Distinguishing the Unexplainable from the Merely Unusual: Adding Explanations to Outliers to Discover and Detect Significant Complex Rare Events

Ted E. Senator, Henry G. Goldberg, Alex Memory
SAIC
{senatort, goldberghg, memoryac}@saic.com

ABSTRACT
This paper discusses the key role of explanations for applications that discover and detect significant complex rare events. These events are distinguished not necessarily by outliers (i.e., unusual or rare data values), but rather by their *inexplicability* in terms of appropriate real-world behaviors. Outlier detection techniques are typically part of such applications and may provide useful starting points; however, they are far from sufficient for identifying events of interest and discriminating them from similar but uninteresting events to a degree necessary for operational utility. Other techniques that distinguish anomalies from outliers, and then enable anomalies to be classified as relevant or not to the particular detection problem are also necessary. We argue that explanations are the key to the effectiveness of such complex rare event detection applications, and illustrate this point with examples from several real applications.

Categories and Subject Descriptors
H.2.8 [**Database Applicaitons**]: Database applications – *data mining*; I.5.2 [**Pattern Recognition**]: Design Methodlogy – *classifier design and evaluation, feature evaluation and selection*.

General Terms
Security

Keywords
Complex Event Detection, Anomaly Detection, Outlier Detection, Explanation

1. INTRODUCTION
In many real applications, the problem to be solved is the discovery and detection of complex rare events. Examples of such applications include detection of money laundering [9], detection of insider trading [2], and detection of insider threats [5], [10]. These applications typically combine two functions: (1) discovery, which means the identification of previously unrecognized event types, and (2) detection, which means the identification of instances of event types. Complex rare events are characterized by contextual combinations of specific actions, often by multiple agents. They typically occur very infrequently. These events manifest in multiple databases, many of which may or may not be initially – or even eventually – observable or available. Such events may result from multiple agents executing many different types of activities simultaneously, all of which are interleaved in the observed data. Often the observed data are insufficient to distinguish between legitimate actions and actions of interest; additional data are required to make this determination by explaining the observed patterns of activity in the available data.

ODD'13 August 11, 2013 Chicago, IL USA.

2. KEY IDEA
The key idea of this paper is that detection and discovery of complex rare events involves three related but distinct levels of abstraction, and that explanations are the essential mechanism to transform between these levels, because events of interest are characterized not by their rarity but rather by their *inexplicability*.[1]

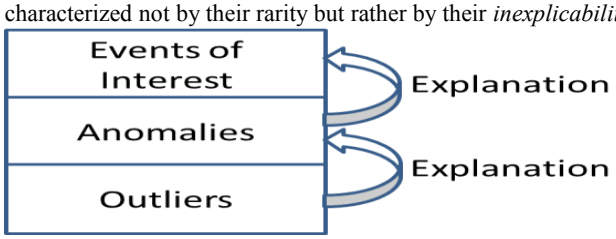

Figure 1: Explanations Enable Transformations Between Outliers, Anomalies and Events of Interest

Such explanations may be provided both implicitly and explicity by the underlying data, models and features incorporated in the system design and by what is presented to a human analyst, respectively.

Explanation provides the ability to infer underlying intent and allows the segmenting of intermingled actions to identify those related to a particular event of interest. This can be as simple as a large number of instances of a condition-action pair, i.e., an anomaly consisting of a set of outliers that would be unlikely to have occurred without intent – from which one can infer the existence of the intent to commit the action when the specified condition occurs. An example of such an anomaly would be repeated personal trading by a stockbroker "ahead" (i.e., in the time period preceding) trades for a customer. The stockbroker would be using his foreknowledge of the customer trade and its likely impact on the price of the security to obtain a profit for himself. A more complex anomaly might be a partial match to a set of observations that would result from execution of a plan to engage in some improper activity. We refer to the behaviors characteristic of such plans as "scenarios" and the corresponding set of observations as "patterns." [8] An example of such a scenario might be an authorized user of a computer system who searches for valuable corporate information in areas distinct from

[1] This idea is related to the concept of "interestingness" in the data mining literature. A specific set of data is "interesting" in these applications because of the real world behavior suggested by its explanation.

his current job responsibilities, downloads such information to removeable media, and then takes an extended vacation.

