Fraud Simulator: AI-Planning Techniques for Internal Control Evaluation and Assessment of Fraud Opportunities

by Miklos A. Vasarhelyi and
Joseph Natovich

Columbia Institute for Tele-Information
Graduate School of Business
809 Uris Hall
New York, NY 10027
(212)854-4222

705

FRAUD SIMULATOR: AI - PLANNING TECHNIQUES FOR INTERNAL CONTROL EVALUATION AND ASSESSMENT OF FRAUD OPPORTUNITIES

Joseph Natovich

Miklos A. Vasarhelyi

Graduate School of Management, Rutgers University, New Jersey

June 1994

Rutgers Accounting Research Center
Ackerson Hall
92 New Street
Newark, New Jersey, 07102
Telephone: (201) 648-5172
email: natovich@draco.rutgers.edu
vasarhelyi@draco.rutgers.edu

FRAUD SIMULATOR: AI - PLANNING TECHNIQUES FOR INTERNAL CONTROL EVALUATION AND ASSESSMENT OF FRAUD OPPORTUNITIES

1. Introduction

Many organizations are struggling in the battle against fraud. A recent survey [KPMG Peat Marwick, 1993] identified poor internal controls as the most frequent cause for fraud. Essential to fraud control is the understanding of opportunities that facilitate its occurrence. Several decision aids and methods have been developed to help the auditor in internal control evaluation and fraud detection. Among them we find: (a) simulation models, (b) expert systems and (c) "red flags". However, with respect to analyzing fraud opportunities, the effectiveness of these methods is limited.

This paper explores a different approach for internal control evaluation and assessment of fraud opportunities using AI - Planning technique. The main idea of this approach is that fraud planning can be regarded as a state-space problem, a type of problems that a computational planner can resolve. This planner requires: (a) a knowledge based representation of the accounting system, (b) definition of the perpetrator's "world", including characteristics of a successful fraud (goals) and a set of available actions for its achievement, and (c) search strategies to find fraud plans successful in the modeled system's domain - the planner's engine. These generated fraud scenarios reveal the weaknesses in the internal control and suggest a direction for a detailed investigation of potential loss from fraud.

This approach aims to assist the auditor where decision aids are not effective; contrary to expert systems, it does not apply "expert knowledge" but a search algorithm using general heuristics, it can simulate fraud and other unstructured threats that traditional simulation techniques cannot model. Also, its output provides a guideline for further audit procedures that can actually discover the fraud, not as the "red flags" method that can support only a vague assessment of the likelihood of fraud .

Threats to the accounting systems other than fraud (e.g. errors, "hacking", computer malfunctioning) can also be modeled. Applying a set of such major threats on the accounting system modeled this way, can help the auditor to determine the overall internal control effectiveness.

A working prototype - **Fraud Simulator** has been built to demonstrate the feasibility of such an approach. This prototype is a Prolog platform, based on the heuristic search of means-ends analysis. It is designed to search for possible fraud, in a given knowledge base model of any accounting system. If fraud is possible, it names the position of the potential perpetrator and lists the sequence of actions representing the fraud plan.

The next section reviews the three major types of advanced decision aids, proposed in the literature, to evaluate the internal control and fraud likelihood: expert systems, simulation and "red flags". The Fraud Simulator approach and the concept of AI Planning is presented in section 3. In section 4, 5, and 6 we describe in details each one of the three Fraud Simulator's components: Accounting system's model, Perpetrator's world and Planner Engine. Section 7 illustrates an example of generating fraud scenarios with Fraud Simulator. We conclude with some comments on the usefulness of this approach, its limitations and a plan for further research.

2. Advanced Decision Aids for Internal Control

The evaluation of internal controls is a difficult judgmental tasks in the audit process [Meservy et al., 1986]. Its purpose is to determine whether the internal controls implemented in the accounting system are strong enough to protect from potential threats (such as omissions and errors, frauds, thefts etc.) . SAS #53, [AICPA, 1988], has significantly increased the auditor's responsibility in internal control fraud likelihood evaluation. In order to cope

with the complexity involved with internal control evaluation several decision aids were developed. This section discusses these aids and their limitations.

2.1 Expert Systems

The traditional internal control review and evaluation consists of questionnaires and check lists that help the auditor to identify existing and absent patterns of control. This traditional approach served as a natural foundation of the computational expert systems approach. In fact, some of the first expert systems started out as automated version of paper questionnaires [Brown, 1991].The expert systems approach is based on a computational representation of an expert auditors' knowledge and on a collection of heuristics used to identify weaknesses and strengths in the internal control[1]. Gal (1985) constructed an expert computer program for evaluating the internal control . This computer model embodies an auditor's knowledge about the evaluation of accounting controls. Meservy, Bailey and Johnson (1986), gathered about 300 rules representing a knowledge of one expert auditor to evaluate internal control and implemented them in an expert systems. The nature of expert system forces specialization in a narrower domains: ICES [Grudnitski, 1986], is an internal control expert system which focuses on Sales/ Account Receivable Transactions. EDP-XPERT, an expert system for internal control evaluation in EDP environment, developed by Hansen and Messier [1986]. Brown [1991] found that five out of the "big six" accounting firms has developed and used an expert system for internal control evaluation. However, expert systems are confronted with increasing criticism, such as [Selfridge and Biggs, 1989]: (a) They break down or perform poorly when confronted with situations outside the boundaries of their domain, (b) they lack human level explanation and learning capabilities that characterize expert behavior and (c) they often do not outperform human experts. Also, they tend to have a negative impact on users, reducing their control over tasks and reducing the job's cognitive demand [Gill, 1993]

2.2 Simulation

Simulation techniques are a process of "walk through" on an internal control model, testing the impact of controls effectiveness on the total amount of error. Burns and Loebbecke [1975] suggest a tailored simulation program as a method to evaluate the internal control in complex, large scale systems. Knechel [1985] improves this approach by developing a generalized simulation function routines for internal control evaluation. The routines are used as a building blocks for modeling a specific system. However, this model assumes that errors are random independent events, and does not take in to account error inter-dependencies. Wiggins and Smith [1985] develop further the simulation approach by providing a better simulation tool - SIM, and by allowing dependencies among errors and controls within the model. It takes into consideration fraud possibilities, however, the fraudulent pattern has to be predefined in the model by the auditor. The simulation process does not generate the fraud scenarios, it only measures quantitatively, in terms of total amount, the losses due to fraud scenarios that are included in the simulation model.

