Towards a Benchmark for E-Contract Checking and Monitoring

Alan S. Abrahams*, David M. Eyers†, and Jean M. Bacon†
* *Department of Operations and Information Management, The Wharton School, University of Pennsylvania*
500 Jon M. Huntsman Hall, Philadelphia, PA 19104-6340, USA
† *University of Cambridge Computer Laboratory, William Gates Building, 15 JJ Thomson Avenue, Cambridge, CB3 0FD, United Kingdom*
asa28@wharton.upenn.edu, {David.Eyers, Jean.Bacon}@cl.cam.ac.uk

Abstract

One good quantitative means of gauging research progress is through like-for-like comparative evaluation of software implementations and algorithms. However, a notable omission from the work in the new field of electronic contracting, is a standard means of evaluating performance of the components used for contract consistency-checking and monitoring, against earlier approaches. In this paper we propose a benchmark for assessing the efficiency of such components. We define a replicable experimental framework and use it to test a basic but expressive contract assessment prototype, EDEE. EDEE's execution metrics set the bar for other contract assessment approaches. We hope that this experimental framework will stimulate researchers in the e-contracting community to develop competing data models and algorithms and raise the bar for time and space efficiency of contract checking and monitoring software.

1 Introduction

There has been much recent work on electronic contracting, but we are not aware of any research that evaluates contract-checking efficiency on meaningful tasks. If the state of the art in this field is to advance, objective means are required of assessing the performance of competing implementations on a standard problem set of reasonable complexity.

This paper begins with an overview of related work in e-contracting (Section 2), highlighting the absence of a yardstick for quantitative comparison. Section 3 then sets out the experimental framework we propose for benchmarking in the community. In Section 4, we record the performance of our own implementation, EDEE, using the proposed testing methodology. We conclude with a set of opportunities for improvement in our architecture, and research challenges for the field as a whole.

2 Related Work

Past research in general rule-based performance assessment (§2.1) and in e-contracting (§2.2) pertains:

2.1 General Performance Assessment Benchmarks

Performance benchmarks from the fields of event monitoring, active databases, view containment-checking, and expert systems are not appropriate for e-contracting environments, since these approaches do not express conflicts and violations, and do not keep event histories that are essential for contract-checking purposes.

E-contracting environments would under-perform against existing *event monitoring* and *active database* systems (e.g. [4, 8, 10, 21]) which monitor for low-level system and database events. The most efficient event-monitoring approaches may detect up to 600 events per second against 6 million subscriptions held in RAM [10]. Such approaches treat events as transient, held briefly in memory rather than stored in a database, with rudimentary event types [1, p18], and event attributes being simple string values only. Event monitors are less expressive than expert systems approaches, which perform slowly as they indulge in more complex inferencing.

In the *database view* literature, the MiniCon containment-checking algorithm – demonstrably more scalable than the Bucket and Inverse-Rules approaches [23] – is able to check a query against 1000 views (named queries) in less than one second. MiniCon is targeted at the data integration problem, and rewrites queries in terms of a maximal union of conjunctive queries over available views. Though it could be relevant, it has not been applied to the challenge of contract consistency-checking.

Expert system benchmarks use artificial intelligence problems with small, static rule sets. Typical commercial rule engines (e.g. [5, 13]) use tests such as the NASA Monkeys & Bananas example as their benchmark. ILOG claims to solve this problem in less than 10 minutes [13]. Problems mentioned in the academic expert systems literature include finding optimal seating arrangements for guests at a dinner party ('Manners' benchmark), labelling lines for simple scenes ('Waltz'), designing computer chip circuitry ('Weaver'), or determining the lowest-cost route plan for an aeroplane ('ARP') [7]. The basic 'Manners'

benchmark uses only 8 rules, while the 'Waltz' benchmark uses 33 rules. The most complex of the benchmarks, 'Weaver', uses 1831 facts and 637 rules; it compiles to a rigid constraint network in 2-3 hours and executes in just over 11 minutes. Commercial contract evaluation scenarios, with large numbers of dynamic and conflicting rules cannot be meaningfully assessed against these fixed object-pattern matching application benchmarks, which bear scant relation to workflow scenarios. For contract monitoring and checking, both object-pattern (check if item is covered by a provision) and dynamic pattern-pattern matching (check if provisions overlap) are required. In commercial trade environments, occurrences must be assessed against a large and periodically changing number of applicable provisions, and in the light of complex historical circumstances.

