Eliciting User Expectations for Data Behavior via Invariant Templates

Orna Raz   Rebecca Buchheit   Mary Shaw
Philip Koopman   Christos Faloutsos

January, 2003

CMU-CS-03-105

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

People expect software that they use for everyday purposes to be dependable enough for their needs. Usually, they can tolerate some failures, provided they can notice and recover from problems. Many dependability enhancement techniques rely on failure detection. Detection requires a model of proper behavior, preferably in the form of specifications. However, the specifications of everyday software are often incomplete and imprecise.

This research uses machine learning techniques to refine the model and accommodates the necessary human participation. We propose a template mechanism to bridge between user expectations and techniques output. The result is an analyzable model of proper behavior that may serve as a proxy for missing specifications. We use the model to detect semantic anomalies—data behavior that is outside the user's expectations.

We test our template mechanism on truck weigh-in-motion (WIM) data. A domain expert interacts with this mechanism to set up the model of proper behavior. We then analyze the usefulness of this model for anomaly detection and show it is useful. We also compare the model to existing documentation and show how this gave the expert insights about the WIM system.

This research is supported by the National Science Foundation under Grant ITR-0086003, by the Sloan Software Industry Center at Carnegie Mellon University, and by the High Dependability Computing Program from NASA Ames cooperative agreement NCC-2-1298.

Keywords: software engineering, semantic anomaly detection, invariant templates, user expectations

1 Introduction

Modern information resources include dynamic data feeds that provide a time-ordered sequence of observations, often derived from multiple sources, including physical sensors. Dynamic data feeds remain under the control of their providers, have incomplete and imprecise specifications, and may dynamically change. Examples are the truck weigh-in-motion data used in this paper, quotes for a stock, and market prices for an item. The utility of dynamic data feeds would greatly increase with increased dependability.

We concentrate on *semantic anomalies*—cases in which a data feed provides syntactically well-formed data in a timely manner, yet this data is incorrect, inconsistent, or otherwise unreasonable. These are the cases in which the data is outside the model of proper behavior. Human intervention is essential in setting up this model because proper behavior is determined by the semantics of the data. Further, acceptable behavior may depend on the user and the goal the user has for the data. For such everyday elements, the user's goal may differ from the providers' goal.

We rely on the user of a data feed and the data that the feed provides in producing a model of proper behavior. We assume the user has a purpose for using the data feed and consequent expectations for the behavior of the data feed. However, these expectations are informal and imprecise, though they are reasonably accurate. Fortunately, we can use existing statistical and machine learning techniques to formally characterize various aspects of the data, in the form of adaptive invariants. *Adaptive invariants* are not only static invariants but also relations and values that may change as the behavior of the data changes. In our work, we use "invariants" to mean "adaptive invariants".

We show how to use invariant templates to make user expectations precise, thus enabling analysis. The accuracy we achieve is commensurate with the user's understanding of the data feed and underlying system. We only do limited computation with the resulting predicates, so propagating inaccuracy is not an issue. We elicit knowledge about the structure of the data feed in the form of skeleton predicates, or *templates*, that express properties the different techniques can recognize. We do so by using the techniques to infer invariants about the normal behavior of the data feed and then asking the user to choose from the list of inferred invariants. We give the user immediate feedback about the consequences of her choices by evaluating the invariants over the data to detect anomalies—data that falsifies an invariant.

In previous work [26], three of the authors proposed a framework for inferring adaptive invariants about the behavior of dynamic data feeds, using and adapting existing techniques for the inference. That work concentrated on the usage stage of the framework and demonstrated that it is possible to use the invariants as proxies for missing specifications to perform anomaly detection in the data feed (for a single data feed with numeric valued attributes, in the context of stock market tickers, using techniques Daikon and Mean). Here we build on that work and extend it as follows: (1) We concentrate on the setup stage of our invariant inference framework. In particular, we rely on a template mechanism to effectively guide the human attention required in setting up a model of proper behavior. (2) We augment the invariant inference with additional techniques. (3) We test our work on real-world truck WIM data.

Truck weigh-in-motion (WIM) data is common in the transportation domain. A scale located in the traffic lane of the road weighs every axle that passes over it. It records the weight on the axle, the time, the lane the axle was in, and any error codes. Software processes this data to map axles to vehicles, estimate the speed and length of the inferred vehicles, calculate a measure of load on an axle called ESAL (Equivalent Standard Axle Load), classify the vehicle type, eliminate passenger cars from the data, and filter out unreasonable values. A WIM system records all this data for each truck that passes over the scale. Civil Engineers use the data for analyses such as road wear.

We have a domain expert (the second author) set up a model of proper behavior for the WIM data through interactions with our template mechanism. We then use this model for anomaly detection and compare it to existing documentation. We show that the template mechanism is effective; we measure effectiveness both by the insights the user gains (the usefulness of the process) and the detection and misclassification rates (the usefulness of the resulting model).

Section 2 introduces the template mechanism and shows that templates can characterize a wide variety of existing invariant inference techniques. Sections 3 and 4 describe our validation experiment and its results, respectively, and show that templates are effective in making user expectations precise. Section 5 discusses related work and Section 6 concludes and discusses future work.

2 Inference techniques and their templates

Our invariant inference framework has two major stages: (1) setting up the model of proper behavior and (2) using it to detect anomalies. The setup stage requires human intervention. Because it is carried out only once, it can rely heavily on human input. The usage stage runs continuously. It permits human intervention, but does not require it. However, periodic usage-stage intervention is often desired in order to better tune the model or adjust to significant changes in the behavior of the data.

In [26] we concentrated on the usage stage of our approach. We now concentrate on the setup stage. Users' expectations are inherently informal. The setup phase aims to formalize these expectations. We do this by introducing a template mechanism.