2.3 "Red Flags"

The accounting profession developed the red flags approach, a circumstantial factor list which might indicate the existence of fraud, and bring the auditor to exert excessive skepticism. SAS # 53 [AICPA, 1988] lists some of these red flags. Loebbecke and Willingham [1988] classified red flags to three categories: (a) *Conditions* - indicators of conditions that allow fraud, (b) *Motivation* - indicators that demonstrate personal motivation to commit fraud by persons in a position of trust (c) *Attitude* - indicators of general low moral and ethics within the organization. Loebbecke et al. [1989] found that red flags consistent with this classification were frequently present in cases of material management fraud. Bell et al. [1991] incorporated red flags into statistical factor combination model, in order to aid assessment of overall risk judgment. The red flags method, still, does not provide a way to detect the fraud. Furthermore, this method requires auditor judgment in areas where he/she is not competent, such as behavioral sciences.

---

[1] For example: existence of adequate segregation of duties.

3. Fraud Simulator Approach

The idea of generating possible fraud scenarios as an approach to examine the fraud likelihood, is not new. Robertson [1993, pp. 351], states that *"when studying business operations, auditor's ability to "think like a crook" to devise ways to steal can help in the planning of procedures designed to determine whether it happen"*. Wallace [1991] suggests that *"when fraud is expected, auditors perform threat analysis, which means they examine what assets are held and how those assets can be taken. Then the auditors strive to outsmart the crooks"*. The manual form of this approach requires creativity and deep knowledge of the system. It is recommended to form an auditor's group brainstorming session to generate possible ways to defraud the system [Leinicke et al., 1990].

This paper suggests an automated form of this approach, that is to perform a computational threat analysis of an accounting system and to generate fraud scenarios by taking advantage on AI planning, a problem solving technique. The rest of this section presents the concept of planning technique and show how fraud can be formulated as a planning problem.

3.1 Concept of AI planning

Planners are problem solvers that deal with complicated problems by decomposing them to manageable subparts (i.e. sequence of small steps) and by recording and handling interactions among subparts during the problem solving process [Rich and Knight ,1991]. Planning problems have to be defined in terms of problem's world ***initial state, goals*** and ***actions*** [Wilkins 1988] .

The state is represented by taking a "snapshot" of the problem's world at one particular time and describing the world as it appears in this snapshot. The problem's world at time zero is the "initial state". Goals are the characteristics of world's final state that indicate the problem has been solved. Actions that can be taken in the world must be represented in such a way that the planner can take the state of the world in which the action is performed, and to map it into the state of the world that will exist after the action is performed. Exhibit 1 illustrates problem definition with classic problem of blocks world.

The solution of the problem is a sequence of actions that transforms the problems world from its initial state to a final state where all goals are achieved . This solution is a result of a search algorithm and heuristics. We discuss this issue later on in section 6.

Exhibit 1 here

3.2 Fraud as a planing problem

Fraud is a crime committed by a person employed in a position of trust by an organization. A position of trust in this context means a position that carries authority over people and /or property belonging to another (usually the employer organization). Fraud can be formed in a variety of illegal acts against different type of victims such as management fraud against shareholders, fraud against external and governmental entities such as the IRS [Bologna 1984]. Fraud schemes are not only versatile but also require multiple-steps. Typically, they consist on three acts (a) *transfer* - transfer of an asset or a business interest to the perpetrator, (b) *concealment* - falsification of records and documents to conceal the act of transfer, and (c) *conversion* - the asset need to be converted in to a value suitable to the perpetrator [West, 1981].

In general, fraud problem can be formulated as the following: **given the accounting system and its internal control, who are in position to commit fraud and what are the ways available to do so?** In particular, we need to define carefully the three elements of the problem: the fraud *world* - the collection of objects that noticeable to

the fraud planner, goa*ls* - the characteristics of a successful fraud, and the set of ***actions*** available to make transformations in the state of the system. Defining these elements of the problem, we enable the planner to search for solutions .
Fraud Simulator is a prototype built to demonstrate feasibility of this concept. it contains three parts (Exhibit 2) :

a) **System's world**. A knowledge based model of the examined accounting system. It defines the specific system's elements in the fraud state-space problem. It contains definition of the system's objects and its attributes. Also it contains the different procedures in the system and the internal controls. The system's world together with the perpetrator's world provides the complete definition of the fraud problem .

(b) **Perpetrator's world**. A general model defines the perpetrator's elements in the fraud state-space problem. It includes some basic assumptions on the perpetrator's behavior, her goals, a special set of illegal actions she can use . The perpetrator's world is general and applicable to various systems examined.

(c) **The planner engine**. This part is based on a general planner that can be used for solve planning problems in different domains. It is an algorithm that searches for a path of actions that can transform the initial state of the system to a final state in where the fraud goals is met. This part use problem definition in both perpetrator's world and system's world to solve it and generates the fraud scenarios.

Together, the system's world and perpetrator's world define the three element of fraud problem - world, goals and actions. It is important to distinguish between the perpetrator's world and the system's world. For each system which is examined, a different system's world should be built, but the perpetrator's world with the fraud goal and the set of available illegal actions, like the planner engine, can be applicable to all systems. In the next sections of the paper we discuss the three parts of Fraud Simulator.

Exhibit 2 here

---

4. The System's World

This part of fraud simulator contains the knowledge based representation of the system examined. The model must be defined for each system examined by Fraud Simulator. The nature of auditing practice, (as opposed, for example to medical practice), is that it has relatively limited number of clients and service is provided on a periodical basis. In addition, there are different tasks in auditing, that require as one of their inputs qualitative knowledge of the accounting system's domain . This makes it attractive to build a knowledge based representation of clients' accounting systems to support multiple audit tools and tasks. TICOM [Bailey et al., 1985] and Savile [Hamscher, 1992] are examples of work in this direction.