Three levels of abstraction that we propose and have found useful in various applications we have designed and implemented are: (1) outliers, (2) anomalies, and (3) events of interest, as depicted in figure 1. We define outliers as unusual (combinations of) data points that are statistically rare but may or may not result from the same generative processes as other data points; anomalies as data points that have resulted from a distinct generative process[2] that may or may not appear in the data as outliers; and events of interest as anomalies that are caused by some type of behavior in the real world that is of concern for the application. It is important to note that outliers are always defined with respect to some baseline population and the distribution of values of some (set of) features of entities comprising said population in the database; anomalies are defined with respect to a generative process that results in the observed data; and events of interest are defined by the external needs of an application (e.g., a rule or regulation that may be violated, or a phenomenon of concern for some other reason, such as a combination of symptoms requiring treatment).

For example, in an application to detect insider trading, an outlier might be an unusually large trade, for a particular trader in the context of his or her trading history, in a particular security, on behalf of a particular owner, by a trader compared to other traders during the same timer period, etc. In a public health surveillance application, outliers might be a significant increase in emergency room admissions or in a particular complaint or symptom. In an insider threat detection application, an outlier might be an unusually large number of file transfers to removeable media.

In the insider trading domain, a corresponding anomaly would be a large trade that precedes a material news event and results in a significant profit. It is considered an anomaly because it is generated by a trader reacting to prohibited insider information, not to normal market information. Such an event might be benign – it could simply be a coincidence – or it might be improper, depending on the explanation. An explanation of an improperly caused anomaly could be the connections or communications between the beneficial owner of the profitable trade and a source of the inside information. (A legitimate explanation might be one in which the beneficial owner had recently acquired a large sum of cash and was conducting numerous trades, of which the one in question happened to coincide with an opportunity to make a large immediate profit.) A repeated pattern of such trades would also be an anomaly, providing an analyst with the ability to infer the existence of the unobserved communication of inside information to the trader, such unobserved communication serving as the explanation of the anomaly.

3. EXPLANATIONS

We consider several types of explanations and show examples of how they transform from outliers to anomalies to events of interest. We group explanations into two categories, depending on whether they aid in the transformation from outliers to anomalies or from anomalies to events of interest.

3.1 Explaining Outliers

Because outliers are defined statistically, in terms of where a particular (set of) data points appear on a distribution of values of particular features on a specified population, the explanations of such outliers must depict where the data ponts lie on these distributions with respect to the population. This is not sufficient, however, for several reasons. First, many users are not trained in statistics and will not appreciate an explanation in terms of p-values, t-tests, and the like. Second, and at least as important, the selection of both the underlying population – and of the definition of the entities of interest – typically involves many implicit and explicit assumptions and choices regarding comparison baselines that affect the recognition of outliers. Third, an explanation of the statistical significance and location of an outlier does not tell a user how likely such an outlier is to suggest the existence of the event of interest. Finally, there is the multiple comparison problem – which, while correctable with appropriate statistical methods, can still mislead users of a detection system. We illustrate the second and third of these points with examples, as the first and fourth require no further explanation.

3.1.1 Comparison Baselines

Consider an application that aims to detect significant events in financial time-series data. Such an application might look for trades that are unusual. This can mean many things. Even if we specify the features of interest (e.g., total dollar amount of a trade, frequency of trading, time of day, etc.) – which is not the subject of this paragraph or this paper – we have many other choices. For any aggregate feature, we compare its value over some time period to its value over preceding time periods. For any entity – individual or aggregate – we want to compare the feature values to those of other similar entities, for some definition of "similar." To analyze an individual trade, we may want to compare it to other trades by the same person in the same security, in similar securities, or in other securities. We may want to compare time-based features such as average trading amount per day or week

We have identified three time periods that must be specified for any comparison: (1) the temporal-extent over which a feature is computed for a particular entity; (2) the look-back period over which the same feature is computed in order to establish the distribution of values of that feature; and (3) the granularity of the computations during the look-back period. For example, consider again a financial time series application that uses the price-volatility of a security as a feature. We might look for unusual volatility numbers based on hours or days; we might compute the distribution of historical volatility by looking back over one, three, or six months; and we might use the daily or weekly prices as the basis of this computation.

