Helping Everyday Users Find Anomalies in Data Feeds

Orna Raz

May 2004

CMU-CS-04-133
CMU-ISRI-04-119

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Mary Shaw, Carnegie Mellon University, Chair
Philip Koopman, Carnegie Mellon University
Christos Faloutsos, Carnegie Mellon University
Michael Ernst, Massachusetts Institute of Technology

Copyright © 2004 Orna Raz

This research was supported in part by the National Science Foundation under ITR Grant CCR-0086003, in part by the Sloan Software Industry Center at Carnegie Mellon University, in part by the NASA High Dependability Computing Program under cooperative agreement NCC-2-1298, and in part by the EUSES Consortium via the National Science Foundation under ITR Grant CCR-0324770

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Government.

Keywords: Semantic anomaly detection, user expectations, everyday information systems, data feeds

Abstract

Much of the software people use for everyday purposes incorporates elements developed and maintained by someone other than the developer. These elements include not only code and databases but also data feeds. Although everyday information systems are not mission critical, they must be dependable enough for practical use. This is limited by the dependability of the incorporated elements.

It is particularly difficult to evaluate the dependability of data feeds. The specifications of data feeds are often even sketchier than the specifications of software components, the data feeds may be changed by their proprietors, and everyday users of data feeds only have enough knowledge about the application domain to support their own usage. These factors inhibit many dependability enhancement techniques, which require a model of proper behavior for failure detection, preferably in the form of specifications.

The research presented here addresses this problem by providing CUES, Checking User Expectations about Semantics. CUES is a method and a prototype implementation for making user expectations precise and for checking these precise expectations. CUES treats the precise expectations as a proxy for missing specifications. It checks the precise expectations to detect *semantic anomalies*—data feed behavior that does not adhere to these expectations.

Three case studies and a validation study, all with real-world data, provide evidence of the practicality and usefulness of CUES. The case studies and the validation study indicate that a user of CUES gets substantial benefit for a modest investment of time and effort. In addition to automated detection of anomalies, the benefit often includes a better understanding of the user's own expectations, of the data feeds, and of existing and missing documentation.

Contents

Acknowledgments

I thank my advisor, Mary Shaw, for giving me the freedom to do what I am interested in and for all her support and guidance. I was fortunate to have an advisor who is not only an outstanding researcher but also a remarkable human being. I have yet to discover a topic I cannot discuss with Mary.

I was fortunate to have thesis committee members who have provided me with ongoing guidance and encouragement. Philip Koopman has been like a second advisor to me and has helped to shape this research from its conception. Thinking in terms of templates was his suggestion. Christos Faloutsos always provided timely and helpful feedback. He helped me clarify the technical problem of this research and introduced me to the truck weigh-in-motion data and domain expert. Michael Ernst always found the time to discuss my research and provided excellent suggestions. His detailed suggestions helped me to greatly improve the presentation of this research.

The software engineering group has provided me with a forum for discussions. Its members provided valuable feedback, in particular, Shawn Butler, David Garlan, Aaron Greenhouse, James Herbsleb, Timothy Halloran, Elizabeth Latronico, Vahe Poladian, and William Scherlis. I would also like to thank Roy Maxion for thought-provoking discussions.

Laurie Hiyakumoto, Jorjeta Jetcheva, Rosie Jones, Jeanie Komarek, Brigitte Pientka, Bianca Schroeder, Anne Siegel, and Belinda Thom have been good friends, providing support, understanding, and fun.

My parents and brother have always supported and encouraged me. Their love made everything easier. My husband, Dan Pelleg, has shared this journey with me. His love and support made it truly enjoyable. Our son, Tomer, has been a source of joy and has helped me to be the most efficient I have ever been.

Thank you!

Chapter 1

Introduction

People expect software that they use for everyday purposes to be dependable enough for their needs. Because everyday software is not usually mission-critical, it may be cost-effective to detect improper behavior and either notify the user or take remedial actions rather than rely solely on prevention. Fault tolerance approaches to increasing dependability include detection and masking. Both detection and masking require specifications. Detection requires specifications of normal, degraded, and abnormal behavior. Masking requires specifications of outputs.