TICOM is designed to support a certain class of queries about the accounting system in respect to the internal control evaluation task . Its purpose is to replace the traditional representation tools such as flowchart diagrams and narratives with an enhanced computerized tool. However, it did not manage to provide an evaluation the internal control. Savile is a knowledge based representation of an accounting system, oriented toward audit planning and designed to support multiple audit tasks. Both TICOM and Savile provide a very similar ontology, enabling the auditor to model various types of accounting systems using a limited action vocabulary (Exhibit 3). The ontology should give the user the flexibility to describe the system in any level of detail.

Exhibit 3 here

The issue of an adequate way to represent the accounting system has been investigated in our work. We basically use the concept suggested in TICOM and Savile, however to represent properly the system's world for the planner, we must develop a richer representation, that will include additional phenomena such as dynamic changes in system's state during the process. Therefore, our extended form of accounting system modeling includes elements such as objects, states events, actions and internal controls .

4.1 Objects

Typical examples of objects in accounting systems are the following:

- positions: who are the performers in this systems.
- instruments: tools and documents that are used during the processes: e.g. general ledger, safe, invoices, receipts, vendor file.
- assets: computer, cash, bank account etc.

4.2 States

The state of a system is a set of object attributes at a specific a point in time . The state is dynamically changed by events and actions. So we can describe a process in the system as a sequence of states.
Examples of the basic attributes that reflect a state are:

- place - the location of the object. For example place can be mailroom, clerk or a file.
- status- status of an object in the procedure. For example: hold, registered, posted.
- bal - the balance of a specific account. In the current prototype balance can be : + (debit), - (credit),or 0 (balanced).

An example of a possible state is:

place of *invoice* is *mailroom*	[place(invoice,mailroom),
status of *invoice* is *registered*	status(invoice,registered),
account's balance of *vendor* is *0*	,bal(vendor,0)].

4.3 Events and Actions

The knowledge base of the accounting system is mainly its procedures and processes. Procedures are usually triggered by an external event. For example, the external event "customer pays" triggers the procedure of cash receipt. Each procedure is built of a sequence of actions. An action must specify the performer and the objects involved . For example:

AP clerk checks *the invoice*	check(apclerk, invoice).

Action can be performed only if defined preconditions in the system's state are met. Also, as explained earlier, actions and events change the state of the system. For each event or action there is a definition of its effect on the system's state. This representation of the actions is required in order to allow the planner's reasoning . An example of the formal representation of an action is presented in Exhibit 4.

Exhibit 4 here

4.5 Internal controls

Internal controls are classified as preventive controls and detective (feedback) controls.

(a) Preventive controls

These are controls that prevent the occurrence of undesirable actions or events. Controls such as access authorization and segregation of duties are included in this category. For example: access control to an object is allowed only to employee who is responsible for this object. This control can be formalized as:

position *P* has access to Object *O* if:	has_access(Position,Object) :-
position *P* is responsible to a list	responsible(Position,List),
of objects *L* , and	
object *O* is included in that list *L*	member(Object,List).

(b) Detective (feedback) controls .

Detective controls are external to the procedures and rely on investigating the state after the procedure is completed. The control is defined as a combination of objects attributes required in the final state. If such a combination is met, the final state is correct, otherwise the final state is suspicious and will be checked.

For example: after completing the procedure of receiving cash, the final state should be either (1) there is cash in the safe and the balance of the cash account is positive, or (2) there is no cash and the balance is 0. Otherwise (e.g. no cash but balance is positive) the final state is suspect.

Control1: check that either-	
place of *cash* is in *safe*	control1([place(cash,safe),bal(cash,+)]).
and account's balance of *cash* is *positive*	
or	
there is *no cash* and the	control1([place(cash,nowhere),bal(cash,0).
account 's balance of *cash* is *zero*	

5 The perpetrator's world

This part of Fraud Simulator, models the general representation of perpetrator behavior in respect to fraud scheme. It includes model assumptions on perpetrator's behavior, her goal and illegal actions she can use.

5.1 Assumptions on Perpetrator's behavior

Some assumptions on perpetrator's behavior were incorporated into the Fraud Simulator. These assumptions are based solely on simple common sense. The purpose of some assumptions is to restrict the search process of the planner only to areas that promise meaningful and useful solutions. Other assumptions has been made for simplicity. The assumptions and their rationale are:

(a) The perpetrator is an employee in the system. Our purpose is to provide a method to test internal control effectiveness against fraud. Since management can override internal controls, it is not effective against management fraud. Therefore we mainly focus on employee fraud.

(b) The perpetrator is familiar with the system and its controls. The perpetrator does not want to be caught therefore he must comply with the controls. This way prevents generation of fraud that internal controls managed to detect. We are interested mainly in the possible undetected frauds.

(c) Perpetrator would not take risks and will comply internal controls without regarding their effectiveness probability. For simplicity, this version of Fraud Simulator is deterministic.

(d) There is no collusion, only one person involved in the fraud. However, he can take advantage of other employees' actions if they act according to the internal regulations. Our rationale was that since internal controls are less effective against collusion, and collusion opportunities are simple to identify, Fraud Simulator's contribution to internal control evaluation is higher when focusing on individual fraud.

5.2 Perpetrator's goals

Donald Cressey [1953] in his book " Other People's Money", concludes that fraud is likely to occur when the following conditions are met (a) perpetrator faces *non-shareable* financial problem (such as gambling debts), (b) perpetrator has the *opportunity* and the technical skills and (c) perpetrator develops a rational *justification* to his misconduct (such as consider it as "borrowing"). The perpetrator's objective is to "borrow" money from his company, and to make good long before his manipulation can be detected [Seidman, 1990].

Accordingly, we defined the Fraud Simulator's goal as:

(a) To get a company's asset. This asset can be cash or cash convertible such as checks or inventory.

(b) Not to be discovered. This goal can be achieved only by complying with the various detective internal control procedures.

