Comet:
An Application of Model-Based Reasoning to Accounting Systems

Robert Nado, Melanie Chams, Jeff Delisio, and Walter Hamscher

Price Waterhouse Technology Centre
68 Willow Road
Menlo Park, CA 94025-3669
{nado chams delisio hamscher}@tc.pw.com

Abstract

An important problem faced by auditors is gauging how much reliance can be placed on the accounting systems that process millions of transactions to produce the numbers summarized in a company's financial statements. Accounting systems contain *internal controls*, procedures designed to detect and correct errors and irregularities that may occur in the processing of transactions. In a complex accounting system, it can be an extremely difficult task for the auditor to anticipate the possible errors that can occur and to evaluate the effectiveness of the controls at detecting them. An accurate analysis must take into account the unique features of each company's business processes. To cope with this complexity and variability, the Comet system applies a model-based reasoning approach to the analysis of accounting systems and their controls. An auditor uses Comet to create a hierarchical flowchart model that describes the intended processing of business transactions by an accounting system and the operation of its controls. Comet uses the constructed model to automatically analyze the effectiveness of the controls in detecting potential errors. Price Waterhouse auditors have used Comet on a variety of real audits in several countries around the world.

Auditors have the task of determining whether the financial statements of a company are a fair presentation of the company's financial position. An important problem faced by auditors is gauging how much reliance can be placed on the accounting systems that produce the numbers summarized in the financial statements. Accounting systems contain *internal controls*, procedures designed to detect and correct errors and irregularities that may occur in the processing of transactions. In a complex accounting system, it can be an extremely difficult task for the auditor to anticipate the possible errors that can occur, to determine their downstream effects in the accounting system, and to evaluate the effectiveness of the controls at detecting them. An accurate analysis must take into account the unique features of each company's business processes. To cope with this complexity and variability, the Comet system applies a model-based reasoning approach (cf. Hamscher *et al.*, 1992) to the analysis of accounting systems and their controls.

Comet supports the creation of hierarchical flowcharts that ultimately describe the processing of business transactions in terms of a set of primitive activities for operating on records and a set of controls for detecting and correcting errors that may occur in the processing. Using knowledge of the basic ways in which the primitive activities can fail, Comet finds potential failures that can occur in the accounting system and uses the structure of the flowchart to analyze the impact of those failures on the validity of the accounts. Comet then matches each potential failure to the set of controls capable of detecting it and evaluates the effectiveness of the controls in reducing the risk that the potential failure will go undetected. Finally, Comet ranks the controls with respect to their relative contribution to reducing the risk of undetected failures and selects a subset of key controls whose proper operation should be tested.

Task Description

In the United States, the SEC requires a yearly independent audit of the financial statements of public companies. Other countries have similar requirements. An accounting firm that is engaged to perform an audit of a public company has the task of issuing an opinion on whether the financial statements are a fair characterization of the financial position of the company and follow generally accepted accounting principles. The numbers that appear in the financial statements are typically the accumulated results of thousands, even millions, of detailed financial transactions in which the company has participated over the previous year.

There are two main approaches that can be taken to assessing the accuracy of financial statements. The *substantive* approach attempts to obtain evidence of the validity of financial statements by examining records of detailed transactions and applying analytical methods to gauge the reasonableness of the reported numbers. By contrast, the *systems-reliant* approach focuses not on

verifying the numbers themselves but on assessing the adequacy of the accounting systems that produced the numbers. In taking a systems-reliant approach, an auditor looks at the internal controls that are in place in the accounting systems and evaluates their effectiveness in detecting and correcting errors that may occur in processing transactions.

For example, a company's "purchases and payables" system handles transactions involving the purchase of goods from suppliers. Such a system is designed to receive and record purchase orders, transmit them to suppliers, ensure that goods are received, payables recorded, and the supplier eventually paid for goods received. In auditing such a system, it is important to focus not so much on the computer system itself but on the business processes which it supports. A business process usually contains both manual and computerized steps and is partially performed by parties outside the company.

There are many things that can go wrong in a purchases and payables system. For example:

- An invoice may be received from a supplier for goods which were never ordered or received.
- The quantity or price of goods listed on the invoice may be incorrect, either due to an error at the supplier or because of an operator error in entering the invoice into the computer system.
- A fictional invoice may be entered into the system as part of an attempt to defraud.

In order to detect and correct such problems, a purchases and payables system should contain a number of internal controls. For example:

- Invoices that have been entered on to the computer system should be matched to corresponding purchase orders and records of goods received, with quantities and prices agreed. Although the matching process can be computerized, any discrepancies will generally need to be manually investigated and resolved.
- Access to the computer system for data entry should be restricted to authorized personnel by means of an appropriate security system.
- Data entry of an invoice should not be performed by the same person who later authorizes or reviews the invoice.