Similarly, we need to specify the population of entities against which a comparison occurs. We might, for example, compare an individual to his or her own behavior, to people in his or her community (i.e., people to whom he or she is connected according to some type of network structure), or to peers (i.e., people with similar job descriptions and functions).

Once we have made the above choices, we then have to decide whether to compare

Table 1: Outlier Detection Data Structure

ResultScore	ResultMetadata
runID (KEY)	runID
flowID (KEY)	flowID
alfoID (KEY)	algoID
dataTypeID (KEY)	dataTypeID (KEY)
nodeID	entityXtent (KEY)
rawScore [optional]	featureID [optional]
normScore	EntityTemp
rankedScore	popXtent
Rank	popSubXtent
endDate	popTemp
analystScore	scoreMean
hasAnalystScore	scoreStdev
analystUpdateDate	scoreCount
	parameters

[2] This concise description in terms of generative processes was suggested by Andrew Emmott of Oregon State University.

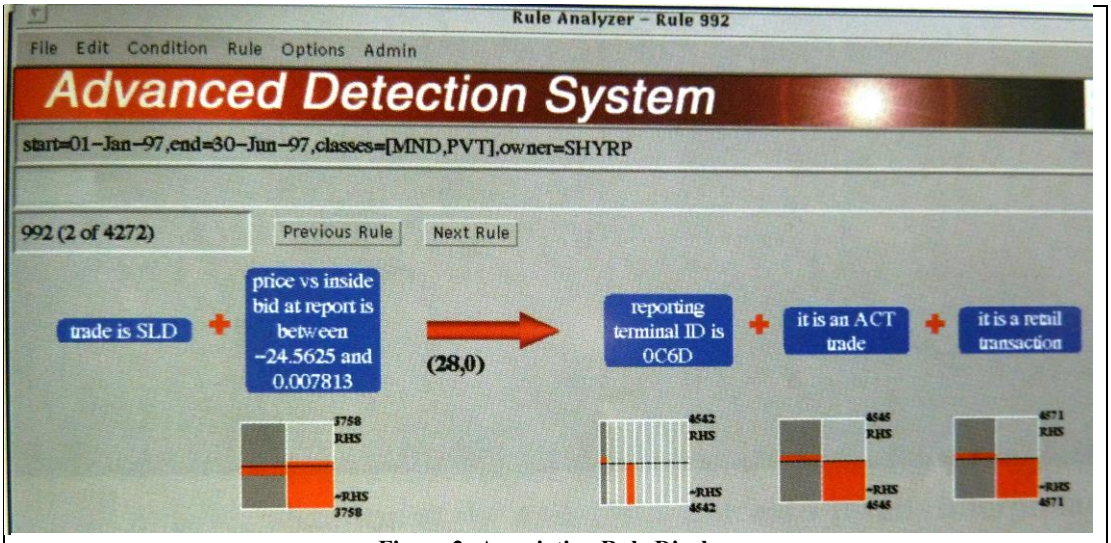

Figure 2: Association Rule Display

absolute values or normalized values. (We may explore a number of possibilities as we design a detection system, to determine which features are the most useful.) If we choose normalized values, we have to determine the basis of normalization. For example, do we normalize the number of communication events of an individual by the number of other individuals with which he has communicated? Do we normalize based on where someone's feature value lies on a distribution based on his/her community and/or peers? Should we normalize based on absolute numbers, units of standard deviation for a particular feature, or percentiles?

However the above choices are made, they need to be communicated explicitly to a human analyst as part of the explanation of an outlier. If multiple versions of such choices are made, then outliers need to be explained in the context of all of these for a full understanding.

3.1.2 How Unusual is an Outlier?

An outlier must be explained not only in terms of its likelihood but also in terms an analyst can understand. Analysts are familiar with their underlying data, so an explanation in these terms has proven effective. It is useful to provide simple visualizations that show where outliers lie according to the distribution of all values of features comprising the outlier. Obviously, such visualizations are limited to two – and occasionally with advanced techniques three – dimensions on a screen, but additional dimensions can be illustrated using color, pattern, iconology, etc. The key idea is that enabling an analyst to see an outlier in the context of the distribution of values of all the relevant features enables the analyst to determine if the outlier is significant. Showing an analyst the number of instances in the data with similar feature values and their resulting interpretation is a useful technique for explaining such outliers.