2.2 E-Contracting

A multitude of electronic contracting approaches exist: (e.g. [6, 9, 11, 12, 14 - 20, 22, 24, 27] - see [1] for a detailed review of these and others). However, most of these approaches cannot express or detect conflicting provisions, and many cannot monitor for contract violations. Furthermore, none of these existing approaches specify metrics for assessing the efficiency of contract consistency-checking and monitoring. Previous contract assessment work has been based on small, single contracts. Lee and co-workers [6] have implemented a Petri-Net-based trade procedure executor, whilst Daskalopulu and colleagues [9] provide a finite-state-machine-based conceptual framework for assessment of a small number of obligations (i.e. 2 or 3). Previous work has focused on developing small-scale conceptual solutions to the problem, rather than on studying actual performance of implementations. In the field of legal expert systems, Sergot et al [138], using the relatively limited computing resources available at the time, showed in Prolog that the citizenship of an individual could be determined on the basis of approximately 500 rules of legislation. Performance figures were not provided. We are not aware of any experimental studies of algorithms for ascertaining contract status or for determining the implications of business occurrences on contract consistency during workflow execution. We believe our evaluation framework (§3) is a significant contribution that fills a void in previous work on this topic.

3 Experimental Framework

An experimental framework for e-contract checking should begin with a practical, albeit stylized, example. Take the following trace of a realistic contracting and operational scenario:

```
Scenario A:
    Provision 1:  SkyHi is obliged to pay Steelmans
                  $25,000.
    Provision 2:  Payments of more than $10,000 to a
                  supplier are prohibited.
    Fact 1:       Steelmans is a supplier.
    Fact 2:       SkyHi pays Steelmans $25,000.
```

In terms of contract *checking*, an effective component should quickly notice that the obligation in Provision 1 conflicts with the prohibition in Provision 2 (though this conflict only becomes evident when Fact 1 comes to light). Regarding contract *monitoring*, the component should rapidly flag that Fact 2 violates the prohibition in Provision 2 (and, simultaneously, fulfils the obligation in Provision 1). At a minimum, e-contracting software should have the expressiveness to assess (check and monitor) Scenario A above. Taking this as our basis for understanding contract assessment, let us now investigate an experimental setup that will exercise e-contract monitoring approaches. The methodology and metrics used are as follows:

3.1 Methodology

Our intention is to exercise conflict detectors and violation monitors quite strenuously by generating a scenario set containing hundreds of intermingled variants of the `provisions` and `facts` (and hence conflicts and violations) in `Scenario A` above. We make use of a biased quasi-random provision, occurrence (i.e. fact), and entity generator in order to achieve this; this creates an interesting mix of provisions, and a good spread of occurrence and entity types, with some common types predominating, as would be expected in real business scenarios. The randomizer ensures that:

Approximately 50% of provisions are obligations, and 50% are prohibitions. Obligations are generated following a general schema for obligations and prohibitions. That is, obligations are "`to (some occurrence) (some amount) to (some participant)`" (an instance of this schema would be the obligation in `Provision 1` above: "`to pay $25,000 to`

`Steelmans`"). Similarly, prohibitions are constructed following the template "`against (more-than/less-than) (some amount) (some role) to (some role)`", which builds instances like the prohibition in `Provision 2` above: "`against more-than $10,000 paid to a supplier`". Approximately 50% of comparisons are `more-than` and 50% are `less-than`.