The final state of the system after fraud completion must comply with all detective controls defined for the accounting system. Recall that the assumption is that perpetrator has a complete knowledge about system's control, otherwise the fraud would be detected by the unknown control and will fail. This can be articulated as:

the goal of position X is to have an asset A and the system state S such that S complies with the detective controls.	goal(X,[has(X,Asset)|systemState]):- comply_controls(systemState)

5.3 Perpetrators Illegal Actions

In order to commit fraud, a set of actions must be available to her. These actions manipulate the system state. The fraud goals can be achieved using a certain sequence of a subset of actions. Albrecht et al. [1984] documented 212 fraud cases through questionnaires mailed to internal audit departments. "Classic" fraud schemes are sketched in [Allen, 1975; Levy, 1985; Seidman 1990; Albrecht, 1991]. In the cash receipt cycle, for example, the following fraud schemes has been documented:

- account transfer : steal money from inactive customer account

- lapping: employee steals receivables from one customer, and attempt to cover the theft by applying to that account later collection from another customer.

- credit memo fraud: stealing the payment made by a customer and issuing that customer a credit memo, indicating that the service rendered or goods received was unacceptable.

- petty cash scheme; employee embezzlement of petty cash camouflaged by false documents.

Cressy found that perpetrators do not depart from their ordinary occupational routine, in order to commit fraud, because they have the technical skills and general information about in this capacity of converting the firm's funds [Cressy , 1951 , pp. 84-85] to their personal usage. In conclusion:

(a) legitimate actions which are part of the accounting systems and

(b) illegal actions which are not defined in the accounting system's activities, but they are the core of the fraud schemes.

For example, in the vendor fraud scheme the employee creates a dummy vendor and a dummy invoice, which are fraudulent actions. Then the procedure of approving the invoice and paying the vendor might be part of the legitimate activities in the accounting system.

Since the legitimate activities are defined as a part of the accounting system (see section 4 above), what is left for definition is fraudulent actions. As in the legitimate activities, it is necessary to define the impact of such activity on the state of the system and the precondition allowing ,physically, this activities. These action representation is required for the planner's reasoning process.

For example a common illegal action is: posting a journal entry in the ledger to write off an asset that has been taken against expense account in order to cover it. The precondition to this action is the asset account must have a debit balance. The action set the asset account balance to zero

6. Planner Engine

The solution of a planning problem is a sequence of actions that transforms the problems world from its initial state to a final state where all goals are achieved. A search algorithm should be used to search systematically for such a sequence. Depth-first and breadth-first are two common "search extensive" alternatives. Another alternative to these search strategies is the depth first iterative deepening algorithm [Korf, 1985]. This method performs successive depth- first searches on succeeding levels. However, most planners use heuristics to reduce search effort. The Logic Theory Machine (LT) is the first heuristic program fully realized on a computer [Newell, Show and Simon 1957]. GPS [Newell and Simon, 1963] and STRIPS [Fikes and Nilsson, 1971] use means-ends analysis, a general heuristic for selecting among alternative actions by looking at their ability to achieve any goal from the goals set . More advanced planners such as NOAH [Sacerdoti, 1975] and TWEAK [Chapman, 1987] use means-ends analysis in a non-linear planning technique that can deal more efficiently with conjunctive goals. ABSTRIPS [Sacerdoti, 1974] presents the abstraction heuristic that gives priority to solve the critical tasks in the

problem first and then to fill in the other details in the plan. Meta-planning [Stefic, 1981] is a technique for reasoning about the planning process itself.

For the initial prototype of Fraud Simulator we found a simple "means-ends analysis" planner as sufficient. The version used in this program is the "goal regression" [Bratko 1990 , pp. 414] . Basically the structure of this algorithm is as the following:

- **Input** - the input to the planner is taken from both the perpetrator's world and the system's world . It is a STRIPS - like problem representation that includes:
  - goals of the fraud as defined in the perpetrator's world .
  - an initial state of the system before any action is taken, as defined in the system's world.
  - a list of possible actions in the system. For each action there is a need to define what are the condition under it can be taken and how this specific action changes the system state. The actions considered are both legitimate actions as defined in the system's world and fraudulent actions as defined in the perpetrator's world .
- **Process**: The planner tries to find a path of actions from the starting state to the goal by regressing the goals. The search goes backward from the end to the beginning : it starts from the goals and try to find the action which achieves one of those goals. But in order to do this action there are some conditions that have to be met, so now these conditions join the goal list instead of the original goal. Then again, planner tries to find another action that achieves one of the new goals list. And so on until the regressed goals list is fully met by the starting state.
- **Output:** The inverse path of actions (that goes from the start state to the goals) is the solution - in our case, the fraud scenario.

7. Example : Cash receipt system

The abilities of Fraud Simulator has been demonstrated using a few examples of accounting system in [Natovich, 1993]. The following example is one of them. This is a procedure of cash receipts. Exhibit 5 contains the system world. In this example there are two positions - cashier and accountant but there is no segregation of duties - the cashier can also post entries to the accounting ledger. However, there is a detective control - periodical cash count i.e. the balance of the cash account should match the actual amount of cash in the safe. Fraud Simulator is run to search for fraud opportunities.

Exhibit 5 here

7.1 First Run

The results of the Fraud Simulator run are presented in Exhibit 6. It shows that this is very simple for the cashier to commit a fraud: Whenever he gets cash he can take it. Though a detective control - cash count, it is not effective because cashier does not register the cash and it remains " off the books" .

Exhibit 6 here

7.2 Second Run

In order to strengthen control, we add another detective control - customer reconciliation. It can be implemented, for example, by sending the customer a copy of his account balance. If the customer paid and his account is still in debit balance, he would inform on it and the fraud would be discovered:

Control 2 :check that either	
customer paid and	control2([status(customer, paid),
account's balance of *customer* is *zero.*	bal(customer, 0)]).
or	
customer did not pay and	control2([status(customer, unpaid),
account's balance of *customer* is in *debit.*	bal(customer,+)]).