Figure 2, taken from reference [7] contains an example of such a visualization. This visualization of an association rule, which consists of multiple conditions on both the left and right hand sides, states each condition using pre-stored natural languge text augmented with specific variable values (in the blue boxes in the figure). The bar graphs appearing below the blue boxes depict the number of trades or quotes for which the rule holds or does not, above and below the horizontal axis, respectively, and shows all values of the variable associated with the particular condition across the horizontal axis. The arrow is labeled with the number of instances in the database on the left hand side for which the right hand side does and does not hold. Additional context is provided by the ability to click on the arrow and bring up a table of the raw data summarized in the rule. This visualization enables analysts to understand the context of any prediction made by the rule and evaluate whether the outlier is truly anomalous.

While visualizations are useful for explaining outliers to human

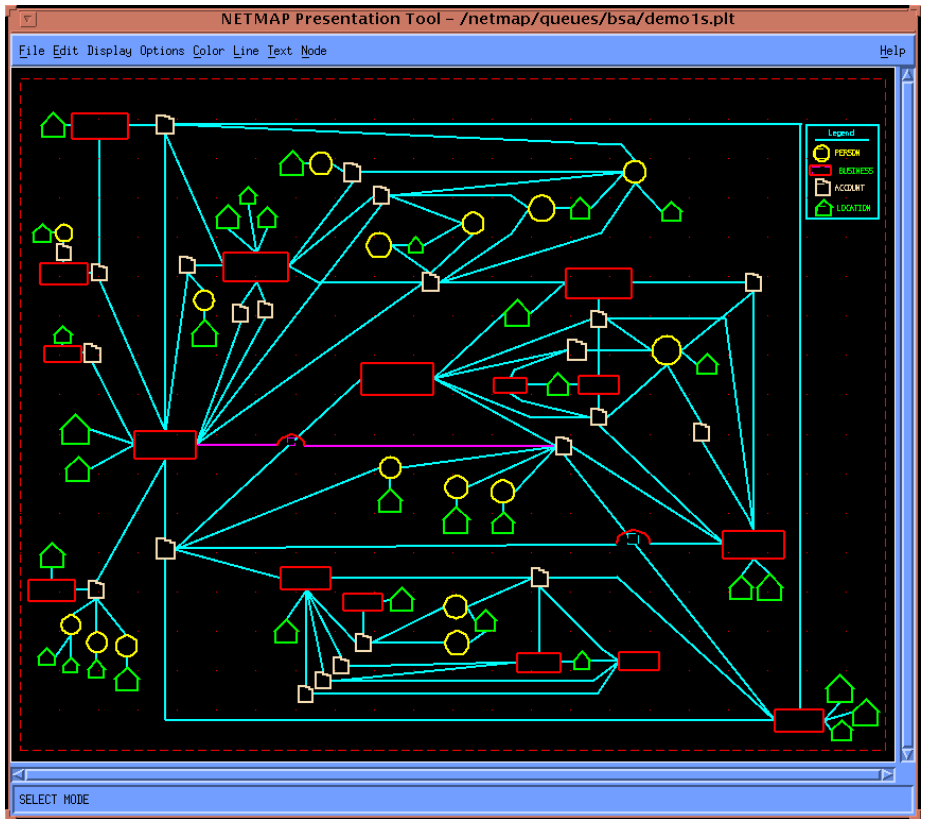

Figure 3: Money Flow Visualization

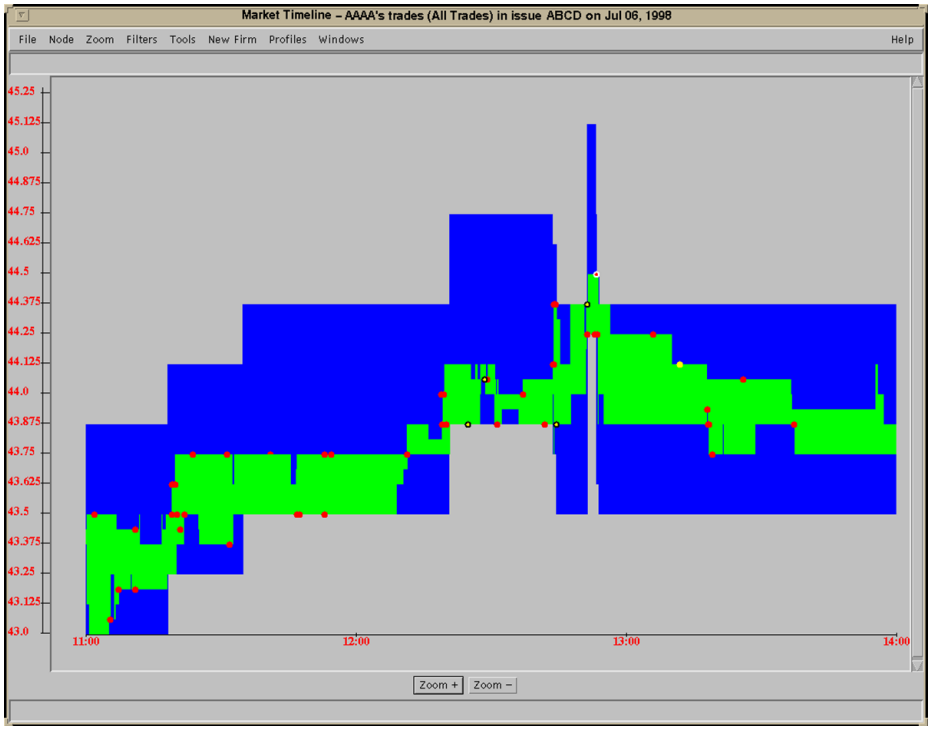

Figure 4: Trade and Quote Visualization

analysts, an automated system requires that detected outliers be stored and available for further analysis. Reference [5] describes a data structure that captures the results of outlier detection, as depicted in table 1. For each run, we allow outlier detection algorithms to compute a raw score, an ordered ranking of all entities, a normalized score, and percentile (ranked) scores. The raw scores may be on different scales and difficult to combine across algorithms. Normalized scores and ranks enable us to compare scores across algorithms. Distributional information such as mean and standard deviation allows us to determine degrees of anomaly.

3.2 Explaining Anomalies

Anomaly explanation differs from outlier explanation by referring to models of processes that may have given rise to the observed data. These models may be based on scenarios derived from domain knowledge or on abstract features of the domain. For example, in the systems described in references [3] and [5], scenarios of behavior are defined and translated to patterns of actions that would be matched in the observed data. The occurrence of matches to these patterns strongly suggests that the behavior modeled by the scenario has occurred. Such scenario-based models may involve multiple interacting agents performing distinct types of actions in a particular temporal sequence or with particular durations.