Approximately 20% of workflow occurrences (i.e. facts) are one of 2 types (`being_supplier` as in `Fact 1` above, and `paying` as in `Fact 2` above), 20% of occurrences are of 4 types, a further 20% of occurrence are one of 4 additional types, another 20% of occurrences are also one of 4 further types, and the final 20% of occurrences are one of a million types.

Approximately 20% of participants are one of 2 individuals (`SkyHi` and `Steelmans` from `Scenario A` above), 20% of participants are one of 4 individuals, a further 20% of participants are one of 4 additional individuals, another 20% of participants are also one of 4 further individuals, and the final 20% of participants are one of a million individuals.

Numbers (`amounts`) are arbitrarily chosen floating-point numbers, between `0` and `1,000,000`.

3.2 Metrics

For our experiments we need to record:

Time

We need to record both the time taken to insert and conflict-check *provisions* in seconds, and the time taken to insert and monitor *occurrences*, in seconds. Varying occurrence-insertion batch sizes (e.g. batch-size 1 vs. batch-size 50) could be measured.

Space

We need to determine the space used by the contract assessment engine for storing *provisions* and *occurrences*, recorded as size-on-disk in megabytes.

Conflicts

We should note the number of *conflicts* detected between individual obligations and prohibitions. This is used to show that the assessment engine is *effective* (and not only time and space *efficient*).

4 Tests and Results

Our contract consistency checking and monitoring prototype is called EDEE. EDEE's implementation and architecture are detailed in our earlier work [1, 2]. Contractual provisions are appendable on a provision-by-provision basis, to support a fine granularity of update. A coverage-checking component [3] looks for conflicting contractual provisions (contract *checking*) and flags violations (contract *monitoring*). Following is the exact hardware and software configuration employed, and the results of our tests.

4.1 Experimental Setup: Hardware and Software

Our algorithm was implemented in Java, and we undertook multiple runs on diverse platforms. Table 1 provides the detailed hardware and software specifications of the machines used for our tests.

Machine Name	Operating System	Database	Java	CPU(s)	Memory
Teme	Windows 2000 Pro	Microsoft Access 2000	1.3.0	800 Mhz AMD Athlon	256 MB
Citadel	Windows XP Pro	Microsoft Access 2002	1.4.0 (_01)	500 Mhz Pentium III	256 MB
Jetset	Windows 2000	Microsoft Access 2000	1.3.1	500 Mhz Pentium III	256 MB
All Windows platforms employed Sun's JDBC-ODBC driver, included with their respective Java distributions.					
Flute	Red Hat Linux 7.2	PostgreSQL 7.2.1	1.4.0 (_01)	1.4 Ghz AMD Athlon	512 MB
hot-spare-00 (elbe)	Red Hat Linux 7.1	PostgreSQL 7.0.3	1.4.0 (_01)	2 x 1.4 Ghz AMD Athlon	2.5 GB
hot-spare-01 (nidd)	Red Hat Linux 7.1	PostgreSQL 7.0.3	1.4.0 (_01)	2 x 1.4 Ghz AMD Athlon	512 MB
hot-spare-02 (loire)	Red Hat Linux 7.2	PostgreSQL 7.2.1	1.4.0 (_01)	2 x 1.4 Ghz AMD Athlon	882 MB
hot-spare-03 (lyd)	Red Hat Linux 7.1	PostgreSQL 7.0.3	1.4.0 (_01)	1.4 Ghz AMD Athlon	878 MB
Gargantubrain	Red Hat Linux 7.1	PostgreSQL 7.1.3-3	1.4.1 beta	4 x 800 Mhz Itanium	16 GB
All Unix platforms employed the Postgres JDBC driver that is included with the PostgreSQL 7.1 distribution.					