In practice, any given audit will combine elements of both the substantive and system-reliant approaches with the relative emphasis dependent on the particular characteristics of the business and its components. With large companies that have complex, computerized accounting systems processing vast numbers of transactions, the systems-reliant approach is becoming increasingly important, both to obtain adequate audit evidence and to reduce the cost of the audit. A specialized category of auditor, called a CIS (Computerized Information Systems) auditor[1], brings to bear skills in both accounting and systems analysis to carry out a systems-reliant audit approach.

In order to take a systems-reliant approach, a CIS auditor must obtain and document an understanding of how an accounting system processes business transactions and of the internal controls that are in place. In preparing this "model", the auditor may make use of available systems documentation from the client. However, systems documentation generally is not prepared from an audit point of view. It may explain how the system works in great detail, but generally does not contain adequate information on controls, does not have a business process focus, and omits the manual components of the business process. The auditor must supplement information obtained from documentation with observation of the system in operation and interviews with key personnel.

In determining the effectiveness of controls, it is important to distinguish the role of a control in the design of an accounting system from how well it is performed in practice. By analyzing the processes and data flows of an accounting system, an auditor attempts to determine those controls that play key roles in the prevention and detection of errors that may affect the validity of the financial statements. In order to obtain sufficient comfort that the system is actually operating as designed, the key controls need to be tested to ensure that they are being properly performed.

For complex accounting systems, a thorough and accurate controls evaluation is almost impossible to perform efficiently without some form of computer-based support. There are many different possible sources of error, some of which may be overlooked. It is extremely difficult to manually trace the effects of possible errors through the transaction processing to determine whether they are significant to the audit. There may be redundancy in the coverage of errors by controls, but detailed analysis is required to determine this with confidence. Because systems evolve rapidly, it is costly to determine the impact of system changes on controls effectiveness. Most importantly, human fallibility in the face of complex systems can lead to costly consequences.

Prior to Comet, CIS auditors have used a combination of flowcharting software and controls checklist software in their evaluation of controls. Commercial flowcharting software can be used to document major activities carried out in an accounting system but the result is not in a form that allows automated analysis. Checklist software is populated with libraries of controls that could be expected to be found in a client's system to address the major areas of risk. Although different libraries of controls can be

1. Variously called an EDP (Electronic Data Processing) auditor, or an ISRM (Information Systems Risk Management) auditor

developed for the major components of generic accounting systems as well as for different accounting software packages, it is difficult to tailor checklist software to reflect the varying characteristics of different industry sectors and the idiosyncratic aspects of a particular client's implementation. Furthermore, controls checklist software takes no advantage of the information captured in flowchart documentation.

The development of Comet was motivated by the intuition that an accounting system can be hierarchically decomposed into a structure that bottoms out in instances of a small set of primitive types of actions for processing records and for implementing internal control. Provided that the behavior of the primitive activity and control types can be suitably characterized, a model-based approach can be taken to the analysis of failures and their detection by internal controls. As a consequence, the auditor can concentrate on developing an accurate model of the accounting system under review, with Comet automating the more burdensome aspects of controls evaluation.

Application Description

Although model-based reasoning has been previously applied to financial domains, the models have generally consisted of equations and constraints representing the relationships between financial and microeconomic quantities [Bouwman, 1983, Hart *et al.*, 1986, Bridgeland, 1990, Hamscher, 1994]. Comet is novel in its application of a model-based approach to analyzing systems for processing financial records.

Basic Modeling Concepts

Accounting systems process records of business transactions through activities that create, use, alter, and store those records. Comet represents the processing performed by an accounting system as a hierarchically structured flowchart graph. The two most important kinds of nodes in a Comet flowchart are collection nodes and activity nodes. Collection nodes represent repositories of records, which may be in either paper or electronic form. Activities are represented hierarchically, starting with nodes representing activities at a high-level of abstraction and progressively decomposing them until nodes representing primitive activities are obtained.