Other generative models may refer less directly to specific actions in a domain, and capture instead relevant abstractions. For the money laundering detection application discussed in reference [9], the most relevant domain concepts involved money flows between people, businesses and accounts. The visualization depicted in figure 3 was useful to explain a particular anomaly to analysts. In this picture, green "house" shaped icons refer to addresses, yellow circles to people, beige cut-off rectangles to accounts, and red rectangles to businesses. This diagram depicts in the center, three people at three different addresses who share an account which is used by three businesses, which share one other account, and so on. The diagram was used to explain this money-laundering case not only to the analysts management chain, but ultimately to a grand jury. While other aspects of the case, such as the timing and amounts of money flows between the people, accounts, and businesses, is not depicted in the diagram, combined with the accompanying text and briefing, it captured enough information to convince appropriate authorities to proceed. An interesting phenomena that we discovered when developing this visualization is that the explanation that helped an analyst understand the data turned out to be the same visualization that enabled him to explain it to his management chain and to external organizations responsible for further prosecution of the suspected violation.

In the case of the stock market example discussed in reference [3], [7], and [8], the most relevant domain abstractions had to do with temporal sequences of quotes and trades by different market participants. Figure 4 is an actual screen shot of a visualization that captures these abstractions. The x-axis shows the time and the y-axis the price of a particular security. Trades are depicted by dots; quotes by the market maker of interest by the blue band; and the "inside quote" – i.e., the best available bid and ask prices, by the green band.