Table 1: Specifications of machines used for experiments

The same seed was used to propagate the random generator (§3.1) in all cases, aside from the tests on the machines `citadel` and `jetset` where a different seed was used for variety. For researchers wishing to replicate the experiments and do a comparison to their own work, the EDEE source code and parameters used for the experiments, and the detailed results from EDEE execution on our platforms, can be obtained from http://www.cl.cam.ac.uk/~asa28/.

4.2 Raw Results

Following is a synopsis of EDEE's results, using the metrics proposed in Section 3.2 (see also [1]):

Time

Figure 1 gives the *average* time in seconds per provision. All results here pertain to a batch-size of 1. Figure 2 shows the *average* time to input occurrences, as the number of provisions varies, for the best-performing machine, `teme`. Figure 3 shows the average time to insert occurrences with 251 stored provisions, comparing batch-size 1 to batch-size 50.

Space

We periodically recorded size-on-disk for the Microsoft Access databases and found: 0.25MB for an empty database, 0.5MB for 10 provisions and 10 occurrences, and 95MB for 351 provisions and 200 occurrences.

Conflicts

Figure 4 shows the total conflicts detected as the number of provisions and occurrences vary.

4.3 Analysis of Results

We have shown for the first time that contract performance assessment and dynamic validation is viable on medium-scale problems with hundreds of provisions, small event histories, and a high proportion of run-time conflicts. Figure 4 shows that EDEE successfully detects provisions coming into conflict with each other at run-time as new occurrences are added.

On the best performing machine, `teme`, it took an average of 280 seconds (almost 5 minutes) to individually insert and coverage-check a provision, when there were 351 provisions in the database. Similarly, it took an average of 136 seconds (just over 2 minutes) to insert and coverage-check an occurrence, for 351 provisions and 500 stored occurrences. For less-conflicting sets of provisions, performance is likely to be substantially better than this, as the main overhead is the storage of overlap relations between provisions.

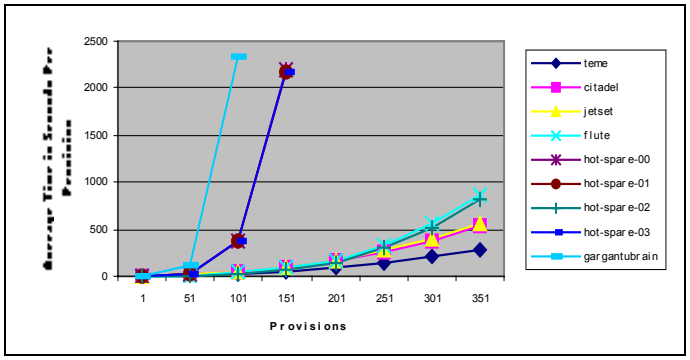

Figure 1: Average time, in seconds per provision, to insert and conflict-check *provisions*

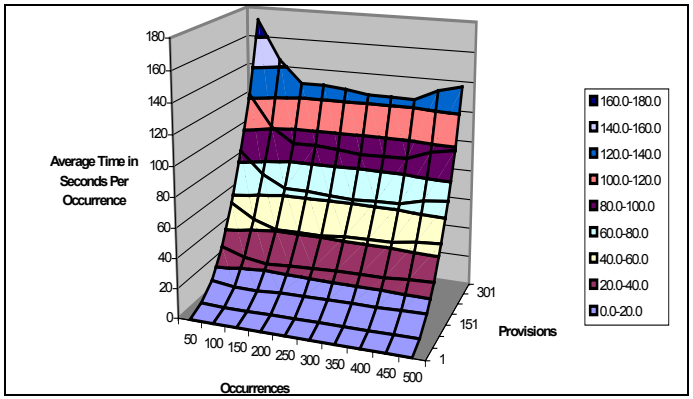

Figure 2: Average time, in seconds per occurrence, to insert and assess *occurrences*

As expected, Figure 3 demonstrates that large batch sizes improve performance, though this leads to a lag in the detection of conflicts, when compared to coverage-checking of each provision or occurrence individually.