Running the Fraud Simulator now shows that fraud will be more complicated but still possible (Exhibit 7). The cashier credits the customer against cash to comply with control 2. Then he credits the cash account against expenses and set the cash balance to zero, in order to comply also with control.

Exhibit 7 here

7.3 Third Run

In order to completely prevent fraud we have to implement a segregation of duties and to limit cashier responsibilities for him not to be able to post transactions:

the cashier is responsible for the cash only.	responsible(cashier,[cash]).
accountant is responsible for accounting books.	responsible(accountant,[books]).

After a "desperate search", the answer of Fraud Simulator this time is:

- no

Which means fraud is not possible .

8. Conclusion

We have shown that a working prototype of the fraud simulator can work and simulate fraud in a given system, and hence, to demonstrate weaknesses in the internal control. In this section we conclude on the applicability of the Fraud Simulator as a potential tool, the limitation of it and the program for further research.

8.1 Potential Application of Fraud Simulator

We described and demonstrated Fraud Simulator ability to generate fraud scenarios in a given system. This ability makes it a potential tool for system and control design, auditing and education

(a) Designing systems: Today, system designers have much to consider in terms of fraud prevention and detection [Le Grand, 1986]. System designers and control experts need to design appropriate internal controls in order to reduce fraud risk. Fraud Simulator provides designers and control experts with the ability to test system's controls in the design phase . Using a "what if" method, different alternative controls can be tested in a low cost before decision on the final design.

(b) Auditing : as discussed previously, Fraud Simulator can assist in internal control evaluation, and assessment of fraud likelihood. Weaknesses in the control and fraud opportunity can be revealed by the generated fraud scenarios. This scenarios also provides directions for further audit procedures required to detect the potential fraud.

(c) Education: Fraud Simulator can be useful aid for education and training in auditing. The feedback capability and the "real world" threats can enhance internal control teaching. Worrel [1992], pointed on the need to develop education aids in accounting to provide the student "real world " experience . He suggested a simulation game to enhance student's understanding of internal control concept. In this game one group of students plays the role of system designers that implement internal control, second group plays the perpetrators that commit frauds and third group plays the auditors that try to detect fraud. Fraud Simulator can improve the game by automate the role of perpetrator, letting the student concentrate in the audit and design roles. Also, it increases the interaction between the players, making it more intensive and effective.

8.2 Limitations

Some limitations have been identified : First, it shows only the fraud scenarios, but it does not point to a specific weakness in the internal controls that allows for this possibility of fraud. Also, it does provide suggestions to improve the weak controls, (although it examines solutions offered by the user). The use of expert systems, associated to this approach, may provide some of these features.

8.3. A Program for Further Research in the Fraud Simulator

In additon to the adoption of a mixed approach with expert systems as discussed above, future research may evolve towards three directions:

- First we want to upgrade the "traditional" planner used in the initial prototype with a case-based planning ability.
- Second, the assumption of deterministic system should be loosen to enable fraud planning under uncertainty of internal control effectiveness.
- Finally, the concept presented in this paper can be generalized to other threats to accounting systems.

(a) case-based planner:

Research shows increasing interest in Case-Based planning. This method is based on the idea that a machine planner should make use of its own past experience in developing new plans relying on its memory instead of a base of rules [Hammond, 1986]. This method has been implemented in several domains such as: engineering - design of hydro-mechanical devices [Navinchandra et al. 1991], law- reasoning from legal precedent [Branting, 1988], and Medicine - diagnosis of heart failure, [Koton, 1988]. In the accounting domain CBR (case-based reasoning) has been used for risk assessment of audit judgment [Denna et al. 1992] and accounting regulation [O'leary, 1992].

There are two main justifications for using Case Based planning in Fraud Simulator: (1) it eliminates the need for any "expert knowledge" in analyzing the fraud activities domain, instead the planner engine will consult a database of fraud cases, (2) The search process can be more efficient, instead of searching and generating a

complete fraud plan from scratch, the planner can use a past plan and modify as well as repair it to work in the given accounting system.

(b) planning under uncertainty

The current prototype assumes that each defined control is completely reliable and there are no failures. A more realistic model of accounting system should assume uncertainty of control reliability. It is also useful to loosen the assumption that the perpetrator would not take any risk committing the fraud. In such a world more fraud plans are available, but their probability of success is dependent on the reliability level of the controls. A planner under uncertainty need to be able generate fraud schemes with probability of success above a defined threshold.

(c) model of various threats

Internal control has to protect the system not only from fraud threats but also from various other threats, such as natural disasters, errors and omissions etc. Building simulators to other threats, using the same approach presented in this paper, enables the performance of a complete threats analysis for an examined system.

9. Summary

As fraud becomes more sophisticated, new advanced method of analysis are needed. This paper suggest a new approach that uses AI planning techniques to generate fraud schemes used to identify exposures in the accounting system and to point on weaknesses in the internal control. Some of the characteristics of this approach were investigated using a computational prototype - Fraud Simulator . Although , further research is necessary to examine and to enhance the usefulness of this approach, some obvious benefits were demonstrated. One possible recommendation is the use of a mixed approach, which integrates both, an expert system and this AI planning approach.

References

AICPA, (1988), Statement of Auditing Standards No. 53, AICPA, New York.

Albrecht, W. S., K. R. Howe and M. B. Romney, (1984), Deterring Fraud: The Internal Auditor's Perspective, The Institute of Internal Auditors Research Foundation, Altamonte Springs, Florida.

Albrecht, W. S. and D. W. Schmoldt, (1988), Employee fraud, in Business Horizon, July/August, pp. 16-18.

Allen, B., (1975), Embezzler's guide to the computer, in Harvard Business Review, July/August, pp. 79-89.

Bailey, A. D. Jr. et al., (1985),TICOM and the analysis of internal control, The Accounting Review, (LX: 2),April , pp. 186-201.

Bailey, A. D., R. P. McAfee and A. B. Whinston, (1981), An application of complexity theory to the analysis of internal control systems, in Auditing: A Journal of Theory and Practice, (1:1), Summer.

