A Compliant Persistent Architecture

Ron Morrison,† Dharini Balasubramaniam,† Mark Greenwood,‡ Graham Kirby,†
Ken Mayes,‡ Dave Munro,◇ and Brian Warboys‡

†*School of Mathematical and Computational Sciences*
University of St Andrews
North Haugh, St Andrews
Fife, KY16 9SS, Scotland
Email: {ron, dharini, graham}@dcs.st-and.ac.uk

‡*Department of Computer Science*
University of Manchester
Oxford Road, Manchester
M13 9PL, England
Email: {markg, ken, brian}@cs.man.ac.uk

◇*Department of Computer Science,*
University of Adelaide,
South Australia 5005, Australia
Email: dave@cs.adelaide.edu.au

The growing requirements of persistent application systems challenge software engineers to provide the appropriate architectural infrastructure. Where the structure of the data is simple, file systems and operating systems are sufficient to supply the infrastructure for the storage and use of data and the execution of programs. When the structure of the data becomes more sophisticated, data models implemented in databases may be used. The contribution of orthogonal persistence [ABC+83] is to integrate the notions of long and short term data and to allow programs to be treated as first class data objects [AM85]. As the application system evolves, more data, meta-data, programs and users are accumulated and must be accommodated by a further step in architectural design. While process modelling systems [War89] provide the conceptual support for this evolutionary style, there is no agreed software architecture to supply these needs [FKN94].

The challenge in supporting process modelling systems is in efficiently providing the radically varying facilities required by different process models [BPR91, RR96, WLM+98]. When implemented on a conventional software architecture, there is often a repetition and duplication of both mechanism and policy in different architectural components which conflict with each other and add complexity to the overall system. The repetition occurs when the same policies are implemented in different components, and duplication occurs when the policies in different components are radically different and may interact in an undesirable manner. This complexity makes it difficult to predict how the system will perform, particularly in terms of failure semantics and run-time efficiency.

Process modelling systems may be seen as archetypal data intensive applications systems, sometimes referred to as Persistent Application Systems (PASs) [AM95]. They are potentially long-lived, concurrently accessed and consist of large bodies of programs and data. In particular, process models evolve frequently and therefore must accommodate dynamic change within the application domain. Thus, the architectural problems encountered in implementing process modelling systems are similar to, if not the same as, those investigated by the persistent programming community over many years now [ABJ+92, ACO85, AM95, AMP86, BM92, BMM+92, Bro89, CBC+90, DBF+94, Dea88, DRH+92, MDB+85, MS93, Mun93, Ros90].

This paper describes a new architectural approach to the construction of persistent systems that accommodates the needs of particular applications. We describe this as being compliant to these needs. The architecture takes advantage of the fact that persistent systems are well structured to discover the dynamically changing demands of applications; since they define precisely the extent to which they are open, the computation which takes place within the persistent environment may be relatively easily tracked.

The novelty of a compliant architecture is that the components are designed, top down, with the philosophy of fitting the architecture to the needs of the particular application. This contrasts with the more traditional, bottom up, approach of providing static abstract components or layers designed to meet the predicted needs of the majority of applications. The key scientific advance, in our approach, is to separate mechanism and policy, a technique that has been used before but not all at once, and consistently, on every component of the architecture.

We report here on our first attempt at building a compliant architecture. We start by proposing a generic architecture for compliance and then show how it may be instantiated with components. We also show how the architecture operates in a manner that is compliant to a target application.

The key components of our initial compliant architecture are: a nano-kernel, HWO, and its associated library-based operating system which together constitute the Arena system [May93, MB97]; a new persistent programming language, ProcessBase [MBG+99]; and the mechanisms for defining and separating policy and mechanism. We postulate, since we have not yet measured, that the benefits of compliant architectures will be a reduction in complexity, with corresponding gains in flexibility, portability, understandability in terms of failure semantics, and performance.

References

ABC+83 Atkinson, M.P., Bailey, P.J., Chisholm, K.J., Cockshott, W.P. & Morrison, R. “An Approach to Persistent Programming”. Computer Journal 26, 4 (1983) pp 360-365.

ABJ+92 Atkinson, M.P., Birnie, A., Jackson, N. & Philbrow, P.C. “Measuring Persistent Object Systems”. In Persistent Object Systems, Albano, A. & Morrison, R. (eds), Springer-Verlag, Proc. 5th International Workshop on Persistent Object Systems (POS5), San Miniato, Italy, In Series: Workshops in Computing, van Rijsbergen, C.J. (series ed), ISBN 3-540-19800-8 (1992) pp 63-85.

ACO85 Albano, A., Cardelli, L. & Orsini, R. “Galileo: a Strongly Typed, Interactive Conceptual Language”. ACM Transactions on Database Systems 10, 2 (1985) pp 230-260.

AM85 Atkinson, M.P. & Morrison, R. “Procedures as Persistent Data Objects”. ACM Transactions on Programming Languages and Systems 7, 4 (1985) pp 539-559.