5 Conclusion

The eventual goal for e-contracting engines should be to assess tens of thousands of occurrences against thousands of provisions within milliseconds. Clearly the

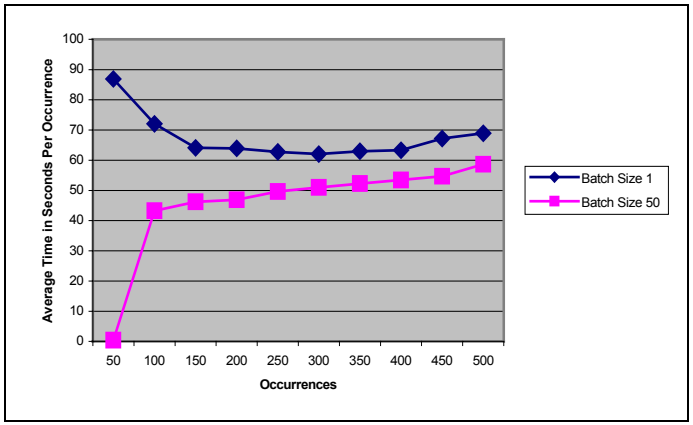

Figure 3: Average time, in seconds per occurrence, to insert and assess *occurrences*

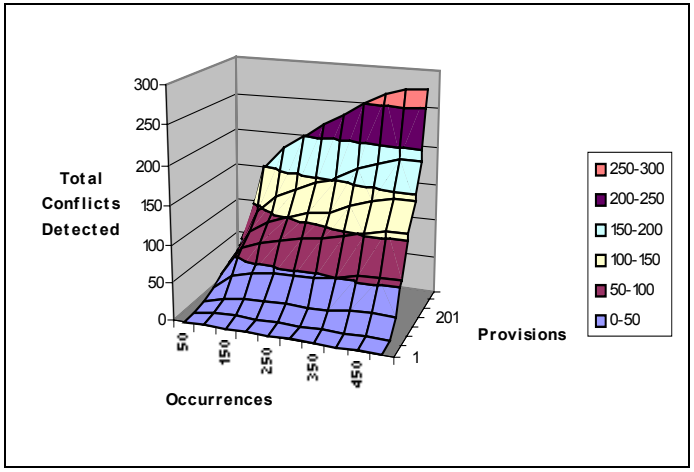

Figure 4: Total conflicts detected between individual prohibitions and obligations

current EDEE implementation is a long way from this target. However, we see significant opportunities for improving EDEE's performance on contract assessment tasks. Firstly, implementation of a native active database layer within the database kernel should yield significantly better performance, and will need to be examined. Mechanisms for reducing the storage space requirement for overlap relations between provisions should be investigated. A possibility is to tag transitively related data with sequence numbers as the data is added to the database; this would allow faster determination of transitive closure. We will need to reconsider the triple store (vertical schema) used by EDEE [1, 2, 3], in search of a more efficient implementation with appropriate data structures. Alternatively, special-purpose optimizations to relational database query planners, such as those recently suggested by Wang, Chang, and Padmanabhan for vertical schemas, could potentially reduce query execution time by up to 80% [26]. Further development of EDEE – to improve its efficiency and user interface – is being pursued under the new name, CAMPACE (Cambridge Policy Analysis and Checking Environment). We have recently reimplemented EDEE's conflict-checker for CAMPACE, using a combination of Prolog and C#, instead of Java, to capture and enforce the conflict detection rules. The new implementation achieves a 60-fold performance improvement over the original. Full source code of both our Java (EDEE) and C#/Prolog (CAMPACE) benchmark implementations are freely available at: http://www.cl.cam.ac.uk/~asa28/

As a final remark, our hope is that other implementers of e-contracting engines will rise to the challenge of building software capable of representing and checking the practical scenario and experimental framework of Section 3. The community can then objectively assess software against the benchmark set by the initial EDEE prototype.