Figure 1 shows the top-level flowchart of PURCHASE, a model of a simple Purchases and Payables accounting system. The top-level flowchart is intended to give a high-level overview of the system, indicating the major activities performed by the system, the relevant general ledger accounts, and important collections of records that are accessed and updated by the processing of a transaction. Activity nodes are distinguished by having a rectangular icon in their lower-left corner. Collection

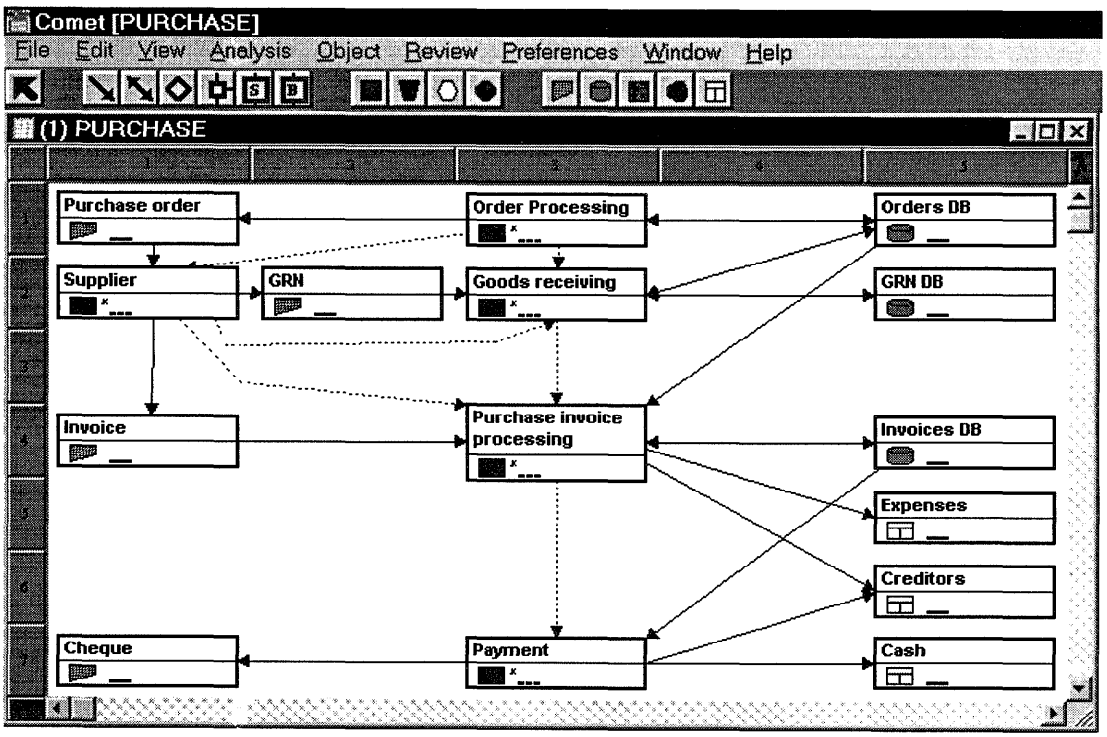

Figure 1: A Top-Level Flowchart

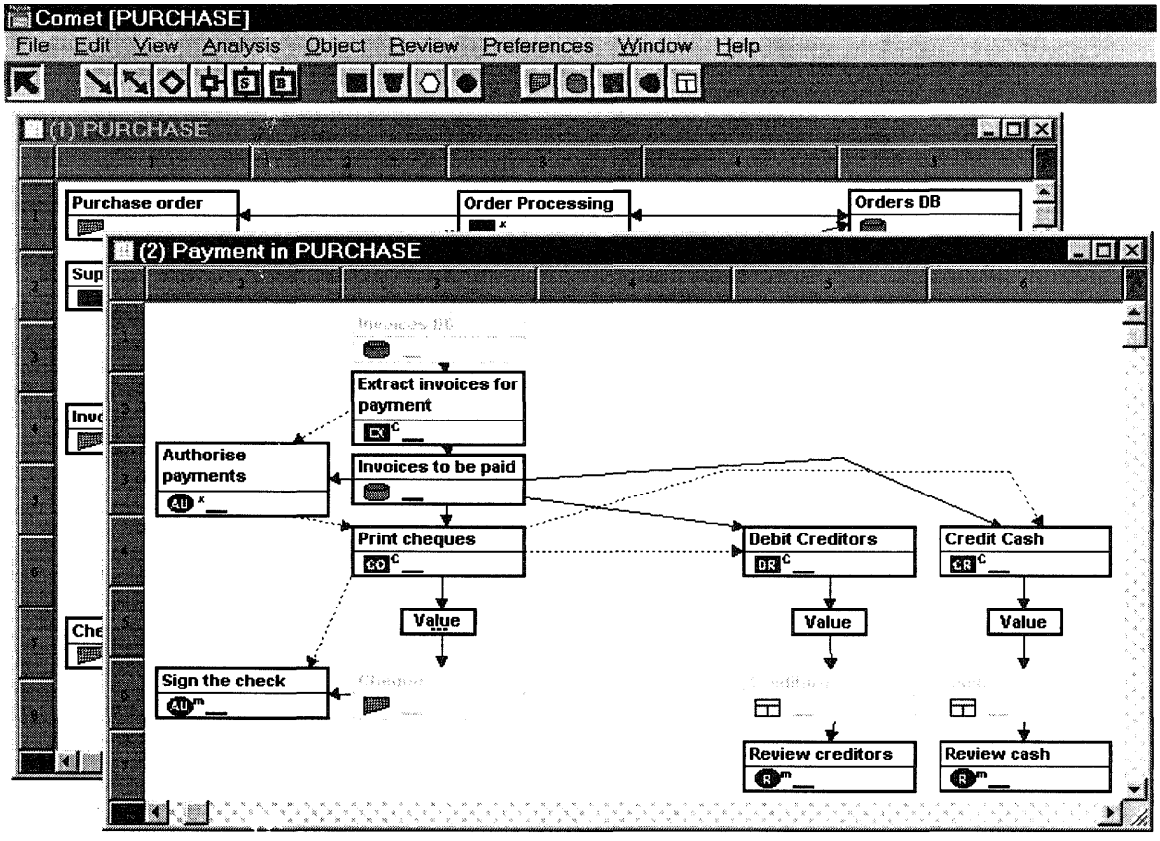

Figure 2: Expansion of the Payment Activity

nodes have a trapezoidal icon for paper records and a cylindrical icon for electronic records. Nodes representing general ledger accounts contain a "boxed T" icon. There are two kinds of arcs connecting nodes in the flowchart. The solid arcs represent data flow relationships between activities and collections. The dashed arcs represent precedence relationships between activities; the activity at the tail of a dashed arc must be completed before the activity at the head of the arc can proceed.

The *Order Processing* activity prepares a *Purchase order*, which is sent off to a *Supplier* to be filled and also recorded in the *Orders DB*. When the *Supplier* fills the order it sends a goods received note (GRN) and an invoice along with the goods. The *Goods receiving* activity records the GRN in the *GRN DB* and tries to match it up with a corresponding record in the *Orders DB*. The *Purchase invoice* activity records the invoice from the Supplier in the *Invoices DB* and compares it with the corresponding record on the *Orders DB*. If a matching order can be found, the *Purchase invoice* activity posts a credit to the *Creditors* account and a debit to the *Expenses* account. The *Payment* activity periodically extracts invoices that are due for payment, prepares checks for payment to suppliers, debits the *Creditors* account, and credits the *Cash* account.