Templates document the kinds of predicates an invariant inference technique can produce. A user interacts with the template mechanism to express his expectations in the vocabulary the techniques can express. This is an iterative process—it requires looking at the results of invariant inference over several data subsets.

We introduced the idea of templates in [25] through examples. We now define what templates are and show they are useful. Section 2.1 describes the template mechanism. Section 2.2 shows that templates can characterize a wide variety of existing techniques. It describes our technique tool kit—the collection of existing invariant inference techniques that we support and adapt. Section 2.3 discusses data processing that may take place prior to interacting with the template mechanism.

2.1 The template mechanism

We characterize an invariant inference technique by the types of predicates it can produce. Templates capture the form of these predicates. For example, a range template may have the form $\# \leq A \leq \#$, where # is a numeric value and A is an attribute. The dimensionality of a template is the number of attributes in the template. The range template above has dimensionality one. A conjunction of two such range templates has dimensionality two.

The template mechanism uses the invariant inference tool kit, consisting of multiple invariant inference techniques, and the anomaly detector—it employs the techniques in the tool kit to infer invariants over a subset of the data (a moving window of observations) then uses the templates to filter these invariants before giving them to the anomaly detector. The template mechanism has a role during both the set up stage and the usage stage.

Premises of our template mechanism include (1) it is easier for a user to understand expectations about data behavior when presented with examples. It is especially useful to examine examples of anomalous behavior, with the invariants that flagged them as anomalous, and (2) it is easier for a user to choose from a list of inferred invariants than to create this list, so having a machine synthesize the list is helpful.

The template mechanism allows users to select only the parts they like from the invariants a technique infers. This provides much flexibility (e.g., filtering invariants that include anomalous values when the data is very noisy) but may hinder the ability to make certain conclusions (because statistical validity, for example, only applies to the model a technique produces as a whole). In addition, combining parts assumes linearity so the tool kit should provide information regarding which templates can be combined.

In the set up stage the template mechanism interacts with the user to initialize and update templates. The result of this process is a model of proper behavior.

The template mechanism presents the invariants the techniques infer along with observations that falsify an invariant (anomalies) to the user and asks the user to classify the invariants. An inferred invariant is a complete instantiation of a template, but the classification may make the instantiation partial by rendering the instantiation of all the numeric values in one or more dimensions void. For example, a technique may infer the invariant $12 \leq \text{length} \leq 40 \wedge 10 \leq \text{weight} \leq 90$. This is a complete instantiation of the template $\# \leq A_1 \leq \# \wedge \# \leq A_2 \leq \#$. A partial instantiation is one of $12 \leq \text{length} \leq 40 \wedge \# \leq \text{weight} \leq \#$, $\# \leq \text{length} \leq \# \wedge 10 \leq \text{weight} \leq 90$, or $\# \leq \text{length} \leq \# \wedge \# \leq \text{weight} \leq \#$ (default).

The user may classify an invariant as either “accept”, “reject”, or “update”. The template mechanism includes the complete instantiation for accept invariants and the partial instantiation that the user chooses for update and reject invariants. The default partial instantiation is the one with only the template attributes instantiated.

The template mechanism treats the invariant inference techniques as black boxes and uses the instantiated templates to filter the invariants a technique infers. It constructs and updates the model of proper behavior from instantiated templates of accept and update invariants. For each subset of the data, it employs the techniques that produce the selected templates and provides the anomaly detector only with invariants that match templates in the model. It will never present the user or the anomaly detector with invariants that match templates of reject invariants.

When interacting with the template mechanism the user may realize that some values should not occur (e.g., speed should be larger than zero). She can then add these as constraints. The template mechanism treats user constraints as accept invariants. Our approach is a ’single-pass’ one. It is also possible to use the constraints to pre-process the data and then re-run the template mechanism on the cleaner data.

The template mechanism eliminates techniques that are not relevant for this user and data: it will not employ an invariant inference technique if the user rejects all the invariants that are associated with this technique.

In the usage stage the template mechanism reports changes and enables tuning the proper behavior model.

The template mechanism is incorporated in the usage stage and operates continuously. A user may periodically tune the model by invoking the template mechanism to change invariant classification. A user may configure the template mechanism to alert her if the anomaly rate is high. The template mechanism then presents to her accept and update invariants that are consistently different from recently inferred invariants, and recently inferred invariants that are new, and allows her to classify these invariants. However, updating the model during usage is beyond the scope of this paper so we defer it to future work.

2.2 Techniques in the tool kit

The tool kit consists of multiple invariant inference techniques. Users may add techniques to the tool kit. Each technique is likely to be useful only for data with certain characteristics. This provides an initial technique filtering criterion. In addition, different users may find different techniques useful or the output of one technique may be largely redundant with another for the data of interest. We want to use human attention for the most promising techniques. Different techniques make different assumptions and often use different vocabularies. Therefore, it is useful to apply multiple techniques to the same data. However, a large number of techniques is likely to burden the human. The template mechanism supports filtering techniques partially on discriminating ability (effectiveness in anomaly detection) and partially on output comprehensibility. Filtering by these criteria enables the user to select the techniques that promise the best use of human attention.

We briefly describe the techniques we currently have in our invariant inference tool kit and show that templates can characterize their output. We selected these techniques because they expose different aspects of the data and because their output is easy for a human to understand. For each technique, we also enumerate its major strengths and weaknesses. Pre-processing the data, as Section 2.3 discusses, may help to overcome certain technique weaknesses.