6 Acknowledgements

This research was supported by grants from Microsoft Research Cambridge, the Cambridge Commonwealth and Australia Trusts, the Overseas Research Students Scheme (UK), and the University of Cape Town Postgraduate Scholarships Office.

7 References

[1] Abrahams, A.S. "Developing and Executing Electronic Commerce Applications with Occurrences". *PhD Thesis*. University of Cambridge Computer Laboratory. Cambridge, England. (2002).

[2] Abrahams, A.S., and J.M. Bacon. "A Software Implementation of Kimbrough's Disquotation Theory for Representing and Enforcing

Electronic Commerce Contracts". *Group Decision and Negotiations Journal*. 11(6), 487-524. (2002).

[3] Abrahams, A.S., D.M. Eyers, and J.M. Bacon. "A Coverage Determination Mechanism for Checking Business Contracts against Organizational Policies". *3rd VLDB Workshop on Technologies for E-Services (TES'02)*. Hong Kong, China. *Lecture Notes in Computer Science 2444*. Springer-Verlag. Berlin, Germany. 97-106. (2002).

[4] Bacon, J.M., et al. "Generic Support for Distributed Applications". *IEEE Computer*. 33(3), 68-76. (2000).

[5] Blaze Software. "*Blaze Advisor Technical White Paper*". Available from: http://www.blazesoft.com/products/docrequest.html. (2000).

[6] Bons, R.W.H., R.M. Lee, R.W. Wagenaar, and C.D. Wrigley. "Modelling Inter-organizational Trade Procedures Using Documentary Petri Nets". *Proceedings of the 28th Hawaii International Conference on System Sciences*. (1995).

[7] Brant, D.A., T. Grose, B. Lofaso, and D.P. Miranker. "Effects of Database Size on Rule System Performance: Five Case Studies". *Proceedings of the 17th International Conference on Very Large Data Bases*. (1991).

[8] Carzaniga, A., D.S. Rosenblum, and A.L. Wolf. "*Design of a Scalable Event Notification Service: Interface and Architecture*". Technical Report CU-CS-863-98. Department of Computer Science, University of Colorado, Boulder. (1998).

[9] Daskalopulu, A., T. Dimitrakos, and T.S.E. Maibaum. "E-Contract Fulfilment and Agents' Attitudes". *Proceedings ERCIM WG E-Commerce Workshop on the Role of Trust in E-Business*. Zurich. (2001).

[10] Fabret, F., and H.A. Jacobsen, F. Llirbat, J. Pereira, K.A. Ross, and D. Shasha. "Filtering Algorithms and Implementation for Very Fast Publish/Subscribe". *Proceedings of the 2001 SIGMOD Conference*. (2001).

[11] Grefen, P., K. Aberer, Y. Hoffner, and H. Ludwig. "CrossFlow: Crss-Organizational Workflow Management in Dynamic Virtual Enterprises". *International Journal of Computer Systems Science & Engineering*. 15(5), 277-290. (2000).

[12] Greunz, M., B. Schopp, and K. Stanoevska-Slabeva. "Supporting Market Transactions through XML Contracting Containers". *Proceedings of the Sixth Americas Conference on Information Systems (AMCIS 2000)*. Long Beach, CA. (2000).

[13] ILOG Corporation. "*Business Rules*". Available at: http://www.ilog.com/products/rules/. (2001).

[14] International Standards Organization. "*(ISO/IEC JTC1/SC21/WG7). Open Distributed Processing – Reference Model – Part 2: Foundations*". International Standard 10746-2 / ITU-T Recommendation X.902. (1995).

[15] Kafeza, E., D.K.W. Chiu, and I. Kafeza. "View-Based Contracts in an E-Service Cross-Organizational Workflow Environment". *Proceedings of the Second International Workshop on Technologies for E-Services (TES'01)*. Rome, Italy. *Lecture Notes in Computer Science 2193*. Springer-Verlag. Berlin, Germany. 74-88. (2001).