Since the top-level flowchart of PURCHASE gives a high-level overview of the system, it contains no primitive activities or controls. Each of the top-level activity nodes has a decomposition into a sub-flowchart that gives more detail about how that activity is performed. Figure 2 shows the flowchart for the decomposition of the *Payment* activity; it may be reached by double-clicking on the *Payment* node in the top-level flowchart. The nodes in Figure 2 that have dashed light-gray borders are called reference nodes; they refer to collections whose primary depiction is elsewhere in the flowchart. When an activity node is decomposed, each collection node to which it is directly connected has a reference node automatically created in the sub-flowchart. The reference nodes allow the input and output collections of the top-level activity to be referenced by the activities in the sub-flowchart.

Comet contains a predefined vocabulary of activity and control types, called *verbs*, that are used as a focal point for organizing the knowledge that Comet contains about accounting systems and their controls. Some verbs, such as *transfer*, *copy*, *create*, *merge*, *find*, *compute*, and *copy-field*, represent typical operations on records that are

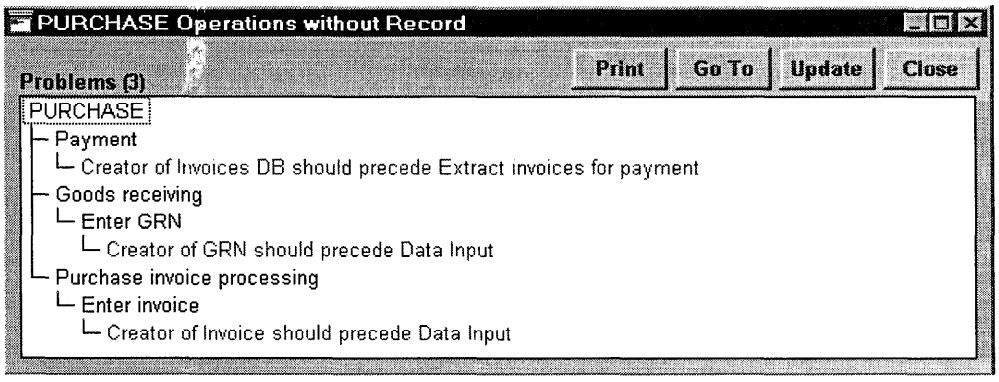

Figure 3: Example Review Dialog Box

treated as primitive by Comet. Other verbs, such as *maintain-standing-data* and *data-entry*, represent processing patterns that are common enough that Comet provides automatic decompositions for nodes using those verbs. For describing internal controls, Comet provides a set of control verbs, including *authorize*, *compare-agree*, *grant-access*, *reconcile*, and *review*. The verb associated with an activity or control node is indicated in the display of that node using a one or two letter code inside the icon in the lower-left corner.