This analysis indicates that the most promising techniques for the WIM data are Rectmix and Percentile, so we use these two in our experiment (Sections 3,4). The user interacts with the template mechanism during the setup phase and may classify invariants as either "accept", "reject", or "update". In our examples, we show snapshots of the setup phase with update and reject invariants. In this paper we do not deal with updating invariants during usage, so at the end of the setup phase the user classifies the invariants she likes most as accept invariants. A template for an accept invariant is identical to the invariant (it is fully instantiated). Section 4 lists some of the accept invariants our expert chose.

The Rectmix technique [23] is a clustering algorithm that supports soft membership (a point can probabilistically belong to multiple clusters). The clusters it finds are hyper-rectangles in N-space. Rectmix provides a measure of uncertainty called sigma (an estimate for the standard deviation) for each dimension. Anomalies are points that are not within a rectangle. Though clusters rarely have a hyper-rectangle shape in reality, Rectmix has the significant advantage of producing output that is easy to understand: a hyper-rectangle is simply a conjunction of ranges, one for each attribute

Class	Length ∧	ESAL∧	Axles ∧	Weight
Update	20–42	0–.43	3–3	12–29
Template	#–#	#–#	3–3	#–#
Update	23–44	0–1.2	2–3	26–47
Template	#–#	#–#	2–3	#–#
Reject	13–100	0–.45	2–7	7–40
Template	#–#	#–#	2–7	#–#
Update	23–29	0–6.7	2–4	27–71
Template	#–#	#–#	2–4	#–#

Table 1: Example of Rectmix invariants with user classification and instantiated templates

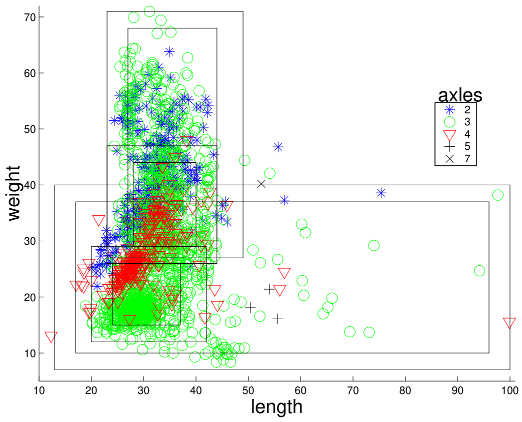

Figure 1: The rectangles of Figure 1 in three dimensions

(see Table 1). Rectmix assumes attributes are similarly scaled. It has two parameters: the number of rectangles and the number of sigmas of uncertainty to allow.

Rectmix always outputs hyper-rectangles, so it has a single template: $\# \leq A_1 \leq \# \wedge ... \wedge \# \leq A_n \leq \#$, where n is the number of attributes (dimensions). Table 1 gives an example of user classification and resulting instantiated templates for rectangles that Rectmix outputs for a subset of the WIM data. These rectangles are four sigmas from the kernel rectangles. ESAL is a derived measure of truck weight (details about the WIM data follow in Section 3.2). Figure 1 depicts three of the four dimension of the Table 1 rectangles: length (x-axis), weight (y-axis), and number of axles (shade and shape). For each kernel rectangle it depicts the bounding rectangle from Table 1.

The Mean and Percentile techniques produce the same kind of invariants: a probable range for the values of each attribute. We prefer Percentile over Mean because it is simpler (makes fewer assumptions).

Mean estimates the mean and standard deviation from the data and expects values to be within a certain number of standard deviations of the mean. Mean is appropriate when the data is roughly normally distributed, but not when extreme values exist or when the distribution is highly skewed

Class	Invariant	Template
Update	40≤speed≤88	#≤speed≤#
Update	17≤length≤39	#≤length≤#
Reject	.06≤ESAL≤.9	#≤ESAL≤#
Update	3≤axles≤3	#≤axles≤#
Update	12≤weight≤49	#≤weight≤#

Table 2: Example of percentile invariants with user classification and instantiated templates

(one of its tails is longer than the other).

The x percentile of a distribution is a value in the distribution such that $x\%$ of the values in the distribution are equal or below it. Percentile calculates the range between the x and 100-x percentiles and allows $y\%$ uncertainty. Percentile makes even fewer assumptions than Mean about the distribution (it only assumes values are somehow centered) and is less sensitive to extreme values.

Mean and Percentile have a single template: #≤A≤#. Table 2 gives an example of user classification and resulting instantiated templates for invariants that Percentile infers over a subset of the WIM data. Percentile (x=25, y=25%) works well for speed, length, axles, and weight, but not for ESAL (ESAL seems to be exponentially distributed).

The Kmeans technique [10] is a clustering algorithm with hard membership—it partitions points into distinct clusters. It assumes attributes are similarly scaled. Anomalies are points that are furthest away from the center of their cluster, according to the Euclidean distance metric Kmeans uses.

Kmeans templates are a set of k cluster centers: $C_1, ..., C_k$, where $C_i = (A_1 = \#\wedge ... \wedge A_n = \#)$ and n is the number of attributes. For the WIM data, when requesting k=2–4 clusters, the centers make sense. However, the furthest observations in each cluster are not necessarily anomalies. This means that the measure of multi-dimensional Euclidean distance is not meaningful for this data. In addition, soft membership is more appropriate for the WIM data than hard membership. Therefore, Kmeans is not a good choice for the WIM data.

The Association Rules technique [1] finds probabilistic rules in an 'if then' form. The rules reflect correlations among attributes but cannot know about cause and effect. They can only give examples with specific values. The advantage is that association rules may detect correlations that may be due to complicated relations. The disadvantage is that they cannot suggest a general relation to explain the correlation. Association rules work on discrete data so numeric data is first divided into bins.