[16] Lee, R.M. "*CANDID: A Logical Calculus for Describing Financial Contracts*". PhD Thesis. Department of Decision Sciences, The Wharton School, University of Pennsylvania. Philadelphia, PA. (1980).

[17] Merz, M. "Electronic Contracting with COSMOS – How to Establish, Negotiate, and Execute Contracts on the Internet". *Proceedings of the 2nd International Enterprise Distributed Object Computing Workshop (EDOC'98)*. IEEE. (1998).

[18] Milosevic, Z. "*Enterprise Aspects of Open Distributed Systems*". PhD Thesis. Department of Computer Science, University of Queensland. 154-248. (1995).

[19] Morciniec, M., M. Salle, and B. Monahan. "Towards Regulating Electronic Communities with Contracts". *2nd Workshop on Norms and Institutions in Multi-Agent Systems, 5th International Conference on Autonomous Agents*. Montreal, Canada. (2001).

[20] Organization for the Advancement of Structured Information Standards (O.A.S.I.S). "*OASIS ebXML Collaboration-Protocol Profile and Agreement Specification Version 2.0*". Available at: http://www.oasis-open.org/committees/ebxml-cppa/documents/ebcpp-2.0.pdf. (2002).

[21] Paton, N.W., and O. Diaz. Active Database Systems. *ACM Computing Surveys*. 31(1), 63-103. (1999).

[22] Peyton-Jones, S., J.M. Eber, and J. Seward. "Composing contracts: An adventure in financial engineering". *International Conference on Functional Programming*. Montreal, Canada. (2000).

[23] Pottinger, R., and A. Levy. "A Scalable Algorithm for Answering Queries using Views". *Proceedings of the 26th International Conference on Very Large Databases (VLDB)*. 484-495. (2000).

[24] Reeves, D.M., B.N. Grosof, and M.P. Wellman. "Automated Negotiation from Declarative Contract Descriptions". *Computational Intelligence*. 18(4), 482-500. (2002).

[25] Sergot, M.J., et al. "The British Nationality Act as a Logic Program". *Communications of the ACM*. 29(5), 370-386. (1986).

[26] Wang, M., Y-C. Chang, and S. Padmanabhan. "Supporting Efficient Parametric Search of E-Commerce Data: a Loosely Coupled Solution". In *Proceedings of the 8th International Conference on Extending Database Technology (EDBT 2002)*. Prague, Czech Republic. Lecture Notes in Computer Science 2287. Springer. 409-426. (2002).

[27] Weigand, H., and W. Hasselbring. "An Extensible Business Communication Language". *International Journal of Cooperative Information Systems*. 10(4), 44-56. (2001).

8 Author Biographies

Alan S. Abrahams is a lecturer in Operations and Information Management at the Wharton School, University of Pennsylvania. Prior to this, he worked as a postdoctoral Research Associate at the University of Cambridge Computer Laboratory. His research interests are contract-driven application development and control, and agent simulation. He holds a PhD from the University of Cambridge, and a Bachelor of Business Science from the University of Cape Town.

David M. Eyers is currently a researcher at the University of Cambridge Computer Laboratory and a member of King's College. He is primarily working on policy-driven distributed access control systems. By the end of 2004, his work focus will shift to a direct involvement with the development of the CamPACE software system. A British EPSRC grant will support this development in collaboration with the Wharton School at the University of Pennsylvania.

Jean M. Bacon leads the Opera research group at the University of Cambridge Computer Laboratory, and is a Reader in Distributed Systems and Fellow of Jesus College Cambridge. She is author of *Concurrent Systems* and, with Tim Harris, *Operating Systems: Concurrent and distributed software design* published by Addison-Wesley. She is also Editor in Chief of the *IEEE Computer Society's Distributed Systems Online* (*DS Online*); see http://dsonline.computer.org. Her current research interests include role-based access control, business contracts, event-based middleware, and policy.