An artificial intelligence fraud management solution comprises an expert programmer development system to build trainable general payment fraud models that integrate several artificial intelligence classifiers like neural networks, case-based reasoning, decision trees, genetic algorithms, fuzzy logic, and rules and constraints. These are further integrated by the expert programmers and development system with smart agents and associated real-time profiling, recursive profiles, and long-term profiles. The trainable general payment fraud models are trained with supervised and unsupervised data to produce an applied payment fraud model. This then is applied by a commercial client to process real-time transactions and authorization requests for fraud scores.
Fig. 1

100

expert development programming

102

model development system

104

trainable general payment fraud model

106

108

account data

110

historical transaction data

112

trained payment fraud model

114

116

real-time transaction data

120

client tuning

118

fraud scores

applied payment fraud model
Fig. 2A

- **user interface**
  - smart agent long term profile recursive profile real time profiling
  - advanced neural network algorithms
  - rules & constraints
  - C++ code generator

- **constraint-based programming tool**
  - genetic algorithms
  - case-based reasoning & learning
  - data compression algorithms

- **fuzzy logic tool**
  - simulation & planning tool
  - language interpretation & compilation
  - database connectivity

- **declarative application**

- **general payment fraud model**
Fig. 3

300 wide universe, long history transaction data

302 supervised unsupervised

304 network filter and distributor

306 fraud model build choice of one particular close fitting network model

308raw unfiltered data in whatever format

308 data cleanup

310 harmonized, unified, error-corrected, standardized

312 purchased and imported

312 data enrichment

314 interpolations, extrapolations

314 trainable general payment fraud model

330 classification model

331 classification model

332 classification model

334 build smart-agent population

336 build long-term profiles

338 fraud model

338 model model model smartagent long-term population profiles

338 model model model smartagent long-term population profiles

338 model model model smartagent long-term population profiles
Fig. 4

Selected real-time transactions or authorization requests messages from merchants

Raw transaction data

Not relevant

Relevant transaction data

Timestamps

Long-term profile recursive profiling

Real-time profiling updates

Smart agents

Classification model

Classification model

Classification model

Weighted summation

Fraud score
Fig. 6

merchant authorization requests

raw transaction data

relevance

relevant

not relevant

timestamps

0:00 reported transaction data

acct. no. x

age (minutes)

push down stack

0:37

1:16

3:11

6:17

10:52

11:05

13:41

14:58

>15:00

last 15-minute's transactions

any acct. no. x?

all transactions for account number x that have occurred in the last 15-minutes
selected real-time transactions or authorization request messages from merchants

Fig. 7

raw transaction data ➔ not relevant

relevant transaction data ➔ timestamps

selected applied fraud model for card present

selected applied fraud model for domestic

selected applied fraud model for credit

selected applied fraud model for contactless

selected applied fraud model for jewelry

weighted summation

client tuning

fraud score
ARTIFICIAL INTELLIGENCE FRAUD MANAGEMENT SOLUTION

BACKGROUND OF THE INVENTION

[0001] Field of the Invention

[0002] The present invention relates to automation tools for the application development of fraud management systems, and more particularly to multi-discipline artificial intelligence machines for real-time use on big data.

[0003] Background

[0004] Anyone impressed by the increasing and stunning speeds of computers, or stunned by their unfailing memory capacity, will not be able to find any manifestations of intelligence as long as everything remains purely algorithmic.

[0005] Algorithmic programs are deductive ensembles of successive operations applied in a fixed order. An algorithm brings the computer to repeat, tirelessly and accurately, long suites of logical operations. However, such programs will neither know how to take any initiative, nor will any ever stray away from a fixed line of march.

[0006] Algorithmic programmers must always be able to dictate the precise succession of steps that the target machine is to follow. But, can you cannot ask any expert to predict all of the risk events that may befall a company or business in the next day, month, or year. And so it is for every field that requires the use of experts.