Association rules templates are of the form 'if $E_1 \wedge ... \wedge E_m$ then E_x', where $E_i \in \{\# \leq A_i \leq \# \mid A_i \geq \# \mid A_i \leq \#\}$ and $m < n$, the number of attributes. For the WIM data, association rules work rather well. An example of the rules they produce is 'if length<34.8∧ESAL<0.11∧2≤axles≤3 then weight<19.2'. However, not all the rules contain all of these four attributes, and the cause and effect relation is often absent (as is the case above). In general, Rectmix performs better over the WIM data.

The Daikon technique [12] was developed for the program analysis domain. It dynamically discovers likely program invariants over program execution traces by checking whether pre-defined relations hold. We map Daikon's program points and variables to our observations and attributes, respectively. Daikon assumes the data is clean, but our data contains anomalies. Therefore, we use voting: we run Daikon on multiple subsets of the data and use the invariants that appear in multiple subsets. Daikon is very effective when strong correlations that can be described by its pre-defined relations exist.

We have a template for each of Daikon's pre-defined relations, for example: '$A_i = \#A_j + \#$'. Because the WIM data has mostly statistical correlations the only useful invariants Daikon outputs are of the form 'axles $\in \{\#\}$'. However, Rectmix and Percentile produce similar invariants, so we prefer these techniques for the WIM data.

2.3 Pre-processing

Data preprocessing may be necessary before the template mechanism interacts with the user. This includes setting parameters of inference techniques, performing data transformations, selecting attributes, and clustering.

To determine technique parameters, the template mechanism runs each technique with several values for each parameter and lets the user select the combination that best reflects his expectations. Alternatively, the user may choose to use the default parameter values. For example, the template mechanism runs Rectmix for three to ten rectangles, and gives the anomaly detector hyper-rectangles that are one to four sigmas from the kernel rectangles. The default parameter values are four rectangles and two sigmas. The template mechanism runs Percentile with percentiles x=25 and x=10, and gives the anomaly detector ranges that are either y=25% or y=50% larger than the inter-percentile ranges. The default parameter values are x=25 and y=50%.

Data transformations are usually straight forward and automated. For example, normalizing each numeric valued attribute to have mean one and standard deviation zero is a common transformation that is necessary for techniques that assume similarly scaled attributes (e.g., Rectmix) and data with differently scaled attributes (e.g., the WIM data).

Attribute selection is useful for techniques that produce multi-dimensional templates both because these techniques tend to work better with less dimensions (attributes) and because as the number of dimensions increases it becomes harder for a human to understand and visualize the results. If different classes of data (clusters) exist, they are likely to behave differently. Therefore, the template mechanism runs the techniques on data in each class to enable the user to create a separate model for each class.

We found Principle Component Analysis (PCA) useful for attribute selection and clustering of the WIM data. PCA [17] is a way to reduce the dimensionality of the data thus enabling visualization. PCA generates a new set of variables, called principal components, where each principal component is a linear combination of the original variables. If linear correlations exist, PCA can serve for attribute selection because it indicates which of the attributes are most strongly linearly correlated. Looking at the data helps in finding different classes of the data. To visualize the data, we run PCA on it and plot the observations along the first two principle components. To check for clusters, we plot the same data when coloring observations according to each of the attributes (a color for each value of a discrete valued attribute, a color for each bin of a numeric valued attribute).

For the WIM data, we observed by looking at these plots that either one of vehicle type or axles clusters the data (and the clusters over-lap). We chose vehicle type as the data class. The first principle component indicated a linear correlation among length, ESAL, axles, and weight. Therefore, we selected these attributes as input for techniques that produce multi-dimensional templates (e.g., Rectmix). The other components did not indicate interesting correlations. The second axis is mostly the speed of a vehicle. Therefore, we added the speed attribute for analysis by techniques that produce one-dimensional templates (e.g., Percentile).

3 Experimental setup

We test our template mechanism by having an expert interact with it to set up a model of proper behavior for the WIM data. Section 3.1 presents our hypothesis. Section 3.2 provides details about the WIM data. Section 3.3 summarizes our experimental methodology. Section 4 presents and interprets the results.

3.1 Hypothesis

This experiment tests this hypothesis: the template mechanism helps users formalize their expectations. Further, the template mechanism, along with the invariant inference tool kit and the anomaly detector, effectively direct the human attention necessary in setting up a model of proper behavior and in analyzing the resulting anomalies.

If our hypothesis is correct

1. The resulting model of proper behavior will be useful in detecting semantic anomalies in the WIM data.
2. The user, an expert in this case, will gain insights about the WIM system through interaction with the template mechanism and through analysis of anomalies.

Anomaly detection is not only a first step in increasing the dependability of dynamic data feeds but it also helps users understand the consequences of their model selection and it draws their attention to interesting data behavior.

3.2 Data

A WIM (weigh-in-motion) scale located in the traffic lane of the road measures individual axle weight and spacing for a vehicle that passes over it. Length and speed are derived from the time that passes between noticing axles. The WIM system also records the time when a vehicle passes the scale, the lane it is traveling in, and any error codes generated by the scale. In addition, it classifies the type of a vehicle (using a proprietary classification algorithm) and calculates its ESAL—a measure of axle load that is based on the weight on each axle, the number of axles, the spacing between axles, and the road design. The system also uses software algorithms to filter incorrect values before entering the observations into a database. The scale notices all vehicles but the system only enters into the database data for vehicle types that correspond to trucks.

Several states in the USA are collecting truck WIM data and analyzing it to better understand transportation issues such as road wear. Though there are different WIM scales, the basic data is very similar.

The data we use in our experiments is experimental data the Minnesota Department of Transportation collected in its Mn/ROAD research facilities between January 1998 and December 2000. The data has over three million observations for ten truck types out of fourteen total vehicle types. The amount of observations the system collects varies by vehicle type. For example, it collects two thousand observations during roughly three weeks for each of vehicle types 4 and 6 and during roughly two days for vehicle type 9.