Figure 2 contains four primitive activity nodes with the verbs *extract* (EX), *copy* (CO), *debit* (DR) and *credit* (CR). Figure 2 also contains four control nodes (the nodes with the circle icon) using two different control verbs, *authorize* (AU) and *review* (R). In addition to nodes representing collections, activities, and controls, Figure 2 contains three smaller, rectangular nodes, called selectors. Selector nodes are used to indicate the fields of records that are accessed or modified by activities. For example, the selector node between the *Debit Creditors* activity and the *Creditors* account indicates that the debits involve a field called *Value*.

Model-building Support

The analysis performed by Comet depends for its validity on the accuracy of the models that it operates on. Auditors attempt to verify the accuracy of a model by walking through the transaction processing steps specified in the model, checking for matching steps performed in the modeled system. Ideally, the walkthrough is performed by a person not involved in the model preparation. Although Comet cannot ensure that the models constructed by users are, in fact, accurate representations of the modeled accounting systems, Comet incorporates a number of tools to aid in the construction of models that are at least internally consistent and that contain enough detail to support Comet's analysis.

Each type of node has an associated set of declarative constraints on the ways that a node of that type may be correctly connected by arcs to neighboring nodes. For example, a Credit activity node must have exactly one input collection and at least one output collection. Every output collection must be an account. Finally, there must be selector nodes intervening between the Credit node and each of its output accounts giving the fields that are posted to the accounts. As the user edits a model, Comet monitors the constraints on each node and draws a red flag on those nodes whose constraints are not satisfied. For any node with a red flag, the user may obtain an explanation of the unsatisfied constraints.

Comet contains a number of review commands for examining the completeness and consistency of a model:

- Finding all nodes with violated syntactic constraints
- Finding all unexpanded generic activity nodes
- Finding control nodes that have been incompletely described
- Finding inconsistencies between the fields read from a collection node and the fields written to it
- Finding activity nodes that access records from a collection node without having a preceding activity node that creates records on the collection
- Finding inconsistencies between the inputs and outputs specified for an activity node and for its sub-flowchart

The results of the review commands are presented in the form of dialog boxes that allow convenient navigation to the points where problems occur in the model (cf. Figure 3).

Failure Generation and Propagation

Comet categorizes the errors and irregularities that can occur in an accounting system into three broad categories of failure corresponding to the focus on the processing of