[0007] Since every algorithm requires an exhaustive enumeration the vast majority of industrial problems will be excluded from computer science. Industrial problems in which the resolution requires a minimum of reasoning cannot be transcribed into an algorithmic form. This is also true for programs based on artificial intelligence, like expert systems and conventional object oriented languages. In the case of expert systems, all the possibilities must be predicted in order to write all the possible rules. Such is obviously impossible. In conventional object oriented languages, all possible methods must be forecast, known and programmed for.

[0008] In the case of business, decisions integrate and often concern an important number of variables that dynamically come and go.

[0009] What is needed is a technology that goes beyond algorithmic techniques, ones that know how to resolve very complex problems without needing to be instructed on how to resolve them.

SUMMARY OF THE INVENTION

[0010] Briefly, an artificial intelligence fraud management solution embodiment of the present invention comprises an expert programmer development system to build trainable general payment fraud models that integrate several artificial intelligence classifiers like neural networks, case based reasoning, decision trees, genetic algorithms, fuzzy logic, and rules and constraints. These are further integrated by the expert programmers and development system with smart agents and associated real-time profiling, recursive profiles, and long-term profiles. The trainable general payment fraud models are trained with supervised and unsupervised data to produce an applied payment fraud model. This then is applied by a commercial client to process real-time transactions and authorization requests for fraud scores.

[0011] The above and still further objects, features, and advantages of the present invention will become apparent upon consideration of the following detailed description of specific embodiments thereof, especially when taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

[0012] FIG. 1 is functional block diagram of an artificial intelligence fraud management solution embodiment of the present invention;

[0013] FIG. 2A is functional block diagram of an application development system (ADS) embodiment of the present invention for fraud-based target applications;

[0014] FIG. 2B is functional block diagram of an improved and updated application development system (ADS) embodiment of the present invention for fraud-based target applications;

[0015] FIG. 3 is functional block diagram of a model training embodiment of the present invention;

[0016] FIG. 4 is functional block diagram of a real-time payment fraud management system like that illustrated in FIG. 1 as applied payment fraud model;

[0017] FIG. 5 is functional block diagram of a smart agent process embodiment of the present invention;

[0018] FIG. 6 is functional block diagram of a most recent 15-minute transaction velocity counter; and

[0019] FIG. 7 is functional block diagram of a cross-channel payment fraud management embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0020] FIG. 1 represents an artificial intelligence fraud management solution embodiment of the present invention, and is referred to herein by the general reference numeral 100. Such solution 100 comprises an expert programmer development system 102 for building trainable general payment fraud models 104 that integrate several, but otherwise blank artificial intelligence classifiers, e.g., neural networks, case based reasoning, decision trees, genetic algorithms, fuzzy logic, and rules and constraints. These are further integrated by the expert programmers inputs 106 and development system 102 to include smart agents and associated real-time profiling, recursive profiles, and long-term profiles.

[0021] The trainable general payment fraud models 104 are trained with supervised and unsupervised data 108 and 110 to produce a trained payment fraud model 112. For example, acountholder and historical transaction data. This trained payment fraud model 112 can then be sold as a computer program library or a software-as-a-service applied payment fraud model. This then is applied by a commercial client in an applied payment fraud model 114 to process real-time transactions and authorization requests 116 for fraud scores. The applied payment fraud model 114 is further able to accept a client tuning input 120.

[0022] FIG. 2A represents an application development system (ADS) embodiment of the present invention for fraud-based target applications, and is referred to herein by the general reference numeral 200. Such is the equivalent of development system 102 in FIG. 1. ADS 200 comprises a number of computer program development libraries and tools that highly skilled artificial intelligence scientists and artisans can manipulate into a novel combination of complementary technologies. In an early embodiment of ADS 200 we combined a goal-oriented multi-agent technology 201 for building run-time smart agents, a constraint-based programming tool 202, a fuzzy logic tool 203, a library of genetic algo-
rithms 205, a simulation and planning tool 206, a library of business rules and constraints 207, case-based reasoning and learning tools 208, a real-time interpreted language compiler 209, a C++ code generator 210, a library of data compression algorithms 211, and a database connectivity tool 212.