The WIM data feed is a time-stamped sequence of observations. Each observation has attribute values for a single truck: date and time, vehicle type, lane, speed, error code, length, ESAL, axles, and weight. To a first order approximation, we do not expect a temporal drift in the WIM data.

3.3 Methodology

We first look for clusters and select attributes as Section 2.3 describes. As a result, the template mechanism interacts with the user for each class (vehicle type) separately and inputs the selected attributes to techniques in the tool kit (length, ESAL, axles, and weight to all techniques, speed to techniques with one-dimensional templates).

For the purpose of validating our template mechanism, we select three out of the ten vehicle types that the data contains. We select the most common vehicle type (type 9, about two million observations) and two additional types (type 4 and type 6, about one hundred thousand observations each). Naturally, detailed analysis would reveal differences, but on the basis of preliminary analysis, the types seem similar from the stand point of what the techniques do. We first let the user create a model for two of the types (4 and 6) then we let the user create a model for the third type (9) using the techniques and parameters she chose for the first two types. This worked well, supporting our preliminary analysis regarding the similarity of the vehicle types with respect to the tool-kit techniques.

A domain expert sets up the model of proper behavior. We concentrate on the setup stage of our approach, as Section 2.2 describes. Therefore, the template mechanism allows the user to select any invariant classification during setup but only accept or reject invariants on the last setup iteration (it does not update invariants during usage).

We give the model the expert set up to the anomaly detector. The anomaly detector runs over subsets of the data. We sort the data by date and divide it into subsets of two thousand consecutive observations each, in order to simulate the on-line data nature.

To analyze the model, we determine the resulting detection rate and the misclassification rate. The detection rate calculates how many attributes the model flags as anomalies out of the total number of attributes. It is an objective measure because the results of using the model for anomaly detection are binary: normal or anomalous. However, it is important to also analyze the usefulness of the model. The misclassification rate quantifies the usefulness of the model. Because we do not have independent information on correctness this is necessarily subjective. In our experiments, we analyze with our expert anomalies and differences between the inferred and documented models. We concentrate on whether the model is effective in detecting anomalies the user cares about, not on whether it detects all the anomalies.

Rectangle	Length ∧	ESAL∧	Axles ∧	Weight
1	20–42	0–.43	3–3	12–29
2	23–44	0–1.2	2–3	26–47
3	23–29	0–6.7	2–4	27–71

Table 3: Rectmix invariants the expert chose for type 6

To help in inspecting the resulting anomalies, we order the output of the anomaly detector according to the measure a technique is using. We can then output either all the anomalies (and the user can examine as many as he chooses) or only the most significant anomalies. In our experiments, we output all the anomalies.

4 Experimental results and analysis

We detect anomalies over the WIM data using the model the expert set up. Section 4.1 presents the results. We analyze the usefulness of the model for anomaly detection and show insights the expert gains in the process of setting up the model, analyzing the anomalies, and comparing the model to existing documentation. Section 4.2 interprets the results and Section 4.3 discusses major differences between the inferred and documented models. Understanding differences among techniques contributes to better understanding the model and the anomalies the model detects. Section 4.4 presents major differences between Rectmix and Percentile. Section 4.5 summarizes the experimental results.

4.1 Detection Rate

We list the model the expert set up (Rectmix and Percentile invariants) for one of the vehicle types—type 6. We then use the model for anomaly detection and present the resulting detection rate. The plots for type 4 and type 9 vehicles are similar except as indicated in the analysis that follows in Section 4.2.

Table 3 shows the Rectmix invariants the user chose for vehicle type 6. This is the result of turning the “update” invariants of Table 1 into “accept” invariants (Figure 1 depicts the corresponding rectangles). The Percentile invariants the user chose for vehicle type 6 are the result of turning the “update” invariants of Table 2 into “accept” invariants.

Figure 2 depicts a count of anomalous attributes flagged by the Rectmix invariants the expert chose for vehicle type 6. Similarly, Figure 3 depicts a count of anomalous attributes flagged by the Percentile invariants.

Data subsets are time ordered; each has two thousand observations. The y-axis in a plot gives the total number of anomalies in one of the subsets, according to the criterion the plot specifies, e.g., length anomalies. The x-axis is the sequential subset index. Figure 2’s left-most plot summarizes the total number of anomalous attributes, out of eight thousand attributes. The other plots show the break-down of this total by attribute, out of two thousand attributes.

The first column in Figure 3 summarizes the number of anomalies for each attribute. The plots in the second and third columns summarize the anomalies that are due to attribute values that are

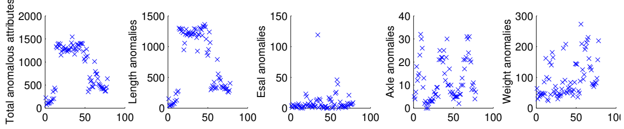

Figure 2: Counts of anomalies detected using Rectmix invariants for vehicle type 6

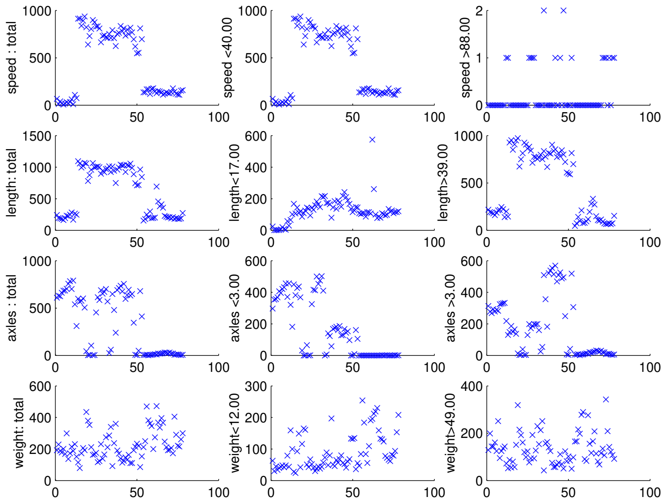

Figure 3: Counts of anomalies detected using Percentile invariants for vehicle type 6

smaller or larger, respectively, than the range bounds. All are out of two thousand attributes.