Other relevant abstractions may be captured as well. Reference [1] uses typical graph structures found to be characteristics in many domains to indentify situations where such structures appear anomalous. Distributional comparisons, such Benford's law (which captures the empirical distribution of the "first digit" in many real sources of data), may also serve as the basis for explaining anomalies.

Finally, a source of explanations for observed anomalies may be additional data types. In the example of insider trading cited earlier, we described two situations which could result in significant, profitable trading in advance of material news. The same event could have resulted from either of two generative processes: (1) the trader had inside information, or (2) the trader had money to invest or losses to cover and bought or sold the stock without knowledge of the insider information. Additional data is required to infer which is the true explanation: call logs, lists of company officers with inside access, names of friends and family, other trading patterns of the trader in questions, etc. In the example of the insider who copies proprietary data, we noted that the files copied were not related to his normal work area. This is an inference which must be made in order to increase our confidence that the event is anomalous and of interest, not just unusual. We can make it either by looking at his other activities in the recent past, or through reference to additional, supplementary data, such as project assignments.

4. ANOMALY DETECTION LANGUAGE

Specifying the functional flow of outlier and anomaly detectors required for a real application, and capturing the multiplicity of choices for baselines and extents, was facilitated by the development and use of a visual anomaly detection lanauge. [4]

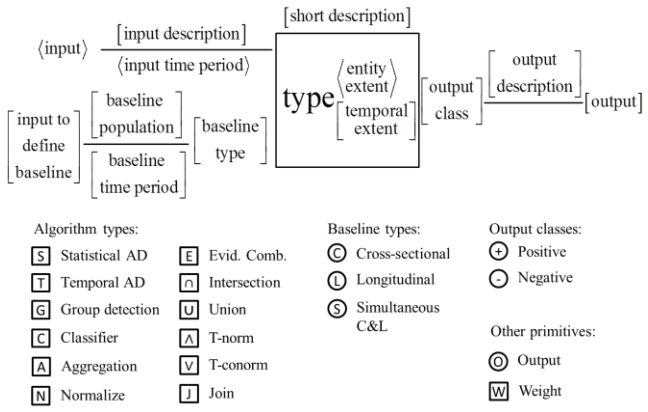

Figure 5: Anomaly Detection Language Syntax

Traditional data flow diagrams cannot express these designs concisely, so we developed a visual anomaly detection language that enables the expression of such combinations of methods, data, baselines, and detection extents. While developed for insider threat detection, the language itself is domain-independent and may be applied to other domains. The language specifies the extent of the entities to be detected (e.g., individual users or groups of users) combined with the temporal extent of potential anomalies. Inputs to these expressions are transactional records of user activity, and outputs are scores on these user-temporal extents.

The syntax of the language is shown in Figure 5; required arguments are in <angle brackets> and optional arguments in [square brackets]. Records are passed along horizontal lines from left to right. Component types are specified by symbols. Entity and temporal extents are super- and sub-scripts, respectively, of component type.

Components may be statistical (denoted by the symbol S) or temporal (T); the latter indicating detectors specialized for anomalies in temporal patterns. Group detectors (G) discover communities of entities, which can be used as baseline populations. Classifiers (C) place input records into classes, which may also be used as baseline populations, or for filtering or partitioning records in general. The classes may be hard, meaning that each record is put into exactly one class, or mixed, in which case a record may be a member of more than one class, possibly to varying degrees. Classifiers might be implemented using a machine-learning method or may be a simple filter based on a lookup on a record. Similarly, aggregators (A) group records with some shared characteristic and summarize their values, e.g., roll-up emails from the same sender to a single record having the count of emails as a new feature; aggregators derive new features from existing ones in this way. Another way to transform features is with a normalizer (N), e.g., rescale real-valued features to the unit interval. Finally, if given a baseline, records are classified and normalized with respect to that baseline.

When sets of records are joined and contain different values for the same feature, ***and*** and ***or*** (∧, ∨) can combine those values, e.g., implement with a t-norm and t-conorm to combine unit-interval values. Evidence combiners (E) also combine values but are more general than ∧ and ∨. And, when no combinations are necessary, union (∪) and intersection (∩) perform the expected operations on input records.