[0023] The highly skilled artificial intelligence scientists and artisans provide graphical and textual inputs 214 and 216 to a user interface (UI) 218 to manipulate the novel combinations of complementary technologies into a declarative application 220.

[0024] Declarative application 214 is molded, modeled, simulated, tested, corrected, massaged, and unified into a fully functional hybrid combination that is eventually output as a trainable general payment fraud model 222. Such is the equivalent of trainable general payment fraud model 104 in FIG. 1.

[0025] It was discovered by the present inventor that the highly skilled artificial intelligence scientists and artisans that could manipulate the complementary technologies mentioned into specific novel combinations required exceedingly talented individuals that were in short supply.

[0026] It was, however, possible to build and to prove out that ADS 200 as a compiler would produce trainable general payment fraud models 220, and these were more commercially attractive and viable.

[0027] After many years of experimental use and trials, ADS 200 was constantly improved and updated. Database connectivity tool 212, for example, tried to press conventional databases into service during run-time to receive and supply data points in real-time transaction service. It turned out no conventional databases were up to it.

[0028] At the present, an updated and improved ADS shown with general reference numeral 230 in FIG. 2 is providing better and more useful trainable general payment fraud models. ADS 230 is the most recent equivalent of development system 102 in FIG. 1. ADS 230 assembles together a different mix of computer program development libraries and tools for the highly skilled artificial intelligence scientists and artisans to manipulate into a new hybrid of still complementary technologies.

[0029] In this later embodiment, ADS 230, we combined an improved smart-agent technology 231 for building run-time smart agents that are essentially only silhouettes of their constituent attributes. These attributes are themselves smart-agents with second level attributes and values that are able to "call" on real-time profilers, recursive profilers, and long term profilers. Such profilers can provide comparative assessments of each data point with the new information flowing in during run-time.

[0030] The three profilers can thereafter throw exceptions in each data point category, and the number and quality of exceptions thrown across the breadth of the attributes then incoming will produce a fraud risk score that generally raises exponentially with that number of exceptions thrown. Oracle explains in C++ programming that exceptions provide a way to react to exceptional circumstances (like fraud suspected) in programs by transferring control to special functions called "handlers".

[0031] At the top level of a hierarchy of smart agents linked by their attributes are the smart agents for the independent actors who can engage in fraud. In a payment fraud model, that top level will be the cardholders as tracked by the cardholder account numbers reported in transaction data.

[0032] These top level smart agents can call on a moving 15-minute window file that has all the transactions reported to the system in the last 15-minutes. Too much activity in 15-minutes by any one actor is cause for further inspection and analysis.

[0033] ADS 230 further comprises a constraint-based programming tool 232, a fuzzy logic tool 233, a library of advanced neural network algorithms 234, a library of genetic algorithms 235, a simulation and planning tool 236, a library of business rules and constraints 237, case-based reasoning and learning tools 238, a data mining tool 239, a text mining tool 240, a statistical tool 241 and a real-time file system 242.

[0034] The real-time file system 242 is a simple organization of attribute values for smart agent profilers that allow quick, direct file access.

[0035] The highly skilled artificial intelligence scientists and artisans provide graphical and textual inputs 244 and 246 to a user interface (UI) 248 to manipulate the novel combinations of complementary technologies into a declarative application 250.

[0036] Declarative application 250 is also molded, modeled, simulated, tested, corrected, massaged, and unified into a fully functional hybrid combination that is eventually output as a trainable general payment fraud model 252. Such is also the more improved equivalent of trainable general payment fraud model 104 in FIG. 1.

[0037] The constraint-based programming tools 202 and 232 limit the number of possible solutions. Complex conditions with complex constraints can create an exponential number of possibilities. Fixed constraints, fuzzy constraints, and polynomials are combined in cases where no exact solution exists. New constraints can be added or deleted at any time. The dynamic nature of the tool makes possible real-time simulations of complex plans, schedules, and diagnostics.