Table 4 summarizes the average detection rate over the subsets of each vehicle type. It gives the detection rate over all (eight thousand) attributes and a break-down by attribute (out of two thousand attributes).

4.2 Anomalies and insights

Anomalies are both real vehicles that are not in the correct class (they are very different from other vehicles in their assigned class) and vehicles that are physically highly improbable. Our expert cares about both these aspects.

We analyze the usefulness of the model the expert selected for semantic anomaly detection. As part of the analysis, the expert suggests possible root causes for the anomalies. Root causes may be problems or changes in: (1) the documentation, (2) the WIM physical system, including sensors and calibration sensitivities, (3) the WIM software system, including the classification and filtering algorithms, and (4) law enforcement (values are correct but a vehicle should not go this fast, be this long, or weigh this much).

It is interesting to notice that the invariants the user selects from different techniques do not necessarily match each other. In fact, in our experiments, they usually do not match. This is because different techniques discover different behavior—see Section 4.4 for a comparison of Rectmix and Percentile. To turn the resulting model of proper behavior into specifications or to use the model

Rectmix	Vehicle type	Average detection rate (%) Total	Length	ESAL	Speed	Axles	Weight
	4	15.5	42.5	7.7		4.4	7.4
	6	10.9	37.7	0.4		0.6	4.8
	9	2.3	5.0	3.4		0.0	0.9
Percentile	4	8.4	8.1		0.8	10.2	14.6
	6	20.2	30.5		22.2	17.0	11.3
	9	0.8	1.0		0.3	0.0	1.9

Table 4: Average detection rate

to refine existing specifications the user needs to invest additional effort. For example, the user should decide when the model describes the behavior better than the specifications and vise versa, and consolidate differences among invariants.

Looking at the detection rate over a number of subsets is insightful. Patterns and changes become visually obvious.

The detection rate (anomalies) for type 9 vehicles is much lower than for the other types. The data of type 9 vehicles seems much cleaner than for the other types. The number of axles is absolutely clean (no anomalies). The weight is usually normal but in some of the subsets there is a very large number of over-weight vehicles (hundreds out of two thousand). This may be due to a weight sensor problems in the scale or calibration problems on specific dates. Type 9 is by far the most common, so probably the scale and software are calibrated to best recognize this type.

Figure 3 draws our attention to a correlation between low speed and over-length—the plots have a similar shape. This helps us to better understand how the length estimation works. It turns out that the length is estimated from the time that passes between axles, assuming high-way speed.

Looking at the anomalies for axles in Figure 3, it appears there was a change in the WIM system starting subset number 54 (observations starting November 1999). The number of axles is very noisy in earlier observations and very clean in later observations. The same behavior occurs in type 4 vehicles. This may be due to a software update in the classification or filtering algorithms or a re-calibration of the WIM scale. Our expert was surprised to see this behavior. She was also surprised to realize the large number of vehicles with one axle (a truck should have at least two axles).

Both Figure 2 and Figure 3 show that during the period of time in which the axle attribute is clean, the length is also cleaner (fewer anomalies). The same behavior occurs in type 4 vehicles. This may be due to the same change that resulted in a cleaner number of axles. Many of the type 6 length anomalies are due to the maximal length the WIM system can record: 99.9. Our expert was unaware of the large number of exceptionally over-length and under-speed vehicles during the early data collection period, for type 6 vehicles. This may be due to problems in either the scale calibration or the software algorithms for vehicle type 6.

The total detection rate cannot be compared between Rectmix and Percentile because the attributes are not all the same. In addition, these two techniques describe different behavior (see Section 4.4). However, it is interesting to compare the detection rates for the identical attributes

Vehicle type	Average misclassification rate (%)	
	Rectmix	Percentile
4	8.5	3
6	2.3	2.3
9	1	.8

Table 5: Average overall misclassification rate

(length, axle, weight). Understanding these differences contributes to a better understanding of the models.

The axles anomaly detection rate is very different between Rectmix and Percentile because the invariants the expert chose differ (see Section 4.3).

Small differences in the ranges for length and weight result in large differences in the detection rate, indicating that the values for these attributes are closely concentrated. The exact cut-off point between normal and anomalous is, therefore, not clear from the data.

The Rectmix length-anomaly detection rate is about five times the Percentile detection rate, except for vehicle type 6 that has an exceptionally high length-anomaly rate. Except for type 6, the Rectmix length ranges are smaller than the percentile ranges, but the differences are small. This may be due to the homogeneous length within a vehicle type. Type 4 Rectmix length anomalies are numerous compared to the other attributes, indicating the bound may be too tight.

The Percentile weight-anomaly detection rate is about twice the Rectmix detection rate. Rectmix weight ranges are larger than percentile ranges, but the differences are small. This may be due to the weight difference between empty and loaded trucks. Rectmix notices a correlation of weight and ESAL in light vs. heavy trucks. The type 6 upper weight bound is much higher for Rectmix, possibly because it is also considering trucks with more axles.

The overall misclassification is $\frac{FP+FN}{Nor+Ab} = \frac{Ab+FP-TP}{Nor+Ab}$ [27] (lower is better), where True Positives (TP) are correctly detected anomalous data, False Positives (FP) are normal data falsely detected as anomalous, False Negatives (FN) are undetected anomalous data, Normal (Nor=TN+FP) is data that is actually anomaly-free, and Abnormal (Ab=TP+FN) is data with actual anomalies.