Bell, T. B., (1991), S. Szykowny and J. J. Willingham, (1991), Assessing the Likelihood Of Fraudulent Financial Reporting: A Cascaded Logit Approach, unpublished paper.

Branting, L. K., (1988), The role of explanation in reasoning from legal precedents, in Proceedings of a DARPA-sponsored workshop on case base reasoning , Kolonder, J. (Ed), May 1988, Clearwater Beach, FL.

Bratko, I., (1990), PROLOG, Programming Language for Artificial Intelligence, 2nd Edition, Addison-Wesley.

Brown, C., (1991), Expert Systems in Public Accounting: Current Practice and Future Directions, Expert Systems With Applications, (3:1), pp. 3-18.

Burns, D. C. and J. K. Loebbecke (1975) , Internal control evaluation how the computer can help, in Journal of Accountancy, August 1975, pp. 60-70

Chapman, D. , (1987), Planing for conjunctive goals, in Artificial Intelligence, (32:3).

Cressey, D. R. (1971), Other people's money, a study in a social psychology of embezzlement, Wadsworth, Belmont, CA.

Denna E. L, J. V Hansen, R. D. Meservy and L.E Wood, (1992), Case-based reasoning and risk assessment in audit judgment, in International Journal of Intelligent Systems in Accounting Finance & Management, (1:3), September, pp. 163-172.

Fikes, R. E and N. J. Nilsson (1971), STRIPS a new approach to the application of theorem proving to problem solving , in Artificial Intelligence,(2:3/4), pp. 189-208.

Gal, G., (1985), Using Auditor Knowledge to Formulate Data Model Constrains: An Expert Systems for Internal Control Evaluation, unpublished Ph.D. dissertation, Michigan State University.

Gill, T. G., (1993) , Why expert systems fail... , in Proceedings of the 6th Florida Artificial Intelligence Research Symposium - FLAIRS, pp. 26-30.

Grudnitski, G. (1986), A prototype of an internal control expert system for the sales / accounts receivable application , working draft, University of Texas, Austin.

Hammond, K. J. (1986), CHEF, a model of case-based planning, in Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86), August 11-15, 1986, Vol. 1 (Science), Philadelphia, PA.

Hamscher ,W., (1992), Modeling Accounting Systems to Support Multiple Tasks: A Progress Report, in Proc. 10th National Conf. on Artificial Intelligence, San Jose, Calif., July 1992.

Hansen J. V. and W.F. Messier (1986), A preliminary investigation of EDP - Xpert, in Auditing: A Journal of Practice and Theory, (6:1), Fall, pp. 44-74, also in M. A. Vasarhelyi (ed.), Artificial Intelligence in Accounting and Auditing: The Case of Expert Systems, Markus Wiener, NY, 1989.

Knechel, W. R. (1985) , A simulation model for evaluating accounting system reliability, in Auditing: A Journal of Practice and Theory, (4:2), Spring, pp. 38-62.

Kotton, P., (1988), Reasoning about evidence in causal explanations, in the Proceedings of a DARPA-sponsored Workshop on Case- Based reasoning , Kolonder, J. (Ed), May 1988, Clearwater Beach, FL.

KPMG Peat Marwick, (1993), Fraud Survey Results, KPMG Peat Marwick.

Le Grand C. H. (1986), Discouraging fraud through system design, in The Internal Auditor, April, pp. 28-35.

Lenicke, L. M., W. M. Rexroad and J. D. Word, (1990), Computer fraud auditing: it works, in The Internal Auditor, August 1990, pp. 26- 33.

Levy, M. M., (1985), Financial fraud: schemes and indicia, in The Journal of Accountancy, August, pp. 78-87.

Loebbecke, J. K. , M Eining and J. Willingham, (1989), Auditors experience with material irregularities: Frequency nature and Delectability, in Auditing: The Journal of Practice and Theory, (9:1), Fall, pp. 1-28.

Loebbecke, J. K and J. J. Willingham, (1988), Review of SEC and auditing enforcement releases, Unpublished working paper.

Meservy, R., A. D. Bailey Jr. and P. E. Johnson, (1986), Internal control evaluation: a Computation Model of the Review Process, Auditing: A Journal of Practice and Theory, (6:1), Fall, pp. 44-73, also in M. A. Vasarhelyi (ed.), Artificial Intelligence in Accounting and Auditing: The Use of Expert Systems, Markus Weiner, NY, 1989.

Natovich, J. (1993) Evaluating Internal Controls Using Fraud Simulator, w*orking paper* , Rutgers University.
Navinchandra, D., K. P. Sycara and S. Narashimhan (1991), Behavioral synthesis in CADET, a case-based design tool, in Proceedings of the Seventh IEEE Conference on Artificial Intelligence, February 1991, Miami, FL.

Newell, A and Simon H. A (1963), GPS a program that simulates human thoughts in E. A Feigenbaum and J. Feldman eds., Computer and Thoughts, McGraw-Hill, New-York.

O'leary, D. E., Case-based reasoning and multiple agent systems for accounting regulation systems with extensions, in International Journal of Intelligent Systems in Accounting Finance & Management, (1:3), January, pp. 41-52.

Rich, E. and K. Knight,(1991), Artificial Intelligence, 2nd Edition, McGraw-Hill

Robertson, J. C. (1993), Auditing, 7th ed., Irwin.

Sacerdoti, E. D., (1975), The nonlinear nature of plans , in Proceedings IJCAI-75.

Sacerdoti, E. D., (1974), Planning in hierarchy of abstraction spaces, in Artificial Intelligence, (5:2), pp. 115-135.

Seidman(1990), A case study of employee frauds, in The CPA Journal, January , pp. 28-35.

Selfridge, M. and S. F. Biggs, (1989), GCX, a Computational Model of the Auditor's Going Concern Judgment, Proc. Audit Judgment Symp., Univ. of Southern California , Los Angeles, CA, February, 1989.

Stefic, M. , Planning and meta-planning, (Molgen part 2) in, Artificial Intelligence, (16:2), pp. 141-169.

Wallace, W. A. (1991), Auditing, 2nd ed., PWS-Kent, Boston, Ma.