Unfortunately, specifications of everyday software are often incomplete and imprecise. The situation is exacerbated in modern information systems that incorporate third-party information resources such as Commercial-Off-The-Shelf (COTS) software components, databases, or data feeds.

Data feeds provide a challenging example of everyday information resources. A data feed is a time-ordered sequence of observations on output values. Data feeds may remain under the control of their providers and may have many users relying, in different ways, on behavior the providers may or may not have anticipated. Many challenging, real-world information resources fall under the category of data feeds, including services found on the Internet and software elements that process sensor data or perform monitoring activities. Examples include truck weigh-in-motion data, online quotes for a stock, and online weather conditions. These examples are the data feeds that we use in our case studies and in our validation study.

We propose *CUES*—Checking User Expectations about Semantics—a three-stage, user-centric method and prototype implementation for coping with incomplete specifications of data feeds. The stages of CUES are as follows:

1. *Setup stage*: CUES begins by helping users make their *expectations* about data feed behavior precise. These precise expectations may serve as proxies for missing specifications.

2. *Checking stage*: CUES then uses these proxies to automatically detect *semantic anomalies*—data feed behavior that violates these expectations.

3. *Tuning stage*: CUES also tunes the precise expectations to account for changing data feed behavior or changing user expectations.

Section 1.1 states the claims of this research. Sections 1.2 and 1.3 introduce CUES and the case studies, respectively. Section 1.4 briefly discusses related work. Section 1.5 summarizes the contributions of this research.

1.1 Claims statement

This research contributes to the engineering of information systems that incorporate data feeds by providing CUES.

- CUES is a practical method for helping users state precisely their own expectations about the behavior of data feeds.

- CUES is a practical means of exploiting the users' descriptions of expectations for semi-automated semantic anomaly detection.

By practical we mean cost effective and useful. We measure cost qualitatively both by computation load and by load on the user. We measure benefit/usefulness both qualitatively, by the insights the user gains (the usefulness of the process), and quantitatively, by the detection and misclassification rates (the usefulness of the resulting model). Three case studies and a validation study, all with real-world data, support our claims.

1.2 CUES in a nutshell

CUES consists of a technique tool kit, a template mechanism, and an anomaly detector. The three stages of CUES rely on these mechanisms, as Figure 1.1 shows.

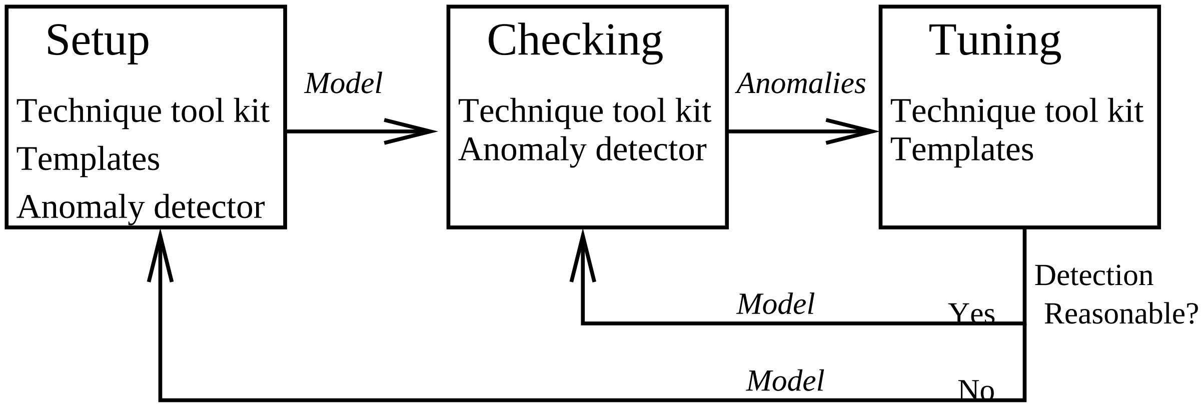

Figure 1.1: CUES: stages and mechanisms. Arrows indicate data flow.