Determining the above measures is subjective even though documentation exists. This is because, on the one hand, the documentation is sometimes incomplete and imprecise, and on the other hand, it sometimes describes behavior that neither Rectmix nor Percentile can express.

To determine Ab, FP, and TP, our expert sets constraints that are based on the analysis of both the anomalies flagged by the anomaly detector and the differences between the inferred and documented models (Section 4.3). There is often a tradeoff between true and false positives. Table 5 summarizes the resulting misclassification rate, averaged over the data subsets of each vehicle type. The rates are reasonable for a human to handle. The slightly higher Rectmix rate for type 4 is due to the restrictive lower bound on length. Type 9 is the cleanest, so the techniques do best on it.

The template mechanism aids in achieving low misclassification rates. It enables the user to choose only invariants that describe behavior she cares about and thinks is normal.

The data providers confirmed the expert insights and cause analysis. They were unaware of the behavior that surprised our expert until recently, when they validated analyses that used this data. It turns out that the WIM scale has two different modes for weighing an axle. The various software algorithms made inconsistent assumptions about the weigh mode. As a result, they occasionally assigned values to the wrong attribute. The next algorithms in the chain did not recognize the problem and made calculations based on the incorrect data. Type 9 vehicles are cleaner because one of the many software providers recognized a problem and made an undocumented correction for type 9. In addition, the system is physically calibrated for this type.

4.3 Inferred model vs. documented model

We use the WIM system documentation of vehicle types and attribute bounds as another indicator of what the system might do, and compare it to what the expert finds interesting. Differences between the documented model and the one the user selects may be due to missing/incomplete documentation, implementation/physical properties that differ from the documentation, or different expectations.

The documentation concentrates on upper bounds. The techniques we use infer invariants about lower bounds as well. The expert found the lower bounds useful. For example, under-speed correlates with over-length.

In general, the software algorithms seem to often work strangely for vehicle types other than 9. The classification is very noisy when compared to the vehicle type documentation. For example, the documentation defines the number of axles for a vehicle type, yet, except for type 9, the actual number of axles is often outside the documentation. This led our expert to think about the way the system is calibrated and its effect on vehicle classification. The system seems to be physically tuned for the common type of trucks (type 9) and possibly passenger cars. Some possible causes for anomalies in other types are: (1) inaccurate physical sensing, (2) unintended interaction effects among the various software algorithms (e.g., the filtering algorithms may not properly clean the output of the classification algorithm), and (3) boundary problems in the classification algorithm.

The class documentation often seems imprecise. Our expert chose invariants that are different from the documentation when they described vehicles she thought belonged in the same class. The documentation defines type 4 as traditional buses with at least two axles. The expert allowed only vehicles with 2–4 axles—stricter than the documentation. The documentation defines type 6 vehicles as vehicles with a single frame having three axles. The expert allowed vehicles with 2–4 axles—more permissive than the documentation. The expert allowed vehicles that are heavy within reason. The documentation lists upper bounds for weights but also indicates an uncertainly about these bounds—vehicles may have permits to be over-weight or over-length. However, the data does not contain permit information.

This comparison illuminates subtle expectation differences. The expert emphasizes equally all vehicle types and also data precision. The providers seem to emphasize most vehicle type 9 and avoiding under-estimation. The models reflect these different emphases.

4.4 Rectmix vs. Percentile

Rectmix and Percentile describe different things. Rectmix looks for attribute correlations in high density regions of the data space (common attribute values). Percentile simply looks for where most of the values of an attribute lie.

Rectmix enables a more refined model for high density areas because it takes into account multiple attributes. An example is allowing 2–4 axles for vehicles type 6 as long as the rest of the attributes are reasonable. Percentile simply finds that most of these vehicles have 3 axles.

The Rectmix output often shows a correlation among attributes. For example, the first two rectangles in Table 1 show a correlation between ESAL and weight when the length and number of axles are similar.

However, considering single attributes, as Percentile does, has advantages—it may indicate correlations that are hard to isolate when using multiple attributes. An example is noticing a correlation between under-speed and over-length. This was obscured by incidental correlations in PCA, so we did not give the speed attribute to Rectmix.

4.5 Summary

The experimental results show that: (1) The model is useful for anomaly detection. It enables detecting actual anomalies that the expert cares about: classification problems and unlikely vehicles. In addition, the misclassification rate is reasonable for a human to handle. (2) The expert gained insights about the WIM system through setting up the model and analyzing the resulting anomalies. These insights are related to both the software algorithms and system calibration. These results support our hypothesis (Section 3.1): the template mechanism is effective in formalizing expectations and directing human attention.

5 Related work

Making human intervention explicit. Our template mechanism concentrates on providing a simple and general way for incorporating users' expectations, to create an analyzable model of proper data behavior by everyday users.

Langley [21] recommends that systems supporting computational scientific discovery provide more explicit support for human intervention. He observes that user effort is often required to modulate an algorithm's behavior for the input data, such as parameter setting and directing a heuristic to good candidates. Our template mechanism can be viewed as a way to help users to do this. However, our goal is making expectations explicit thus analyzable, not insights worthy of a scientific publication in their domain.

Lapis [22], a tool for lightweight structured text processing, includes an outlier finder as a way to focus human attention where human judgment is needed. We use anomaly detection in a similar way, though our domain is different.

We use and adapt existing un-supervised learning techniques from the areas of statistics, machine learning, and data mining. Co-training [5] investigates ways to reduce the human effort that labeling data for supervised learning requires. Active learning [7] investigates statistical ways to select the most promising training data for a technique.