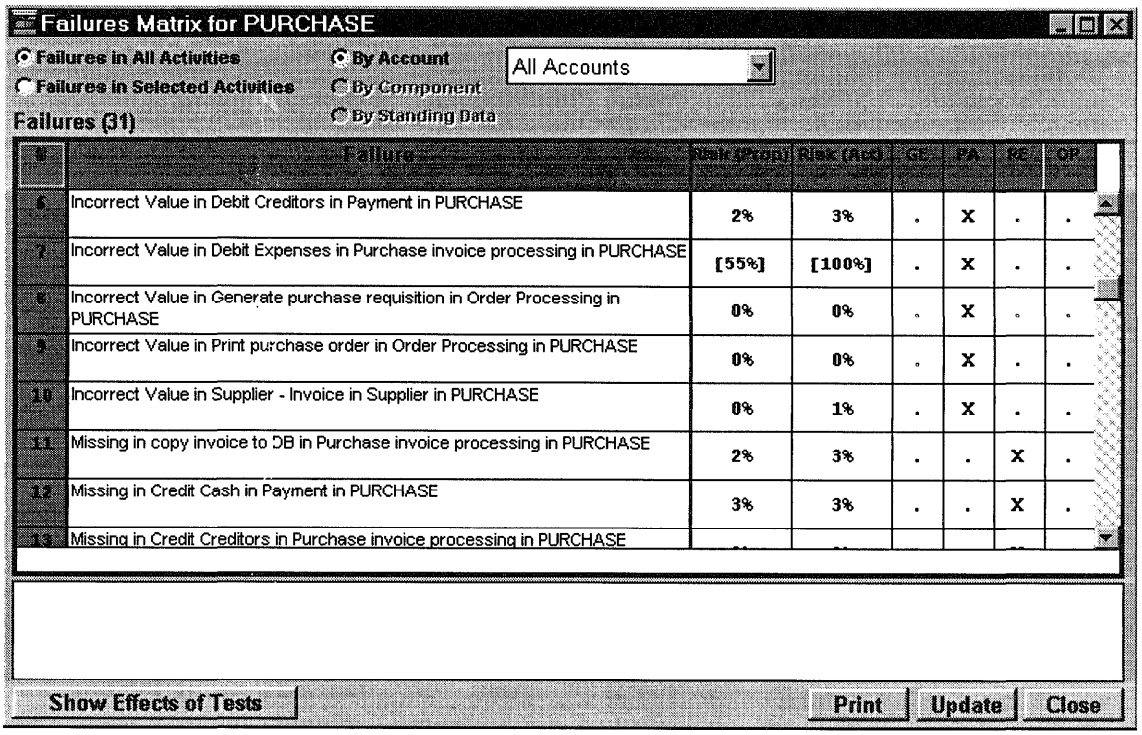

Figure 4: Failure Coverage Risks

records. A *missing* failure occurs when an activity that should have produced a record as output fails to do so. A *spurious* failure occurs when an activity produces an unauthorized or duplicate record as output. Finally, an *incorrect* failure occurs when an activity produces an incorrect value for a field in a record. An incorrect failure is associated with the name of the affected field. Each primitive activity type has associated with it the categories of failure to which it may give rise.

The first stage of Comet's analysis of a model generates the set of potential failures corresponding to each of the primitive activities in the model. Comet then determines which of the potential failures have audit significance. A failure has audit significance if its downstream effects in the flowchart model could cause any of several types of disagreement between the transactions that actually occurred and the way that they are recorded in the accounts. Comet works backwards in the flowchart from the account collections using a few fixed simple rules for the different primitive activity types to determine how failure effects on an output collection may be produced from failure effects on input collections. The result of this stage is to determine for each potential failure the impact that it may have, if any, on the validity of each account collection.

When constructed at a level of detail appropriate to the control evaluation task, a Comet model typically contains on the order of hundreds of primitive activities. Since each of these can fail only in a small number of ways, it is a tractable task to enumerate the set of potential failures and to determine their effects on the validity of accounts.

Control Evaluation

In order to evaluate the controls documented for an accounting system, Comet assesses for each potential failure with audit significance the likelihood that, if it occurs, it will not be detected by any control in the system. We call this likelihood, for a given failure, its *failure coverage risk*. To determine whether the potential failures are adequately covered by detecting controls, a CIS auditor using the system is required to associate with each account an *allowable risk level*. The allowable risk is the highest level of risk the auditor is willing to accept that any failure that occurs and is relevant to the account is not detected by any control.

Figure 4 shows a table generated by Comet of those potential failures generated for the PURCHASE model that have audit significance and the failure coverage risks that have been determined for them. Certain controls in a Comet model may be designated as proposed; proposed controls are used to explore the effects of recommending to the client that additional controls be added to the accounting system to address control weaknesses. The failures table in Figure 4 contains two columns listing failure coverage risks in percentage terms. The first

column (Prop) gives the failure coverage risk taking into account both proposed controls and controls that are actually present in the modeled system; the second column (Act) takes into account only controls that are actually present. If a failure coverage risk is above the allowable risk level for one of the accounts that the failure affects, that failure coverage risk is highlighted by enclosing it in brackets. A failure with a bracketed failure coverage risk indicates a potential control weakness in the accounting system that the CIS auditor should carefully examine.

In determining the failure coverage risk for a failure, Comet first determines the set of controls in the flowchart model that are relevant to the detection of the failure and then assesses, for each relevant control, the likelihood that the control will fail to detect the failure, called the *control detection risk*. The failure coverage risk for a failure is determined by multiplying together the control detection risk for each control that could detect the failure. The risks are multiplied together because we assume that the controls operate independently, and for a failure not to be detected, all of the potentially detecting controls would have to miss it.

In assessing the control detection risk for a given control and potential failure, Comet takes into account three different factors -- control strength, control defeat, and control attenuation:

- Control strength is an assessment of the intrinsic effectiveness of the control, based on its type and how well it is performed. In Comet, the control strength is initially determined from the answers supplied by the modeler to a generic series of questions about how the control is performed. The control strength may be later adjusted as a result of testing the control.
- Control defeat is an assessment of the degree to which a control is rendered ineffective by problems with the maintenance of reference data upon which it depends. For example, a control cannot be relied upon if the maintenance process for a database of information that it employs has potential failures that are not sufficiently mitigated by controls.
- Control attenuation is a measure of the degree to which the effectiveness of a control is reduced by the distance in the flowchart between the control and the primitive activity whose failure it may detect. Control attenuation varies with the type of control and the types of the activities along the path from the control to the failing activity.