The *technique tool kit* is a collection of existing statistical, machine learning, and program analysis techniques for predicate inference that CUES supports and adapts. The synthesis of these techniques for the purpose of creating a model of proper behavior of data feeds is new. This synthesis is done via a novel *template mechanism*.

CUES applies the tool kit techniques to help discover meaningful information in the data. These techniques precisely characterize various aspects of the data. However, to characterize *relevant* behavior, CUES must elicit the user expectations as well. The template mechanism guides the human attention required in making expectations precise, using templates that document the predicates a particular technique can output.

Each user relies on a data feed in certain ways and expects the behavior of the data feed to support this usage. Therefore, a given user may only care about a subset of the properties the data feed specifies. For example, in our truck weigh-in-motion case study, the providers specified exact ranges for various vehicle attributes such as length and weight. Our user had more relaxed expectations and did not care about exact values as long as vehicles in the same class had similar attribute values. Moreover, a user may care about behavior that is missing from existing specifications or even unnoticed by the providers. For example, in the weigh-in-motion case study, the providers only cared about overestimation, so they specified upper bounds but not lower bounds. Our user cared about both upper and lower bounds, resulting in predicates that immediately detected, for example, a large number of one-axle vehicles, suggesting a problem in the weigh-in-motion system. Users' expectations are informal and imprecise, though they are reasonably accurate. For example, a user may expect trucks reported by an on-road scale to be physically plausible but may not be able to specify a priori all the properties and values that define such plausibility.

The *anomaly detector* uses the precise expectations as a model of proper behavior and reports as anomalies any data feed observations that differ from the expectations. Anomaly detection is straightforward when a model of proper behavior and a criterion for comparing precise expectations to actual behavior exist. However, providing these is challenging when specifications are incomplete. The main contribution of this research is providing automated assistance to users in creating a model of proper behavior for data feeds from the users' informal expectations. In addition, CUES provides principled guidance for adding existing techniques to the tool kit. A user adds a technique by providing details that CUES requires, including a comparison criterion that is appropriate to the technique the user adds.

Figure 1.1 gives a diagram of CUES, showing the data flow. The three stages of CUES rely on the above mechanisms. Chapter 3 provides details about the mechanisms and their synthesis in CUES.

The setup stage initializes a model of expected behavior by eliciting user expectations. It relies on all three mechanisms. The setup stage identifies appropriate techniques for the problem from the technique tool kit, utilizes these techniques to infer predicates that describe data behavior, and creates a model from predicates the user indicates as matching the user's expectations. The setup stage is semi-automated. The main role of the user in this stage is to select predicates that match the user's expectations. The user does this by classifying the predicates that CUES infers. CUES infers predicates automatically by utilizing the tool-kit techniques. The user may also intervene by selecting values other than the default parameter values, over-riding any of CUES' recommendations, and adding predicates to check.

The checking stage checks the model resulting from the setup stage. It relies on the technique tool kit and on the anomaly detector. The checking stage detects semantic anomalies when a new observation violates a predicate in the model. The checking stage is fully automated.

The tuning stage tunes the model to account for changing data behavior or user expectations. It relies on the technique tool kit and the template mechanism. The tuning stage examines the results of the checking stage on the current data subset. If the number of reported anomalies is reasonable (the detection rate is lower than a percentage the user indicates) then the tuning stage re-applies the selected tool kit techniques to the current data subset and gives the re-inferred model to the checking stage. The checking stage will check this model on the next data subset. Otherwise, CUES goes back to the setup stage. The tuning stage is semi-automated. The main role of the user is to inspect the model when CUES notices the model it automatically re-infers may no longer be effective for anomaly detection. The user does this by re-classifying predicates. CUES limits the frequency with which the user has to intervene by combining intervention requests with presenting the detected anomalies to the user.