If a baseline is provided, a baseline type specifies how the baseline is used by the component and is indicated by a symbol inside a small circle to the left of the component to which the baseline input connects. With a cross-sectional baseline (C), entity extents are compared to others within the same temporal extent. In contrast, with a longitudinal baseline (L) each entity will be handled individually, and different temporal extents for that entity are compared to one another. A simultaneous baseline (S) combines the first two and compares each input extent to all baseline extents. If a baseline or input time period is not specified, this means that the two cover all available time periods.

Whenever a component may output more than one output class of records, e.g., a binary classifier has (+) and (−) output classes, they should be placed to the right of the component inside circles connected to output lines, unless only one class of output is needed and that class is clear from context, in which case the output class can be omitted.

Weights are scalars in the unit interval used to transform features – usually scores – and are drawn as the letter w inside a rectangle. The type of weighting should be put in a description above the rectangle. Finally, the output of the system is drawn as the letter "O" inside a circle.

Figure 6 uses the anomaly detection language to specify a system for targeting an Intellectual Property (IP) Thief Ambitious

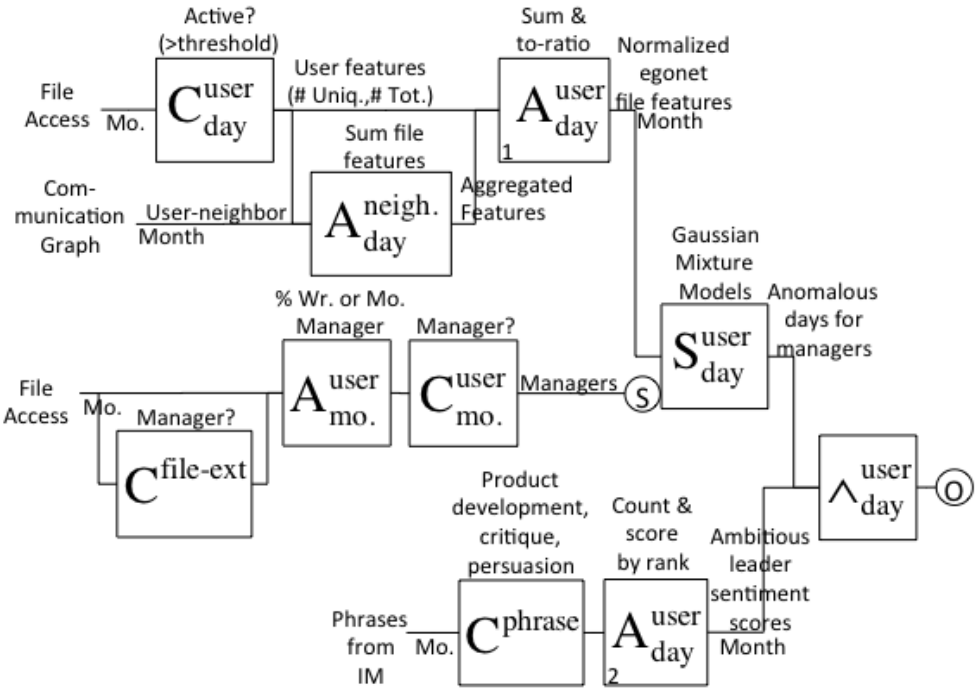

Figure 6: IP Thief – Ambitious Leader Scenario Diagram

Leader scenario in the insider threat detection domain, in which we find a leader of a group of insiders who each steal a few files to be inconspicuous. To counter their strategy, we combine the file activity from the neighbors surrounding each user – known as an egonet – in the IM communication graph, making the leader more anomalous.

We start by filtering user-days to those with sufficient file activity (C_{day}^{user}), then join those records with the IM user-neighbor adjacency list and sum up the features for each "neighbor" $(A_{day}^{neigh.})$. We next add that total for each user to the user's own features and convert the feature totals into ratios $_1(A_{day}^{user})$ that can be compared across egonets of different sizes, e.g. number of unique files to number of all files.

To limit the baseline population to users fitting the profile of a leader, we keep the users $(C_{mo.}^{user})$ with a high fraction of file accesses $(A_{mo.}^{user})$ fitting the manager role according to file extension $(C^{file-ext})$ and use this set as a simultaneous baseline to score (S_{day}^{user}) each user-day.