Key Controls Selection

A set of key controls is a subset of controls in the model that is sufficient to adequately mitigate the risk of all those potential failures that both have audit significance and are adequately mitigated by the full set of controls. Since placing reliance on a set of controls requires that the controls be tested for proper operation, testing costs can be reduced by choosing a minimal set of key controls. Unfortunately, the problem of finding a minimal set of key controls is a computationally intractable minimal set covering problem. Comet uses a greedy algorithm that works well in practice, but does not guarantee a minimal set.

In selecting a set of key controls for testing, Comet uses a relative measure of the importance of a control in reducing the failure coverage risk of potential failures; this measure is called control contribution. The control contribution for a control is relative to a set of failures, F, to be covered, and a set of controls, C, to be compared. At each point in the selection process, the set F consists of those potential failures whose risk is sufficiently mitigated (with respect to allowable risks) by the complete set of controls in the model, but not yet by those controls already selected for testing. The set C consists of those controls not yet selected for testing. If there are any failures in F that have unique detectors in C with a control detection risk that is less than 1, all these unique detectors are added to the set of key controls. Otherwise, the next control selected for addition to the key controls is that control with the highest control contribution relative to F and C. The algorithm terminates when the set F is empty or there are no controls in C with non-zero control contributions.

Performance

Comet has been successfully used by Price Waterhouse CIS auditors to construct and analyze models of complex client accounting systems. A representative example is a stock trading room system whose Comet model has a total of 934 nodes, including 217 primitive activities, 104 composite activities, 118 collections, and 139 controls. Comet's analysis produced 709 potential failures, of which 338 were found to have an impact on the validity of accounts and 68 were potential defeators of controls. Of these relevant failures, all but 17 were found to be adequately covered by the controls in the system. Comet found 60 controls to be key and therefore candidates for inclusion in a plan for testing controls. The total time required for the analysis was under 30 seconds on a 66Mz Pentium PC.

Application Use and Payoff

A Beta release of Comet has been used on a pilot basis by Price Waterhouse CIS auditors on a variety of real audits in several countries around the world, including Australia, Argentina, Brazil, India, Malaysia, Mexico, the U.S., the U.K., and much of Western Europe. The pilot audits have involved clients from a representative cross-section of different industries, including banking, insurance, oil and gas, manufacturing, and entertainment. The official 1.0 version of Comet was released this April.

The CIS audit partners and managers who have supervised the pilot audits believe, based on their experience, that use of Comet will lead to a significant improvement in auditor productivity. It is difficult at this point to reasonably estimate the size of the gain as a number of factors must be taken into account:

- The nature of the work performed changes with use of Comet. Business processes and their controls are documented to a greater level of detail and more rigorously than they would have been previously. This increases the documentation cost but the analysis performed by Comet allows the auditor to spend much less time anticipating possible errors and thinking about the controls available to detect and correct them. The increased detail and rigor of the models in conjunction with the analysis performed by Comet allows a greater reliance to be placed on controls with a comparable level of auditor effort.
- There is a nontrivial learning curve that applies to efficient use of Comet to model and analyze systems. Experience on the pilot audits suggests that it takes a typical user three to four jobs before they become truly proficient in the use of Comet. Part of what a user needs to learn through experience is the choice of an appropriate level of detail at which to model a system. Enough detail needs to be added to allow a useful Comet analysis to be performed; too much detail adds to the modeling cost without an additional payoff from the analysis.
- The cost of using Comet to model a system and its controls can be more effectively amortized over several years than previous methods of documenting the system. Comet is most appropriately used in a "year of change", either when a new or substantially updated system has been installed by the client or with a new client. In subsequent years, when minor system updates occur, the Comet model can be quickly updated and the impact of the changes on controls' effectiveness analyzed. This justifies somewhat greater initial modeling effort in the year of change as the work that needs to be performed in subsequent years is reduced.
- Use of Comet can reduce the cost of testing. Because of the difficulty of manually performing a thorough and precise evaluation of controls, there is a temptation to perform more detailed testing of transaction records than would be required if the controls work could be done more efficiently. Comet's ability to automatically generate lists of key controls also leads to more focused controls testing, as each control to be tested has been determined to make an important contribution to mitigating the risk of possible failures in the system.
- Comet's rigorous analysis can uncover both control weaknesses and control redundancies, leading to recommendations to the client that are a key value-added function of the audit.

