Integrating user-centred design with software engineering using formal methods: a personal view

M. D. Harrison[a,1]

[a]*School of Computing, Newcastle University, Urban Sciences Building, Newcastle upon Tyne, NE4 5TG, UK*
[b]*Department of Computer Science, The Computational Foundry, Swansea University Bay Campus, Fabian Way, Swansea, SA1 8EN*

Abstract

When I arrived in York in 1983 Harold Thimbleby, Colin Runciman and Andrew Monk were already talking about ways in which their different activities could contribute to understanding the interaction between users and computers. Harold had already done relevant work as part of his thesis with Richard Bornat on wysiwig editors. This had involved the development of the ded text editor. Harold had developed Generative User Engineering Principles (gueps), non functional requirements, as principles that would govern such WIMP interfaces [1, 2]. Colin Runciman was a functional programmer and Andrew Monk a cognitive psychologist. My background was formal methods, originally denotational semantics with practical experience of designing and implementing computer systems in Computer Analysts and Programmers Ltd. and Inmos. I was particularly interested in formal specification notations such as Z [3], VDM [4] and Larch [5] and the possibility of proving requirements of designs using these techniques, particularly Harold's gueps.

In the early years Colin and Alan Dix (who joined as a Research Assistant on our first "interdisciplinary" research project) developed the PiE model to capture some of these principles by The PiE model [6] was a very simple formalisation of interactive systems that enabled the description of some useful principles, in particular *predictability*.

Later models extended this simple structure by making a distinction between state and and display [7, 8], and considered requirements of concrete systems, for example Gregory Abowd and Alan Dix's requirements relating to "undo" [9, 10]. The work evolved naturally to consider the role of formal software engineering notations. David Duke [11], in particular, developed notions of "interactor" inspired by Paterno and Faconti's work using LOTOS [12]. David used both Object Z and MAL (Modal Action Logic) in his work and considered issues associated with refinement in formal specification of interactive systems [13].

A further stage in this process involved the integration of user and system modelling notations used in HCI. This theme was an important component of the AMODEUS EU research project [14] and culminated in David Duke, Ann Blandford and Paul Curzon's work modelling users [15, 16]. Additional work

integrating formal software engineering and behavioural models has concerned the broader integration of models of context, for example using Hutchin's work [17]. Pete Wright and Bob Fields [18] developed a model of resources that can be used to produce a richer understanding of the environment in which an interactive system is developed. This was used with MAL in later work by Gavin Doherty and Jose Campos [19].

The latest stage in this work has involved the use of tools developed to analyse and prove properties of specifications. These tools have been applied to interactive systems and properties can be derived from "templates" that aid their refinement to designs. These templates have evolved from Harold's original gueps. Karsten Loer [20] produced an approach using statemate and model checking while Jose Campos used MAL and SMV [21]. The latter approach has been used, and is being used, to model and analyse existing medical devices [22, 23]. Current work involving Paolo Masci also involves the use of the PVS theorem proving assistant to properties of systems [24].

References

[1] H. W. Thimbleby, Character level ambiguity: consequences for user interface design, International Journal of Man-Machine Studies 16 (1982) 211–225.

[2] H. W. Thimbleby, Generative user-engineering principles for user interface design, in: B. Shackel (Ed.), Human-Computer Interaction — INTERACT'84, North-Holland, 1985, pp. 661–666.

[3] J. M. Spivey, The Z reference manual, Tech. rep., Oxford University Programming Research Group, DPhil Thesis (1986).

[4] J. Bicarregui, J. Fitzgerald, P. Lindsay, R. Moore, B. Ritchie, Proof in VDM: A Practitioner's Guide, FACIT Series, Springer-Verlag, 1994.

[5] J. Guttag, J. Horning, J. Wing, Larch in five easy pieces, Tech. Rep. 5, DIGITAL Systems Research Center (1985).

[6] A. J. Dix, C. Runciman, Abstract models of interactive systems, in: P. Johnson, S. Cook (Eds.), People and Computers: Designing the interface, Cambridge University Press, 1985, pp. 13–22.

[7] M. D. Harrison, A. J. Dix, Modelling the relationship between state and display in interactive systems, in: P. Gorny, M. J. Tauber (Eds.), Visualization in Human Computer Interaction, no. 439 in Lecture Notes in Computer Science, Springer-Verlag, 1990, pp. 241–249.

[8] M. D. Harrison, A. J. Dix, A state model of direct manipulation, in: M. D. Harrison, H. W. Thimbleby (Eds.), Formal Methods in Human Computer Interaction, Cambridge University Press, 1990, pp. 129–151.

[9] G. Abowd, A. Dix, Giving undo attention, Interacting with Computers 4 (3) (1992) 317–342.

[10] A. J. Dix, Formal Methods for Interactive Systems, Academic Press, 1991.

[11] D. J. Duke, M. D. Harrison, Abstract interaction objects, Computer Graphics Forum 12 (3) (1993) 25–36.

[12] G. Faconti, F. Paternò, An approach to the formal specification of the components of an interaction, in: C. Vandoni, D. Duce (Eds.), Eurographics 90, North-Holland, 1990, pp. 481–494.

[13] D. J. Duke, M. D. Harrison, Mapping user requirements to implementations, Software Engineering Journal 10 (1) (1995) 13–20.

[14] D. J. Duke, P. J. Barnard, J. May, D. A. Duce, Systematic development of the human interface, in: Asia Pacific Software Engineering Conference, IEEE Computer Society Press, 1995, pp. 313–321.

[15] D. J. Duke, P. J. Barnard, D. A. Duce, J. May, Syndetic Modelling, Human Computer Interaction 13 (4) (1998) 337–393.

[16] R. Rukšėnas, P. Curzon, A. Blandford, J. Back, Combining human error verification and timing analysis: a case study on an infusion pump, Formal Aspects of Computing (2013) 1–44doi:10.1007/s00165-013-0288-1.
URL http://dx.doi.org/10.1007/s00165-013-0288-1

[17] E. Hutchins, Cognition in the Wild, MIT Press, 1994.

[18] P. Wright, R. Fields, M. Harrison, Analyzing human-computer interaction as distributed cognition: the resources model, Human-Computer Interaction 15 (1) (2000) 1–42.

[19] J. C. Campos, G. Doherty, M. D. Harrison, Analysing interactive devices based on information resource constraints, International Journal of Human-Computer Studies 72 (2014) 284–297.

[20] K. Loer, M. Harrison, An integrated framework for the analysis of dependable interactive systems (IFADIS): its tool support and evaluation, Automated Software Engineering 13 (4) (2006) 469–496.

[21] J. C. Campos, M. D. Harrison, Model checking interactor specifications, Automated Software Engineering 8 (2001) 275–310.

[22] M. Harrison, J. Campos, P. Masci, Reusing models and properties in the analysis of similar interactive devices, Innovations in Systems and Software Engineering 11 (2) (2015) 95–111.

[23] M. D. Harrison, L. Freitas, M. Drinnan, J. C. Campos, P. Masci, C. di Maria, M. Whitaker, Formal techniques in the safety analysis of software components of a new dialysis machine, Science of Computer Programming 175 (2019) 17 – 34.

[24] M. Harrison, P. Masci, J. Campos, Verification templates for the analysis of user interface software design, IEEE Transactions on Software Engineering 45 (8) (2019) 802–822.