The difficulties of defining a model to describe the data may be viewed as an instance of Kolmogorov complexity [8]. The best model is the one generating the data. However, this minimal description length is the Kolmogorov complexity, and Kolmogorov complexity is non-computable.

Detecting/Mitigating problems. Increasing dependability includes prevention and detection/mitigation of problems. Our approach concentrates on detecting semantic problems by the client of a data feed. In general, prevention and detection are complementary as complete prevention is rare.

Detection/mitigation is necessary for all problem types. Solutions exist for specific connectivity [14, 2, 13] and syntax/form [18, 4, 19] problems. But, solutions to semantic problems are scarce and either require domain knowledge [24, 18, 4, 19] or provide a specific technique [15].

Our approach of inferring the characteristics of a data feed from its behavior is similar to work in the areas of program analysis [12, 11, 3], testing [9], and intrusion detection [20, 16]. However, these naturally have a different domain, and often concentrate on a single technique. Daikon [12] dynamically discovers likely program invariants from program executions. We incorporate Daikon in our invariant inference tool kit. “Bugs as deviant behavior” [11] infers beliefs from source code so it is inappropriate for data. “Specifications mining” [3] uses a machine learning approach for discovering specifications of interface protocols. However, it uses techniques specific to code. “Observation-based testing” [9] uses clustering and visualization techniques over program execution profiles to identify unusual executions. We have similar techniques in our tool kit. Intrusion detection mostly looks for patterns in sequences of events, such as network traffic [16]) or user shell commands [20].

Many people have been analyzing WIM data. However, most are concerned with transportation issues, not data quality. Quality analysis was done in [6]. However, it was domain specific, concentrating on the daily sum of ESAL.

6 Conclusions and future work

We proposed a template mechanism to help users formalize their expectations for data behavior. The result is an analyzable model of proper behavior that may serve as a proxy for missing specifications. Our experimental results support the effectiveness of the template mechanism. The model that our expert set up was useful for anomaly detection over the WIM data. Further, the process of model setup and anomaly analysis helped the expert to better understand the system documentation, overall design, and calibration.

The template mechanism enabled the expert to select a good engineering approximation to the desired behavioral model from the invariants the machine learning techniques infer. As a result, she gained insights—surprises about the WIM system that she was previously unaware of. We were able to independently confirm these surprises with the system owners (the data providers).

Many challenges remain in this area. Some of the challenges we plan to work on in the future are: validating the template mechanism on additional domains, evolving invariants, supporting time-correlated data, providing tool support for the template mechanism, and better understanding the human-computer interface issues.

7 Acknowledgments

We thank Dan Pelleg for allowing us to use his Rectmix code, the Auton Lab (www.autonlab.org) for making their dataset processing and analysis software (SPRAT) available to us, the Minnesota Department of Transportation for their WIM data, Beth Latronico and Charles Shelton for comments on this paper.

References

[1] R. Agrawal et al. Mining association rules between sets of items in large databases. In *SIGMOD*, 1993.

[2] Alexa browser enhancement. URL: `http://www.alexa.com`. Accessed April 2001.

[3] G. Ammons et al. Mining specifications. In *POPL*, 2002.

[4] Bauer and Dengler. Trias: Trainable information assistants for cooperative problem solving. In *Agents*, 1999.

[5] A. Blum et al. Combining labeled and unlabeled data with co-training. In *COLT*, 1998.

[6] R. Buchheit et al. Automated procedures for improving the accuracy of sensor-based monitoring data. In *AATT*, 2002.

[7] D. Cohn et al. Active learning with statistical models. *Artificial Intelligence Research*, 1996.

[8] T. Cover et al. *Elements of Info. Theory*. John Wiley, 1991.

[9] W. Dickinson et al. Finding failures by cluster analysis of execution profiles. In *ICSE*, 2001.

[10] R. Duda et al. *Pattern Classification,*. John Wiley & Sons, 2nd edition, 2000.

[11] D. Engler et al. Bugs as deviant behavior: A general approach to inferring errors in systems code. In *SOSP*, 2001.

[12] M. Ernst et al. Dynamically discovering likely program invariants to support program evolution. In *IEEE TSE*, 2000.

[13] Google search engine. URL: `http://www.google.com`. Accessed April 2001.

[14] Go!Zilla download manager. URL: `http://www.gozilla.com`. Accessed April 2001.

[15] Rulequest. GritBot, autonomous data quality auditor. URL: `http://www.rulequest.com/gritbot-info.html`. Accessed January 2002.

[16] S. Hofmeyr et al. Architecture for an artificial immune system. In *Evolutionary Computation*, 2000.

[17] I. Jolliffe. *Principal component analysis*. Springer-Verlag, 1986.

[18] C. Knoblock et al. Accurately and reliably extracting data from the web: A machine learning approach. In *Data Engineering Bulletin*, 1999.

[19] N. Kushmerick. Regression testing for wrapper maintenance. In *AAAI*, 1999.

[20] T. Lane et al. Approaches to online learning and concept drift for user identification in computer security. In *KDD*, 1998.

[21] P. Langley. The computational support of scientific discovery. *Human-Computer Studies*, 2000.

[22] R. Miller et al. Outlier finding: Focusing user attention on possible errors. In *UIST*, 2001.

[23] D. Pelleg et al. Mixtures of rectangles: Interpretable soft clustering. In *ICML*, 2001.

[24] V. Raman et al. Potters wheel: An interactive data cleaning system. In *VLDB*, 2001.

[25] O. Raz et al. Enabling adaptation in systems with under-specified elements. Submitted to WOSS, 2003.

[26] O. Raz et al. Semantic anomaly detection in online data sources. In *ICSE*, 2002.

[27] P. Runeson et al. A classification scheme for studies on fault-prone components. In *PROFES*, 2001.