Application Development and Deployment

In 1991, the Savile project was begun at the Price Waterhouse Technology Centre to examine the potential of applying a model-based approach to evaluating accounting systems and their internal controls. An initial prototype, also called Savile, was developed in Lucid Common Lisp running on a UNIX workstation to establish proof of concept. The record processing performed by an accounting system was described using an imperative programming language called SPLAT. Expressions in the SPLAT language were transformed into a causal network to support the evaluation of controls (Hamscher, 1992).

The CIS audit community within Price Waterhouse responded enthusiastically to the Savile prototype and resources were authorized to implement the Savile approach on the standard platform found in Price Waterhouse practice offices -- IBM PC clones running Microsoft Windows. In late 1992, work began on developing a more graphical form of representation for Savile models that would both support a highly interactive flowcharting system and support the analysis of failures and evaluation of controls. Franz Inc's Allegro Common Lisp for Windows was chosen as the implementation language to support rapid application development in the Windows environment.

Since early 1993, an average of three full-time programmers have worked on the development of Comet. In addition, the involvement of CIS auditors was critical to developing a system that matched the requirements of the CIS audit task. A senior CIS manager was assigned to the Price Waterhouse Technology Centre for two months in 1994, two months in 1995, and one month in 1996 to work intensively with the Comet developers to refine the system design.

CIS audit staff have developed a training course in the effective use of Comet in response to increasing worldwide demand. To date, approximately 20% of the total number of CIS auditors in Price Waterhouse firms worldwide have taken the course. In the European firm, all CIS auditors with more than one year of experience are being trained in the use of Comet and it is the recommended tool for use with relatively complex client systems.

Maintenance

As a model-based application, Comet does not contain a large knowledge base encoding expert experience in the domain of CIS audit. This eliminates the often difficult

issues surrounding knowledge base update and maintenance. Rather, the behavior of Comet's analysis engine is a product of the properties of a small set of primitive activity and control types and the structure of the particular accounting system model being analyzed. The set of primitive activity and control types has been remarkably stable over the course of Comet's development and has been found adequate to model a large variety of different client systems encountered during the pilot audits.

After the official release of Comet, responsibility for evolutionary development will transfer from the R&D group in the Price Waterhouse Technology Centre to a Price Waterhouse organization responsible for supporting audit-related software.

Conclusion

Most applications of model-based reasoning have been to engineering domains. Comet applies model-based reasoning techniques to a new task domain, the analysis of the effectiveness of controls in accounting systems. Because of the complexity and variability to be found in realistic accounting systems, CIS auditors have difficulty evaluating controls to the level of detail required to place a high degree of reliance on systems when performing an audit of a company's financial statements. Comet allows a CIS auditor to focus on building a model that accurately describes the accounting system, then makes use of that model to automate the analysis of the adequacy of the controls for detecting potential errors in the system. Demand from the Price Waterhouse CIS audit community for deployment of Comet has been high because it is an effective tool in support of delivering high-quality audits to clients.

Acknowledgments

We would like to acknowledge the contributions of two Price Waterhouse CIS auditors who have been instrumental in the development and deployment of Comet. Pat Russell gave us early guidance on the issues of importance to CIS audit and has been a tireless champion of Comet within the firm. Robert Halliday worked intensively with us on key design issues and has been our day-to-day liaison with the CIS audit practice, giving us numerous suggestions on how to make Comet more useful to the CIS auditor.

References

Bouwman, M. J. 1983. Human diagnostic reasoning by computer: An illustration from financial analysis. *Management Science*, 29(6):653-672.

Bridgeland, D. M. 1990. Three qualitative simulation extensions for supporting economics models. In Proceedings of the Sixth Conference on Artificial Intelligence Applications, 266-273. Los Alamitos, Calif.: IEEE Computer Society Press.

Hamscher, W. C. 1992. Modeling Accounting Systems to Support Multiple Tasks: A Progress Report. In Proceedings of the Tenth National Conference on Artificial Intelligence, 519-524. Menlo Park, Calif.: AAAI Press/ The MIT Press.

Hamscher, W. C., Console, L., and de Kleer, J. eds. 1992. *Readings in Model-based Diagnosis*. San Francisco, Calif.: Morgan Kaufmann.

Hamscher, W. C. 1994. Explaining Financial Results. *International Journal of Accounting, Finance, and Management*, 3(1): 1-19.

Hart, P. E.; Barzilay, A.; and Duda, R. O. 1986. Qualitative reasoning for financial assessments: A prospectus. *AI Magazine*, 7(1):62-68.