As an additional indicator, we count $_2(A_{day}^{user})$ phrases seen in IMs between users that fit the scenario (C^{phrase}) and finally combine (Λ_{day}^{user}) with the anomaly scores.

5. COMPLEX EVENT DETECTION SYSTEMS WITH EXPLANATIONS

Real complex event detection systems are multi-layered, consisting of a series of classifiers, each of which is more accurate, primarily because of the availability of additional data to explain outliers or anomalies detected at an earlier stage, and which are biased towards minimizing false negatives at the early stages and minimizing false positives at the later stages [6]. At present, explanations must be provided by humans, sometimes at great cost in terms of investigating false positives or collecting additional data, and requiring significant human analyst involvement and judgement. Adding automated explanation capabilities will enable systems to more accurately distinguish events of interest from other anomalies and enable human analysts to focus their efforts on follow-up investigations and actions for those situations most demanding and worthy of their attention.

Our vision for applications that can be effective at detecting significant complex rare events involves the development of automated explanation techiques to reduce false positive ratios by enabling the transformation of outliers to anomalies and then to events of interest. Techniques that appear promising for such automated explanations might include plan generation, providing the ability to generate plans to accomplish a high-level goal specified in domain terms and translate such plans into an appropriate set of features and baselines that may be present in the data, as well as techniques that could infer potential plans as possible explanations of observed data. Development of ontologies of anomaly types in terms of domain characteristics will be another necessary development to enable this vision.

6. ACKNOWLEDGMENTS

This work contains ideas developed over the course of many years by the authors while employed by several different organizations. It has benefitted from discussions with colleagues in all these organizations. The content of the information in this document does not necessarily reflect the position or the policy of any of these organizations, including the US Government, and no official endorsement by any of these organizations should be inferred.

7. REFERENCES

[1] Chau, D. H., Kittur, A., Hong, J. I. and Faloutsos, C. 2011. Apolo: making sense of large network data by combining rich user interaction and machine learning. In CHI 2011.

[2] Goldberg, Henry G., J. Dale Kirkland, Dennis Lee, Ping Shyr, Dipak Thakker: The NASD Securities Observation, New Analysis and Regulation System (SONAR). IAAI 2003: 11-18

[3] Kirkland, J. Dale, Ted E. Senator, James J. Hayden, Tomasz Dybala, Henry G. Goldberg, Ping Shyr: The NASD Regulation Advanced-Detection System (ADS). AI Magazine 20(1): 55-67 (1999)

[4] Memory, A. et al. 2013. Context-Aware Insider Threat Detection. *Proceedings of the Workshop on Activity Context System Architectures.* Bellevue, WA.

[5] Senator, Ted E., Detecting Insider Threats in a Real Corporate Database of Computer Usage Activity, KDD 2013, to appear.

[6] Senator, T. E. 2005. Multi-stage Classification. In ICDM '05 Proceedings of the Fifth IEEE International Conference on Data Mining Pages 386-393. IEEE Computer Society Washington, DC.

[7] Senator, Ted E. et. al., The NASD Regulation Advanced Detection System: Integrating Data Mining and Visualization for Break Detection in the NASDAQ Stock Market. In *Information Visualization in Data Mining and Knowledge Discovery*, Usama Fayyad, Georges G. Grinstein, and Andreas Weirse, eds., Academic Press (2002)

[8] Senator, Ted E. Ongoing management and application of discovered knowledge in a large regulatory organization: a case study of the use and impact of NASD Regulation's Advanced Detection System (ADS). KDD 2000: 44-53

[9] Senator, Ted E., Henry G. Goldberg, Jerry Wooton, Matthew A. Cottini, A. F. Umar Khan, Christina D. Klinger, Winston M. Llamas, Michael P. Marrone, Raphael W. H. Wong: The Financial Crimes Enforcement Network AI System (FAIS) Identifying Potential Money Laundering from Reports of Large Cash Transactions. AI Magazine 16(4): 21-39 (1995)

[10] Young, W. T et al. 2013. Use of Domain Knowledge to Detect Insider Threats in Computer Activities. Workshop on Research for Insider Threat, *IEEE CS Security and Privacy Workshops*, San Francisco, May 24, 2013.

`