[0038] The constraint-based programming tools are written as a very complete language in its own right. It can integrate a variety of variables and constraints, as in the following Table.

| Variables: Real, with integer values, enumerated, sets, matrices and vectors, intervals, fuzzy subsets, and more. |
| Arithmetic Constraints: +, -, *, /, |, >, <, <=, >=, interval addition, interval subtraction, interval multiplication and interval division, max, min, intersection, union, exponential, modulo, logarithm, and more. |
| Temporal (Allen) Constraints: Control allows you to write any temporal constraints including Equal, Not-equal, Before, After, Meets, Overlaps, Starts, Finishes, and personal temporal operators such as Disjoint, Start-by, Overlapped-by, Met-by, Finished-by, and more. |
| Boolean Constraints: Or, And, Not, Xor, Implication, Equivalence |
| Symbolic Constraints: Inclusion, Union, Intersection, Cardinality, Belonging, and more. |

[0039] The constraint-based programming tools 202 and 232 include a library of ways to arrange subsystems, constraints and variables. Control strategies and operators can be defined within or outside using traditional languages such as C, C++, FORTRAN, etc. Programmers do not have to learn a new language, and provides an easy-to-master programming interface by providing an in-depth library and traditional tools.

[0040] Fuzzy logic tools 203 and 233 recognize many of the largest problems in organizations cannot be solved by simple yes/no or black/white answers. Sometimes the answers need
to be rendered in shades of gray. This is where fuzzy logic proves useful. Fuzzy logic handles imprecision or uncertainty by attaching various measures of credibility to propositions. Such technology enables clear definitions of problems where only imperfect or partial knowledge exists, such as when a goal is approximate, or between all and nothing. In fraud applications, this can equate to the answer being “maybe” fraud is present, and the circumstances warrant further investigation.

[0041] Tools 204 and 234 provides twelve different neural network algorithms, including Back propagation, Kohonen, ART, Fuzzy ART, RBF and others, in an easy-to-implement C++ library. Neural networks are algorithmic systems that interpret historical data to identify trends and patterns against which to compare subject cases. The libraries of advanced neural network algorithms can be used to translate databases to neurons without user intervention, and can significantly accelerate the speed of convergence over conventional back propagation, and other neural network algorithms. The present invention’s neural net is incremental and adaptive, allowing the size of the output classes to change dynamically.

An expert mode in the advanced application development tool suite provides a library of twelve different neural network models for use in customization.

[0042] Neural networks can detect trends and patterns other computer techniques are unable to. Neurons work collaboratively to solve the defined problem. Neural networks are adept in areas that resemble human reasoning, making them well suited to solve problems that involve pattern recognition and forecasting. Thus, neural networks can solve problems that are too complex to solve with conventional technologies.

[0043] Libraries 205 and 235 include genetic algorithms to initialize a population of elements where each element represents one possible set of initial attributes. Once the models are designed based on these elements, a blind test performance is used as the evaluation function. The genetic algorithm will be then used to select the attributes that will be used in the design of the final models. The component particularly helps when multiple outcomes may achieve the same predefined goal. For instance, if a problem can be solved profitably in any number of ways, genetic algorithms can determine the most profitable way.

[0044] Simulation and planning tool 206 can be used during model designs to check the performances of the models.

[0045] Business rules and constraints 207 provides a central storage of best practices and know how that can be applied to current situations. Rules and constraints can continue to be captured over the course of years, applying them to the resolution of current problems.

[0046] Case-based reasoning 208 uses past experiences in solving similar problems to solve new problems. Each case is a history outlined by its descriptors and the steps that lead to a particular outcome. Previous cases and outcomes are stored and organized in a database. When a similar situation presents itself again later, a number of solutions that can be tried, or should be avoided, will present immediately. Solutions to complex problems can avoid delays in calculations and processing, and be offered very quickly.

[0047] Language interpretation tool 209 provides a constant feedback and evaluation loop. Intermediary Code generator 210 translates Declarative Applications 214 designed by any expert into a faster program 230 for a target host 232.

[0048] During run-time, real-time transaction data 234 can be received and processed according to declarative application 214 by target host 232 with the objective of producing run-time fraud detections 236. For example, in a payments application card payments transaction requests from merchants can be analyzed for fraud activity. In healthcare applications the reports and compensation demands of providers can be scanned for fraud. And in insider trader applications individual traders can be scrutinized for special knowledge that could have illegally helped them profit from stock market moves.

[0049] File compression algorithms library 211 helps preserve network bandwidth by compressing data at the user’s discretion.

[0050] FIG. 3 represents a model training embodiment of the present invention, and is referred to herein by the general reference numeral 300. Model trainer 300 can be fed a very complete, comprehensive transaction history 302 that can include both supervised and unsupervised data. A filter 304 actually comprises many individual filters that can be selected by a switch 306. Each filter can separate the supervised and unsupervised data from comprehensive transaction history 302 into a stream correlated by some factor in each transaction.

[0051] The resulting filtered training data will produce a trained model that will be highly specific and sensitive to fraud in the filtered category. When two or more of these specialized trained models used in parallel are combined in other embodiments of the present invention they will excel in real-time cross-channel fraud prevention.

[0052] In a payment card fraud embodiment of the present invention, the filters 304 are selected by switch 306 to filter through card-not-present transactions, card-present transactions, international transactions, domestic transactions, debit card transactions, credit card transactions, contactless transactions, or other groupings or financial networks.

[0053] A data cleanup process 308 is used to harmonize, unify, error-correct, and otherwise standardize the data coming from transaction data history 302. The commercial advantage of that is clients can provide their transaction data histories 302 in whatever formats and file structures are natural to the provider.

[0054] A data enrichment process 310 computes interpolations and extrapolations of the data and produces as many as two-hundred and fifty data points from the forty or so relevant data points originally provided by transaction data history 302.

[0055] A trainable fraud model 312, like that illustrated in FIG. 1 as trainable general payment fraud model 104, is trained. Such can be imported as a purchased commercial product or library.

[0056] A selected applied fraud model 314, like that illustrated in FIG. 1 as applied fraud model 114, results from the selected training that results from the switch 306 setting and the filters 304. A variety of selected applied fraud models 316-323 represent the applied fraud models 114 that result with different settings of filter switch 306. Each selected applied fraud model 314 will include a hybrid of artificial intelligence classification models represented by models 330-332 and a smart-agent population build 334 with a corresponding set of real-time, recursive, and long-term profilers 336. The enriched data from data enrichment process 310 is fully represented in the smart-agent population build 334 and profilers 336.

[0057] FIG. 4 represents a real-time payment fraud management system 400 like that illustrated in FIG. 1 as applied...
A raw transaction separator 402 filters through the forty or so data items that are relevant to the computing of a fraud score. A process 404 adds timestamps to these relevant data points and passes them in parallel to a selected applied fraud model 406. This is equivalent to a selected one of applied fraud models 316-325 in FIG. 3 and applied payment fraud model 114 in FIG. 1.

During a session in which the time-stamped relevant transaction data flows in, a set of classification models 408-410 operate independently according to their respective natures. A population of smart agents 412 and profilers 414 also operate on the time-stamped relevant transaction data inflows. Each new line of time-stamped relevant transaction data will trigger an update 416 of the respective profilers 414.

Their attributes 418 are provided to the population of smart agents 412. The classification models 408-410 and population of smart agents 412 and profilers 414 all each produce an independent and separate vote or fraud score 420-423 on the same line of time-stamped relevant transaction data. A weighted summation processor 424 responds to client tunings 426 to output a final fraud score 428.

FIG. 5 represents a smart agent process 500 in an embodiment of the present invention. For example, these would include the smart agent population build 334 and profiles 336 in FIG. 3 and smart agents 412 and profiles 414 in FIG. 4. A series of payment card transactions arriving in real-time in an authorization request message is represented here by a random instantaneous incoming real-time transaction record 502.

Such record 502 begins with an account number 504. It includes attributes A1-A9 numbered 505-513 here. These attributes, in the context of a payment card fraud application would include data points for card type, transaction type, merchant name, merchant category code (MCC), transaction amount, time of transaction, time of processing, etc.

Account number 504 in record 502 will issue a trigger 516 to a corresponding smart agent 520 to present itself for action. Smart agent 520 is simply a constitution of its attributes, again A1-A9 and numbered 521-529 in FIG. 5. These attributes A1-A9 521-529 are merely pointers to attribute smart agents. Two of these, one for A1 and one for A2, are represented in FIG. 5. Here, an A1 smart agent 530 and an A2 smart agent 540 are respectively called into action by triggers 532 and 542.

A1 smart agent 530 and A2 smart agent 540 will respectively fetch correspondent attributes 505 and 506 from incoming real-time transaction record 502. Smart agents for A3-A9 make similar fetches to themselves in parallel. They are not shown here to reduce the clutter for FIG. 5 that would otherwise result.

Each attribute smart agent like 530 and 540 will include or access a corresponding profile data point 536 and 546. This is actually a simplification of the three kinds of profiles 336 (FIG. 3) that were originally built during training and updated in update 416 (FIG. 4). These profiles are used to track what is “normal” behavior for the particular account number for the particular single attribute.

For example, if one of the attributes reports the MCC’s of the merchants and another reports the transaction amounts, then if the long-term, recursive, and real-time profiles for a particular account number x shows a pattern of purchases at the local Home Depot and Costco that average $100-$300, then an instantaneous incoming real-time transaction record 502 that reports another $200 purchase at the local Costco will raise no alarms. But a sudden, unique, inexplicable purchase for $1250 at a New York Jewelers will and should throw more than one exception.

Each attribute smart agent like 530 and 540 will further include a comparator 537 and 547 that will be able to compare the corresponding attribute in the instantaneous incoming real-time transaction record 502 for account number x with the same attributes held by the profiles for the same account. Comparators 537 and 547 should accept some slack, but not too much. Each can throw an exception 538 and 548, as can the comparators in all the other attribute smart agents. It may be useful for the exceptions to be a fuzzy value, e.g., an analog signal 0.0 to 1.0. Or it could be a simple binary one or zero. What sort of excursions should trigger an exception is preferably adjustable, for example with client tunings 426 in FIG. 4.

These exceptions are collected by a smart agent risk algorithm 550. One deviation or exception thrown on any one attribute being “abnormal” can be tolerated if not too egregious. But two or more should be weighted more than just the simple sum, e.g., $(1+1)^{-2}$ instead of simply $1+1=2$. The product is output as a smart agent risk assessment 552. This output is the equivalent of independent and separate vote or fraud score 423 in FIG. 4.

FIG. 6 represents a most recent 15-minute transaction velocity counter 600, in an embodiment of the present invention. It receives the same kind of real-time transaction data inputs as were described in connection with FIG. 4 as raw transaction data 402 and FIG. 5 as records 502. A raw transaction record 602 includes a hundred or so data points. About forty of those data points are relevant to fraud detection an identified in FIG. 6 as reported transaction data 604.

The reported transaction data 604 arrive in a time series and randomly involve a variety of account numbers. But, let’s say the most current reported transaction data 604 with a time age of 0:00 concerns a particular account number x. That fills a register 606.

Earlier arriving reported transaction data 604 build a transaction time-series stack 608. FIG. 6 arbitrarily identifies the respective ages of members of transaction time-series stack 608 with example ages 0:73, 1:16, 3:11, 6:17, 10:52, 11:05, 13:41, and 14:58. Those aged more than 15-minutes are simply identified with an age>15:000. This embodiment of the present invention is concerned with only the last 15-minutes worth of transactions. As time passes transaction time-series stack 608 pushes down.

The key concern is whether account number x has been involved in any other transactions in the last 15-minutes. A search process 610 accepts a search key from register 606 and reports any matches in the most 15-minute window with an account activity velocity counter 612. Too much very recent activity can hint there is a fraudster at work, or it may be normal behavior. A trigger 614 is issued that can be fed to an additional attribute smart agent that is included with attributes smart agents 530 and 540 and the others in parallel. Exception from this new account activity velocity counter smart agent is input to smart agent risk algorithm 550 in FIG. 5.

FIG. 7 represents a cross-channel payment fraud management embodiment of the present invention, and is referred to herein by general reference numeral 700.

Real-time multi-channel monitoring uses track cross channel and cross product patterns to cross pollinate
information for more accurate decisions. Such track not only the channel where the fraud ends but also the initiating channel to deliver a holistic fraud monitoring. A standalone internet banking fraud solution will allow a transaction if it is within its limits, however if core banking is in picture, then it will stop this transaction, as we additionally know the source of funding of this account (which mostly is missing in internet banking).

In FIG. 3, a variety of selected applied fraud models 316-323 represent the applied fraud models 114 that result with different settings of filter switch 306. A real-time multi-channel monitoring payment network server can be constructed by running several of these selected applied fraud models 316-323 in parallel.

FIG. 7 represents a real-time multi-channel monitoring payment network server 700, in an embodiment of the present invention. Raw incoming transaction data is selectively filtered for relevancy by a process 702. The resulting relevant data is time-stamped in a process 704. A bus 706 feeds the same such data in parallel line-by-line, e.g., to a selected applied fraud model for card present 708, a selected applied fraud model for domestic 709, a selected applied fraud model for credit 710, a selected applied fraud model for contactless 711, and a selected applied fraud model for jewelry 712. In other words, this example array in FIG. 7 will very tightly analyze domestic contactless credit card transactions that are card-present in a jewelry purchase.

Each selected applied fraud model 708-712 provides exceptions 718-722 to the instant transactions on bus 706 that a weighted summation process 724 can balance according to a client tuning 726. A result 728 can be expected to be highly accurate with low false positives.

In general, embodiments of the present invention are adaptive. Adaptive learning combines three learning techniques. First is the automatic creation of profiles, or smart-agents, from historical data, e.g., long-term profiling. The second is real-time learning, e.g., enrichment of the smart-agents based on real-time activities. The third is adaptive learning carried by incremental learning algorithms.

For example, two years of historical credit card transactions data needed over twenty seven terabytes of database storage. A smart-agent is created for each individual card in that data in a first learning step, e.g., long-term profiling. Each profile is created from the card’s activities and transactions that took place over the two year period. Each profile for each smart-agent comprises knowledge extracted field-by-field, such as merchant category code (MCC), time, amount for an mcc over a period of time, recursive profiling, zip codes, type of merchant, monthly aggregation, activity during the week, weekend, holidays, Card not present (CNP) versus card present (CP), domestic versus cross-border, etc. this profile will highlights all the normal activities of the smart-agent (specific card).

Smart-agent technology has been observed to outperform conventional artificial and machine learning technologies. For example, data mining technology creates a decision tree from historical data. When historical data is applied to data mining algorithms, the result is a decision tree. Decision tree logic can be used to detect fraud in credit card transactions. But, there are limits to data mining technology. The first is datamining can only learn from historical data and it generates decision tree logic that applies to all the cardholders as a group. The same logic is applied to all cardholders even though each merchant may have a unique activity pattern and each cardholder may have a unique spending pattern.

A second limitation is decision trees become immediately outdated. Fraud schemes continue to evolve, but the decision tree was fixed with examples that do not contain new fraud schemes. So stagnant non-adapting decision trees will fail to detect new types of fraud, and do not have the ability to respond to the highly volatile nature of fraud.

Another technology widely used is “business rules” which require expert to write rules, e.g., if-then-else logic. The most important limitations are that business rules require writing rules that are supposed to work for whole categories of customers. This requires the population to be sliced into many categories (students, seniors, zip codes, etc.) and ask experts to think about rules that will be applied to all the cardholders of the category. How could the US population be sliced? Even worse, why would all the cardholders in a category all have the same behavior? It is obvious that business rules has limits, and poor detection rates with high false positives. What should also be obvious is the rules are outdated as soon as they are written because they don’t adapt at all to new fraud schemes or data shifts.

Neural network technology also limits, it uses historical data to create a matrix weights for future data classification. The Neural network will use as input (first layer) the historical transactions and the classification for fraud or not as an output. Neural Networks only learn from past transactions and cannot detect any new fraud schemes (that arise daily) if the neural network was not re-trained with this type of fraud. Same as datamining and business rules the classification logic learned from the historical data will be applied to all the cardholders even though each merchant has a unique activity pattern and each cardholder has a unique spending pattern.

Another limit is the classification logic learned from historical data decision trees is outdated the same day of its use because the fraud schemes changes but since the neural network did learn with examples that contains this new type of fraud schemes, it will fail to detect this new type of fraud it lacks the ability to adapt to new fraud schemes and do not have the ability to respond to the highly volatile nature of fraud.

Contrary to previous technologies, smart-agent technology learns the specific behaviors of each cardholder and create a smart-agent that follow the behavior of each cardholder. Because it learns from each activity of a cardholder, the smart-agent updates the profiles and makes effective changes at runtime. It is the only technology with an ability to identify and stop, in real-time, previously unknown fraud schemes. It has the highest detection rate and lowest false positives because it separately follows and learns the behaviors of each cardholder.

Smart-agents have a further advantage in data size reduction. Once, say twenty-seven terabytes of historical data is transformed into smart-agents, only 200-gigabytes is needed to represent twenty-seven million distinct smart-agents corresponding to all the distinct cardholders.

Incremental learning technologies are embedded in the machine algorithms and smart-agent technology to continually re-train from any false positives and negatives that occur along the way. Each corrects itself to avoid repeating the same classification errors. Data mining logic incrementally changes the decision trees by creating a new link or updating the existing links and weights. Neural networks update the weight matrix, and case based reasoning logic
updates generic cases or creates new ones. Smart-agents update their profiles by adjusting the normal/abnormal thresholds, or by creating exceptions.

[0087] Although particular embodiments of the present invention have been described and illustrated, such is not intended to limit the invention. Modifications and changes will no doubt become apparent to those skilled in the art, and it is intended that the invention only be limited by the scope of the appended claims.

1. An automation tool for the application development of multi-discipline artificial intelligence fraud management systems, comprising:
   an expert programmer development system with tools and software libraries for building trainable general payment fraud models that are constructed of and integrate several artificial intelligence classifiers, including smart agents, neural networks, case-based reasoning, decision trees, and business rules;
   wherein, said trainable general payment fraud models are capable of being trained with historical transaction data that causes:
   an initial population of smart agents and associated profiles to be generated,
   an initial set of neural networks to assume a beginning weight matrix,
   an initial decision tree to be structured from data mining logic,
   an initial case-based reasoning set to be structured, and
   an initial set of business rules to be fixed;

2. The automation tool of claim 1, wherein said trainable general payment fraud models further comprise:
   an incremental learning technology embedded in a runtime machine algorithm and smart-agent technology able to continually re-train said artificial intelligence classifiers using false positives and negatives that occur during use.

3. The automation tool of claim 2, wherein said incremental learning technology further comprises:
   data mining logic for incrementally changing the initial decision trees by creating new links or updating its existing links and weights.

4. The automation tool of claim 2, wherein said incremental learning technology further comprises:
   means for the initial neural networks to have their weight matrix updated.

5. The automation tool of claim 2, wherein said incremental learning technology further comprises:
   means for the initial case-based reasoning logic to update its generic cases or create new ones.

6. The automation tool of claim 2, wherein said incremental learning technology further comprises:
   means for the initial population of smart-agents to self-update their profiles and to adjust their normal/abnormal thresholds or by creating exceptions.

* * * * *