West, H., (1987), Fraud, the growth industry, Kogan Page, Corby, Great Britain.

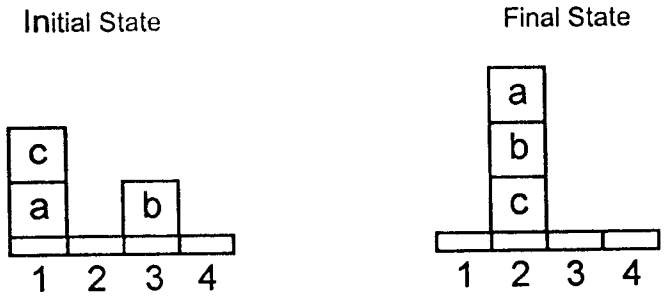
Wiggins, C. E. Jr. and L. M. Smith (1987), A generalized audit simulation tool for evaluating the reliability of internal controls, in Contemporary Accounting Research (3:2), pp. 316-337.

Wilkins, D. E., (1988), Practical Planning: Extending the Classical AI planning paradigm, Morgan Kaufmann, San Mateo, CA.

Worrel, M. L. (1992), Using a simulation game to improve learning in accounting and auditing : a field study, a dissertation proposal ,Graduate School of Management, Rutgers University.

a) A problem in the blocks world:

Find a sequence of actions that achieves the goals: a on b and b on c . These actions transform the initial state (on the left) to the final state (on the right).



b) A definition of the problem as a planning problem

- A blocks world
  blocks and places are objects
  a, b and c and are blocks
  1, 2, 3, 4 are places
- Initial state
  2 is clear, 4 is clear, b is clear, c is clear, a is on 1, b is on 3, c is on a
- Goals
  a is on b, b is on c
- Action
  move B from F to T :

  Preconditions to the action move B from F to T :
  B is a block, F and T are objects, B is clear, T is clear , B is on F ,B is not F , F is not T , B is not T

  State transformation as a result of the action move B from F to T :
  Add elements to state: B is on T, F is clear.
  Delete elements from state: B is on F , T is clear

Exhibit 1 : The Blocks World - The Classic Planning Problem Example, [Bratco, 1990]

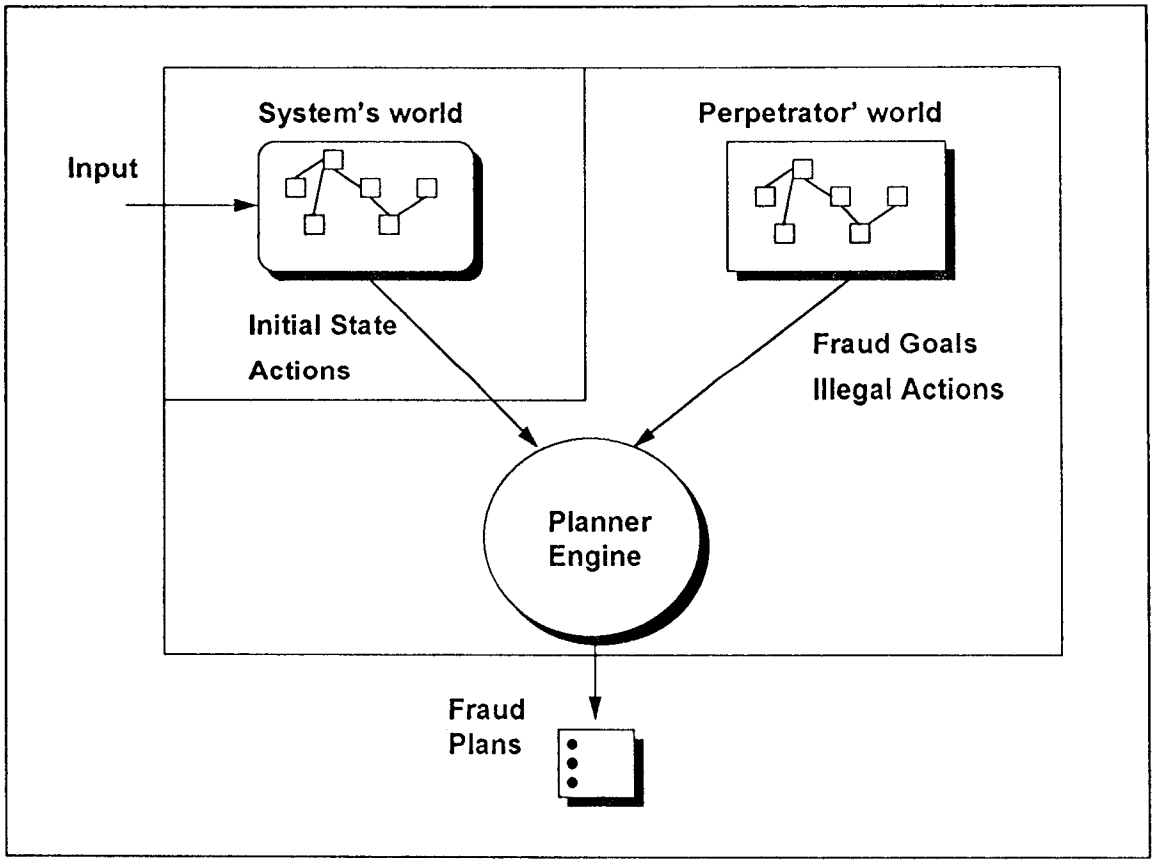


Exhibit 2: Fraud Simulator Components

assign (field destination-record) (field source record)
Put the contents of the source record field in to the destination record field

compute (field-destination record) &rest arguments
Assign to the destination record field a result that depends in some way on the arguments.

create record record-class &optional source-record
Create new new record and assign all fields that it has in common with the source record

copy source-record record
Create a new record that is a copy of the source record

credit or debit record account
Use the "amount" field of the record to create a new debit or credit record in the account

post record debit-account credit-account
perform both a credit and debit operation to the appropriate account.

ensure-equal first-record second-record
Check that all of the fields that the two record have in common agree and repair the problem if they do not.

get record repository
Extract a record from the repository

put record repository
Move a record into the repository.

