrap time	115
7.2.2 PyPredict	116
7.2.3 Web interface	116
7.2.4 Exception handling	117
7.2.5 Persona use case summary	117

7.3 Technical criteria	117
7.3.1 Distribution	117
7.3.2 Synchronisation	118
7.3.3 Package management	120
7.3.4 Error handling	121
7.3.5 Development environment	122
7.4 Prototype Evaluation	123
8 Conclusion	126
8.1 Thesis summary	126
8.2 Future research	127
A Glossary of terms	129
B Logic Model	132
References	161

List of figures

Figure 1. The workspace transference framework seperated as conceptual layers	30
Figure 2. Peer-to-peer prototype networking model	52
Figure 3. Network layout of centrally connected connected prototype nodes	52
Figure 4. Logical overview diagram of the prototype architecture	57
Figure 5. Emulation logic model	60
Figure 6. The prototype installation sequence presented in its initial form	76
Figure 7. The prototype installation begins	77
Figure 8. Prototype installation is completed	77
Figure 9. The prototype package installation screen	78
Figure 10. Installation status of a specific package	78
Figure 11. The word processor installation of is complete.....	79
Figure 12. System tray icons displaying a installed software.....	80
Figure 13. Unsaved work in the OpenOffice word processor	80
Figure 14. The same document now with purposely overwritten text	81
Figure 15. Extended installation screen showing more options	83
Figure 16. The now installed Firefox package is updated in the package manager window	84
Figure 17. The installed Firefox package's main window	85
Figure 18. The same web-browsing session now installed on a Microsoft Windows machine	87
Figure 19. The result of a simple Perl program being compiled on an Ubuntu Linux system.....	89
Figure 20. The same application running under the prototype's portability environment.....	90
Figure 21. Installing the prototype within a purely console-driven environment	91
Figure 22. The package creation process as seen in the console	92
Figure 23. Symbols used in this chapter to denote the logical flow of execution	133
Figure 24. Overall conceptual logical flow diagram of the core system	134
Figure 25. Logical flow diagram of the GUI sub-system.....	139
Figure 26. Logical flow diagram of the watcher sub-system	142
Figure 27. Logical flow diagram of the synchroniser sub-system	144
Figure 28. Logical flow diagram of the Unison sub-system	146

Abstract

Workspace transference is defined as the transportation of applications and associated data between different computer environments regardless of hardware, operating systems or networks. Existing research only provides partial solutions to workspace transference. A gap still exists to unify this research under a set of requirements. Therefore there is a need for an integrated framework that binds specific requirements together to provide a complete solution. This thesis defines the concept of workspace transference and constructs a framework which provides a solution to this research problem.

A series of evaluation criteria will then be created to assess any constructed solution followed by a prototyping methodology to construct a series of prototypes to test against the workspace transference framework. Two prototype implementation cycles are described using the Perl and Python development environments. Each prototype is tested against a series of technical criteria derived from the workspace transference framework and a use case task list generated for a range of user personas. The final prototype performed all the use case tasks well and met the stringent technical criteria. This outcome indicates that a comprehensive solution to workspace transference is a feasible and practical proposition.

The workspace transference framework unifies the existing practical solutions for workspace transference and provides a technological yardstick against which future solutions can be judged.

The final version of the prototype named WorkTran meets all the requirements of the workspace transference framework. WorkTran supports three major operating systems on Windows, Macintosh OSX and several varieties of Linux, and forms a significant research outcome.

Statement of Original Authorship

The work contained in this thesis has not been previously submitted for a degree or diploma at any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signature:

Date: