Critical National Need:
A Computational Infrastructure for 21st Century Accounting

Rahul Simha
Department of Computer Science
The George Washington University
Washington, DC 20052

Jagdish Gangolly and **S.S. Ravi**
Department of Management Science & Information Systems,
Department of Computer Science
State University of New York, Albany
Albany, NY 12222

1 Summary

The societal challenge: Today's information systems that support financial accounting are baroque and outdated – they are not only often unreliable and inefficient, but are easily outsmarted by wily perpetrators of fraud. The nation needs a fundamental overhaul of the way accounting systems are built.

Background. In the 13th century AD, the merchants of Venice developed a system of accounting that would both beget them trade advantages and revolutionize the world of commerce. Their system – double-entry bookkeeping – was eventually codified by Luca Pacioli in 1494 and remains the cornerstone of accounting today. At the same time, the burden of accounting has grown significantly as a result of vastly increased complexity in business law, exotic financial instruments, internationalization and the sheer volume of business transactions. Some of the mismatch between accounting tools and what business principals are able to accomplish is exemplified in cases such as the Enron scandal in which a massive auditing effort was needed to identify the real problems artfully hidden among myriad obscuring details. Indeed, the ordinary single-employee embezzlement of yore has now been replaced with *fraud-by-complexity* in which purpose is concealed by routing money flows through a maze of multiple entities. All of this leads to some fundamental questions worth asking: Is the accounting profession making the best use of modern ideas in computing? What should the computational infrastructure for 21st century accounting look like?

To see what might be possible, consider two examples of relatively complex systems that are made manageable through the use of equally sophisticated computational tools. The first example is the modern computer chip: a vast, intricately connected network of a billion units (transistors). Its design is itself a complex process that is managed by a suite of tools, from specialized hardware design languages to automated theorem-provers that verify correctness. The second example is software development: software systems now routinely comprise of millions of lines of code and are increasingly reliant on specialized tools that aid in development and verification.

Research themes. We believe that a computational infrastructure for 21st century accounting should feature:

- Formal models for accounting systems that incorporate numbers, actions, roles and entities as basic primitives;

- New mechanisms for tracking and auditing data while simultaneously ensuring security, privacy and integrity;
- Programming language support aimed solely at accounting needs that will both result in robust systems and also allow rapid, legally sound, resolution of disputes;
- Run-time infrastructures to help track and verify the integrity of multi-party transactions;
- Verification tools using modern automated verification and theorem-proving techniques that have proven successful in other areas.

Impact and transformative potential. The transformative potential arises from opening the door to a rich intersection between computer science and accounting. Today's accounting systems merely use relational databases as a storage mechanism for what is, in essence, traditional bookkeeping. Future accounting systems, we posit, will make use of a full suite of hardware and software tools to manage the complexity of modern accounting. The project also provides new directions for research in computer science, as the accounting application requires new formal models and programming language features, as well as new primitives in run-time support across distributed systems.

The past few years have seen seismic events in the financial world, from the aforementioned Enron scandal to the collapse of global markets. A fundamental rethinking of regulation is already being considered; thus, as part of this process, it makes sense to examine whether powerful new computational tools can help in this evolution of financial systems.

2 Additional Detail

Background: the essence of an accounting system[1]. In a broad sense, the purpose of accounting is to record economic events as they apply to particular entities (organizations or individuals) in a manner that allows post-event examination and assertion of properties pertaining to those events. More informally, since most accounting issues involve numbers (money, financial assets, and quantities of goods), modern accounting addresses two questions: "Who may do what to which number?" and, when something goes wrong and triggers an audit, "Who did what to which number and when?" Thus, at an abstract level there are certain *numbers* of interest that transactions modify and therefore, at a fundamental level, any computational infrastructure for accounting must support the creation, access and auditing of both the numbers and the people (roles) that modify them.

As an example, consider the *round-trip* or *wash sales* transaction in Figure 1 below where company A arranges to buy and sell the same goods from and to company B during a time-interval that straddles two accounting periods in order to inflate revenues. Often, in fraud cases [8], company B is a front company legally organised as a so-called *Special Purpose Entity (SPE)* – it is noteworthy that some fraud cases have involved at least 4000 such entities [6].

The right side of the figure shows a simple buy-sell transaction across two companies, C and D. Today, there is no way to correlate these transactions. Auditors of company C can only examine C's accounts and receipts (including those issued by D) but cannot correlate them with an audit of D unless a formal investigation of both companies occurs through litigation. Understandably, companies are skittish about allowing their access to their proprietary data using a common system. Thus, future systems should enable cross-entity tracking and verification while also preserving sufficient privacy.

[1]Note: We have attempted to write for a general technical audience. Accordingly, we have simplified some of the accounting terminology for ease of exposition, hopefully, without causing offense to disciplinary experts.

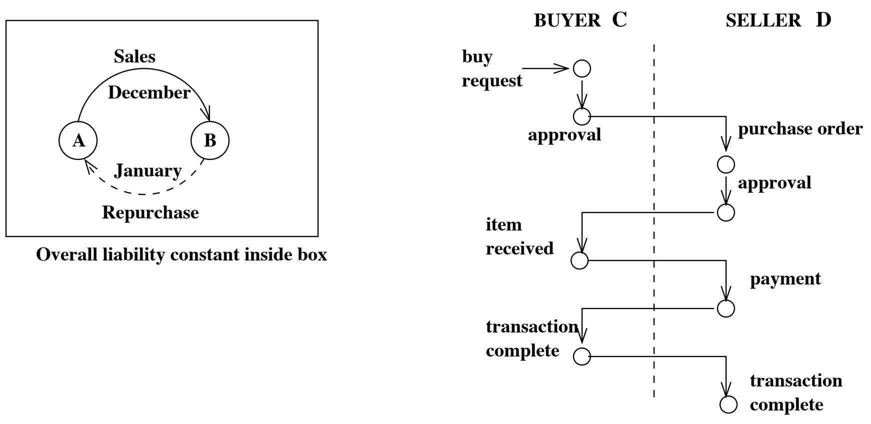

Figure 1: Two examples: (1) A model of one of the money-flows in a fraud case; (2) A buy-sell transaction

What are today's accounting systems like? Current systems can be understood as a combination of centuries-old double-entry bookkeeping, forced into a relational database system to which middleware (application) complexity is continually added as and when rules and reporting requirements change. Thus, the "numbers" are stored as entries in relational tables, rules are written in proprietary database systems particular to each organization, from which complex reports are generated for auditors to fathom and certify. In many organizations, the standard is even lower as the "numbers" live in Excel spreadsheets that must then be absorbed into a larger system for aggregration.

What's wrong with today's infrastructure? In some sense, today's systems are crawling along and the world continues to turn. To see why we might want to overhaul this "working" system, consider the following analogy with database systems. Prior to relational systems, databases were entirely proprietary and built from scratch, and as complexity grew, became increasingly unwieldy and resource-hungry. The advent of relational databases resulted in a formal model with high-level abstractions (the relational algebra), a standardized programming language (SQL), automated verification (keys, foreign keys, views) and a focus on efficient implementation across a distributed system (client-server model). With these features it became feasible to build systems of far greater *complexity* that also have greater reliability and efficiency than those earlier generation systems.

To see the potential present in overhauling this working system, one may draw an analogy with the primitive approach to programming in the early days of computing when programs were written in hexadecimal or assembly – those systems also "worked" and were considered adequate to the task at the time. However, a half-century of advances in computing have shown that with higher-level abstractions in programming languages and accompanying system support, Furthermore, some of these abstractions have lent themselves to formal modeling which by itself has led to sophisticated tools for program correctness and verification that are today an integral part of developing large applications.

Thus, in this vein, we identify the following shortcomings in today's accounting systems:

- *Lack of formal models and useful abstractions.* Much of the intellectual discourse in the accounting literature focuses on the roles of individuals within an organization, on categorizing numbers (judging whether a cost is permissible), interpretation of laws and regulations, and on economic consequences (what is the impact of this interpretation or that). This literature typically assumes that a perfect system of monitoring and auditing is in place, run by professional auditors. Although some formal thinking about bookkeeping exists [17], there is as yet no formal model similar to, for example, the relational algebra for databases.
- *Lack of support for cross-entity or within-hierarchy operations.* A basic auditing need is to assert that an increase in revenue of company A that sold something to company B is reflected as a cost in

company B. Yet, today, no system is in place to automatically assert cross-organization properties of this kind.

- *Lack of accounting-specific programming language features*. Today's accounting rules and regulations are implemented in middleware, the layer of code that lies between a relational database and the user. This code is almost always proprietary and is often developed using proprietary software. However, such code is difficult to maintain, even more difficult to analyze for correctness, and furthermore, lacks independent verification that such code is working correctly.
- *Lack of system support for real-time monitoring*. Most auditing occurs after the fact. What is desirable instead is a system that provides a "dashboard" of indicators for auditors that is updated in real-time.
- *Lack of tools for efficient auditing and post-event querying*. Once an audit occurs, a great deal of effort is expended on examining the recorded data and correlating events to assert propriety. There are no tools to assist in auditing other than reports generated by the same complex middleware performing the transactions.
- *Lack of tools to manage complexity*. The complexity in transactions today is directly embedded in the abovementioned middleware and thus, any attempt to make changes must contend with the functioning of such middleware. But this middleware is itself written by hand (often by contract programmers), is poorly documented and constitutes a major cost in building today's IT systems. There is a need for tools that provide verification both at design time (for prevention) and at runtime (for detection).

Can these features be easily incorporated in today's systems? Naturally, one wonders whether the above desired features (aside from the formal model) can be directly implemented in today's systems. Certainly, middleware programmers can be directed to implement almost any feature for their proprietary system just as any database query could have been implemented in early systems. However, this approach merely increases the complexity burden – what is desirable, instead, is to develop higher-level abstractions that are formally verified and implemented in programming languages.

What is novel about the proposed area of research? To some extent, it is possible and desirable to import and refine a good many ideas from the computer science literature that apply to accounting. In particular, there is now a body of work on secure workflows [3, 21, 22], on formal access models [10, 16, 20, 25], and related languages [11, 15, 19, 26]. Some of these core ideas could carry over into the accounting domain, even if the details are different. At the same time, there is a need for specific and practical formal models that apply to financial accounting and entirely new programming language features directed at specific needs in accounting.

Why should the NIST TIP program fund this type of research? The Department of Commerce would appear to be the right place to consider an overhaul of the nation's approach to financial systems. According to the TIP criteria:

- *Nation's capabilities*. It is clear that reliable, trustworthy and efficient accounting systems will ease the burden of accounting on business, will help build trust in the nation's financial system, and will increase the efficiency of markets.
- *Other sources of funding*. This type of work does not appear to fit into the mission of the other funding agencies.

The most important reason is that this is a critical need that companies are loath to admit, for which there are few academic initiatives, and *nobody else is doing anything about it.*

References Cited

[1] J.R.Alexander. History of accounting, *Association of Chartered Accountants in the United States*, 2002.

[2] K.Apt and M.Wallace. *Constraint Logic Programming using ECLiPSe*, Cambridge University Press, 2006.

[3] V. Atluri. Security for workflow systems, *Information Security Technical Report*, Volume 6, Number 2, 2001, Elsevier Science, pages 59-68.

[4] R.W.H.Bons, F.Dignum, R.M.Lee and Y-H.Tan. A formal specification of automated auditing of trustworthy trade procedures for open electronic commerce. *Proc. 32nd Hawaii Int.Conf.System Sci.*, 1999.

[5] C.Cachin and A.Samar. Secure distributed DNS, *Int. Conf. Dependable Systems and Networks*, 2004.

[6] H.Carpenter. Special-purpose entities: a description of the now loathed corporate financing tool, *Missisipi Law Journal*, 2003.

[7] R.O.Duda, P.E.Hart and D.G.Stork. *Pattern Classification*, Wiley, 2002.

[8] Dynegy settles securities fraud charges involving SPEs, round-trip energy trades. *U.S. Securities and Exchange Commission*, press-release 2002-140, `http://www.sec.gov/news/press/2002-140.htm`, 2002.

[9] P.Elsas. *Computational Auditing*, published by Vrije University and Deloitte Consulting, 1996. The thesis received the Alfred Coini Prize for the best publication in auditing in 1996.

[10] D.Ferraiolo, D.R.Kuhn and R.Chandramouli. *Role-based Access Control*, Artech House, 2003.

[11] M.Gaiman, R.Simha and B.Narahari. Privacy preserving programming using Sython, *J. Computers and Security (COSE)*, Vol. 26, 2007, pp. 130-136.

[12] J.S.Gangolly. On formal modeling of accounting information systems. Keynote address, *Int. Conf. Digital Accounting Research*, Univ. Huelva, Spain, 2004.

[13] J.Gray and A.Reuters. *Transaction Processing: Concepts and Techniques*, Morgan-Kaufman, 1993.

[14] D.Gusfield, *Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology*, Cambridge University Press, 1997.

[15] N.Habra and I.Mathieu. ASAX: Software architecture and rule-based language for universal audit trail analysis, *European Symposium on Resarch in Computer Security*, November 23-25 Toulouse, 1992.

[16] C.Hanson, T.Berners-Lee, L.Kagal, G.J.Sussman and D.Weitzner. Data-purpose algebra: modeling data usage policies. *8th Int.Workshop on Policies for Distributed Systems and Networks*, 2007.

[17] Y-H.Hyon and S-W.Park. A new two-dimensional double-entry bookkeeping system and three-dimensional accounting, *2nd Asia-Pacific Interdis. Research. Accounting (APIRA)*, August 1998, Osaka, Japan.

[18] A Universally Unique IDentifier (UUID) URN Namespace, *IETF RFC 4122*.

[19] P.Kumaraguru, L.F.Cranor, J.Lobo and S.Calo. A survey of privacy policy languages, *Symp. Usable Security and Policy*, 2007.

[20] N.Li and Q.Wang. Beyond separation of duty: an algebra for specifying high-level security policies, *ACM Conf. Computer and Communications Security (CCS)*, November 2006.

[21] F.Montagut, R.Molva, and S.Tecumseh. The pervasive workflow: a decentralized workflow system supporting long running transactions, *IEEE Transactions on Systems, Man and Cybernetics*, Volume 38 No.3, May 2008, pp 319-333.

[22] F.Montagut and R.Molva. Traceability and integrity of execution in distributed workflow management systems, *12th European Symposium On Research In Computer Security*, September 24-26, 2007, Dresden, Germany, pp 251-266.

[23] E. Ould-Ahmed-Vall, D.Blough, B.Heck, and G.Riley. Distributed unique global ID assignment for sensor networks, *Proc. 2nd IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems*, pp. 573-580, 2005.

[24] C.R.Palmer and G.V.Cormack. Operation transforms for a distributed shared spreadsheet. *Proc. Computer Supported Cooperative Work (CSCW'98)*, Seattle, WA, 1998.

[25] R.Sandhu, D.F.Ferraiolo and D.R.Kuhn. The NIST model for role based access control: toward a unified standard. *5th ACM Workshop Role-Based Access Control*, pp.47-63, 2000.

[26] Transforming financial information – use of XBRL in federal financial management. *American Council for Technology*, 2007.