AM95 Atkinson, M.P. & Morrison, R. “Orthogonally Persistent Object Systems”. VLDB Journal 4, 3 (1995) pp 319-401.

AMP86 Atkinson, M.P., Morrison, R. & Pratten, G.D. “A Persistent Information Space Architecture”. In Proc. 9th Australian Computing Science Conference, Australia (1986).

BM92 Brown, A.L. & Morrison, R. “A Generic Persistent Object Store”. Software Engineering Journal 7, 2 (1992) pp 161-168.

BMM+92 Brown, A.L., Mainetto, G., Matthes, F., Mller, R. & McNally, D.J. “An Open System Architecture for a Persistent Object Store”. In Proc. 25th International Conference on Systems Sciences, Hawaii (1992) pp 766-776, Technical Report ESPRIT BRA Project 3070 FIDE FIDE/91/31.

BPR91 Bruynooghe, R.F., Parker, J.M. & Rowles, J.S. “PSS: A System for Process Enactment”. In Proc. 1st International Conference on the Software Process: Manufacturing Complex Systems, Redondo Beach, California, Dowson, M. (ed) (1991) pp 142-158.

Bro89 Brown, A.L. “Persistent Object Stores”. Ph.D. Thesis, University of St Andrews (1989).

CBC+90 Connor, R.C.H., Brown, A.L., Carrick, R., Dearle, A. & Morrison, R. “The Persistent Abstract Machine”. In Persistent Object Systems, Rosenberg, J. & Koch, D.M. (eds), Springer-Verlag, Proc. 3rd International Workshop on Persistent Object Systems, Newcastle, Australia, In Series: Workshops in Computing, van Rijsbergen, C.J. (series ed) (1990) pp 353-366.

DBF+94 Dearle, A., di Bona, R., Farrow, J., Henskens, F., Lindstrm, A., Rosenberg, J. & Vaughan, F. “Grasshopper: An Orthogonally Persistent Operating System”. Computer Systems 7, 3 (1994) pp 289-312.

Dea88 Dearle, A. “On the Construction of Persistent Programming Environments”. Ph.D. Thesis, University of St Andrews (1988).

DH99 Dearle, A. & Hulse, D. “Operating System Support for Persistent Systems: Past, Present and Future”. Submitted to Software-Practice and Experience (1999).

FKN94 Finkelstein, A., Kramer, J. & Nuseibeh, B. (eds) Software Process Modelling and Technology. Research Studies Press (1994).

May93 Mayes, K.R. “Trends in Operating Systems Towards Dynamic User-Level Policy Provision”. University of Manchester Technical Report UMCS-93-9-1 (1993).

MB97 Mayes, K.R. & Bridgland, J. “Arena - a Run-Time Operating System for Parallel Applications”. In Proc. 5th EuroMicro Workshop on Parallel and Distributed Processing (PDP’97) (1997) pp 253-258.

MDB+85 Morrison, R., Dearle, A., Bailey, P.J., Brown, A.L. & Atkinson, M.P. “The Persistent Store as an Enabling Technology for Integrated Project Support Environments”. In Proc. 8th IEEE International Conference on Software Engineering, London (1985) pp 166-172, Technical Report Universities of Glasgow and St Andrews Report PPRR-15-85.

MS93 Mukherjee, B. & Schwan, K. “Experimentation with a Reconfigurable Microkernel”. In Proc. USENIX Symposium on Microkernels and other Kernel Architectures, San Diego, California (1993) pp 45-60.

MS93 Matthes, F. & Schmidt, J.W. “System Construction in the Tycoon Environment: Architectures, Interfaces and Gateways”. In Proc. Euro-ARCH ’93, Hamburg, Spies, P.P. (ed) (1993) pp 301-317.

Mun93 Munro, D.S. “On the Integration of Concurrency, Distribution and Persistence”. Ph.D. Thesis, University of St Andrews. Technical Report CS/94/1 (1993).

Ros90 Rosenberg, J. “The MONADS Architecture - A Layered View”. In Implementing Persistent Object Bases, Dearle, A., Shaw, G.M. & Zdonik, S.B. (eds), Morgan Kaufmann, Proc. 4th International Workshop on Persistent Object Systems, Martha’s Vineyard, USA (1990) pp 215-225.

RR96 Reisig, W. & Rozenberg, G. “Informal Introduction to Petri Nets”. In Lecture Notes in Computer Science 1491, Reisig, W. & Rozenberg, G. (eds), Springer, Proc. Advanced Course on Petri Nets, Dagstuhl, ISBN 3-540-65306-6 (1996) pp 1-11.

War89 Warboys, B. “The IPSE 2.5 Project: Process Modelling as the Basis for a Support Environment”. In Proc. 1st International Conference on System Development Environments and Factories, Berlin, Germany (1989).

WLM+98 Wise, A., Lerner, B.S., McCall, E.K., Osterweil, L.J. & Sutton, S.M. “Specifying Coordination in Processes Using Little-JIL”. Department of Computer Science, University of Massachusetts at Amherst Technical Report 98-38 (1998).