Exhibit 3 : A segment of action vocabulary in Savile [Hamscher, 1992]

- preconditions for the action "AP *clerk* checks the *invoice" are* :

  the invoice is at the clerk.
  the invoice status is "hold".

- The action updates the state as following :
  the new invoice status is "checked".
  the invoice is transferred to AP manager

- The action deletes from the state:
  the old invoice's status
  the old invoice's state

```
prec(check(apclerk,invoice),
    [place(invoice,apclerk),
      status(invoice,hold)]).

add(check(apclerk,invoice),
    [Status(invoice,checked),
     place(invoice,apmanager)]).

del(check(apclerk,invoice),
    [status(invoice,hold),
     place(invoice,apclerk)]).
```

Exhibit 4 : **Representation of the action: AP clerk checks an invoice**

• **Positions in the system:**	
the positions are a *cashier* and	`position(cashier).`
an accountant	`position(accountant).`
• **Responsibilities on objects of each position:**	
cashier is responsible to *books and cash*	`responsible(cashier,[books,cash]).`
accountant is responsible to *books.*	`responsible(accountant,[books]).`
•**Access control**	
position *P* has access to Object *O* if:	`has_access(Position,Object) :-`
position *P* is responsible to a list	`responsible(Position,List),`
of objects *L* , and	
object *O* is included in that list *L*	`member(Object,List).`
• **Chart of accounts**	
cash account	`account(cash).`
sales account	`account(sales).`
expense account	`account(expense).`
customer account	`account(customer).`
• **Initial state :**	
account's balance of *cash* is *zero* and	`state1([bal(cash,0),`
account's balance of *customer* is *debit*	`bal(customer,+)`
customer's status is *unpaid*	`status(customer,unpaid)]).`
• **Trigger event :**	
customer pays	`event(customer_pay).`
<u>preconditions</u>:	`prec(customer_pay,`
balance of cash account is zero and	`[bal(cash,0),`
balance of customer is debit.	`bal(customer,+)`
customer's status is unpaid	`status(customer,unpaid)]).`
<u>update state:</u>	`add(customer_pay,`
customer's status of is paid	`[status(customer,paid),` cash
status is hold	`status(cash,hold),`
cash is in the mailroom	`place(cash,mailroom)]).`
<u>delete from state:</u>	`del(customer_pay,` customer's status is unpaid
	`status(customer,unpaid]).`

Exhibit 5: Representation of a cash receipt system

- **Actions in the procedure.**

(1) cashier gets the cash.

```
action(get(cashier,cash),cashier).
```

preconditions:

```
prec(get(cashier,cash),
```

cash is in the mailroom

```
     [place(cash,mailroom),
```

cash status is hold

```
      status(cash,hold)]).
```

update state:

```
add(get(cashier,cash),
```

cash is at the cashier's

```
     [place(cash,cashier)]).
```

delete from state

```
del(get(cashier,cash),
```

cash is in the mailroom

```
     [place(cash,mailroom)]).
```

(2) cashier issues receipt

```
action(issue(cashier,receipt),cashier).
```

preconditions:

```
prec(issue(cashier,receipt),
```

cash is at the cashier's

```
     [place(cash,cashier),
```

ash status is hold

```
      status(cash,hold)]).
```

update state :

```
add(issue(cashier,receipt),
```

cash status is registered

```
     [status(cash,registered)]).
```

delete from state:

```
del(issue(cashier,receipt),
```

cash status is hold

```
     [status(cash,hold)]).
```

(3) cashier posts entry to ledger - debit cash account and credit customer.

```
action(post(cashier,account(cash),
                    account(customer)),cashier).
```

preconditions:

```
prec(post(cashier,account(cash),account(customer)),
```

cash status is rergistered

```
                    [status(cash,registered)]).
```

update state:

```
add(post(cashier,account(cash),account(customer)),
```

cash status is posted

```
                    [status(cash,posted),
```

account's balance of customer is zero

```
                    bal(customer,0),
```

account's balance of cash is debit

```
                    bal(cash,+)]).
```

delete from state:

```
del(post(cashier,account(cash),account(customer)),
```

cash status is registered

```
                    [status(cash,registered),
```

account's balance of customer is debit

```
                    bal(customer,+)
```

account's balance of cash is zero

```
                    ,bal(cash,0)]).
```

(4) cashier deposits the cash in the safe.

```
action(deposit(cashier,cash),cashier).
```

preconditions:

```
prec(deposit(cashier,cash),
```

cash status is posted

```
     [status(cash,posted),
```

cash is at the cashier's

```
     place(cash,cashier)]).
```

update state:

```
add(deposit(cashier,cash),
```

cash is in the safe

```
     [place(cash,safe)]).
```

delete from state

```
del(deposit(cashier,cash),
```

cash place is at the cashier's

```
     [place(cash,cashier)]).
```

- **Detective control**

Control1: check that either-

place of *cash* is in *safe*

```
control1([place(cash,safe),bal(cash,+)]).
```

and account's balance of *cash* is *positive*

or

there is *no cash* and the

```
control1([place(cash,nowhere),bal(cash,0).
```

account 's balance of *cash* is *zero*

Exhibit 5: Representation of a cash receipt system - cont.

Exhibit 6: **Results of first run**

perpetrator: *cashier*

The fraud plan is:

---

customer pays	`customer_pay`
cashier gets the *cash*	`get(cashier,cash)`
cashier takes the *cash*	`take(cashier,cash)`

Exhibit 7: Results of second run

Perpetrator: *cashier*

The fraud plan is:

---

customer pays	`customer_pay`
cashier gets the *cash*	`get(cashier,cash)`
cashier issues *receipt*	`issue(cashier,receipt)`
cashier posts entry to ledger - debit *cash* account and credit *customer*	`post(cashier,account(cash),account(customer))`
cashier posts entry to ledger - debit *expense* account and credit *cash.*	`post(cashier,account(expense),account(cash))`
cashier takes the *cash*	`take(cashier,cash)`