CUES has the advantages of (1) requiring no knowledge about implementation details of the data feed, including source code or binaries and (2) requiring no user data mining expertise. CUES assumes only that (1) it can observe the data feed over time, in the form the user or automated processing uses it, (2) the data feed usage will tolerate recognition and repair of faults rather than require prevention, and (3) the user has sufficient domain knowledge to select precise characterizations of the data feed behavior.

1.3 Case studies in a nutshell

Three case studies with real-world data feeds and a validation study provide empirical evidence in support of the usefulness of CUES. Each case study is principally focused on the validation of one of the stages of CUES: the truck weigh-in-motion case study, presented in Section 1.3.1, focuses on the setup stage, the stock quotes case study, presented in Section 1.3.2, focuses on the checking stage, and the stock daily low case study, presented in Section 1.3.3, focuses on the tuning stage. The validation study, presented in Section 1.3.4, focuses on the technical feasibility and practicality of the CUES end-to-end approach.

1.3.1 Truck weigh-in-motion (WIM) case study

The *truck weigh-in-motion (WIM) case study* [Raz et al., 2004a, Raz et al., 2004b, Raz et al., 2003] used a real-world data feed from an experimental truck WIM system from the Minnesota Department of Transportation. The data was collected over roughly two years. Truck WIM data is common in the transportation domain, where civil engineers use it for analyses such as road wear. A scale located in a traffic lane of a road weighs every axle that passes over it. It records the weight on the axle, the time of day, the lane the axle was in, and any error codes. Software components analyze this data to map axle data 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 (purportedly) filter out unreasonable values.

In the truck WIM case study a domain expert interacted with the template mechanism to create a model of proper behavior for the WIM data feed from her informal expectations. These informal expectations could be summarized as: (1) vehicles in the same class should be similar and (2) vehicles should be physically plausible. CUES successfully turned these vague expectations into precise predicates, using two tool-kit techniques (Percentile and Rectmix). CUES checked the resulting model to detect anomalies in the WIM data feed. Together with the expert, we compared this model to existing documentation of the data feed. The case study showed that the template mechanism was effective: using CUES, it took just hours to detect problems that had taken the data providers months to detect independently. These problems surprised our user even though she had previously analyzed the same data feed.

1.3.2 Stock quotes case study

The *stock quotes case study* [Raz et al., 2002] used correlated real-world data feeds of current stock information. Each data feed contained data from one of three Internet services for one out of three stock ticker symbols (CSCO, SUNW, TXN). This data was collected Mon–Fri, every ten minutes between 10am and 4pm eastern standard time, for about six weeks. The data feeds were nearly redundant: they provided similar information, but each data feed had a slightly different subset of attributes. The attributes in the superset were: current value, last value, change in value, highest and lowest values in 52 weeks, highest and lowest daily values, value when daily trade began, stock's anticipated fluctuations relative to the market fluctuations (beta), and stock volume.

In the stock quotes case study CUES detected semantic anomalies in the data feeds by using predicates inferred by two of the tool kit techniques (Mean and augmented Daikon). The experimental results demonstrated that it was possible to infer useful predicates over a single data feed of numeric attributes and that this could be done, to a large extent, automatically (in the context of stock market tickers). Not only were the predicates effective in discovering semantic anomalies in a data feed but also they helped us to deduce implicit specifications of the data feed. The case study utilized the existence of nearly redundant data feeds in a voting heuristic that improved the misclassification rate.

1.3.3 Stock daily low case study

The *stock daily low case study* used three of the stock quotes case study data feeds. It used feeds for the same stock ticker symbol from three different online services.

It used only the daily low attribute from each of the feeds, ignoring other attributes that the feeds provided.

In the stock daily low case study an everyday (non-expert) user interacted with the template mechanism to create a model of proper behavior for the daily low attribute of the three feeds. The case study used two of the tool-kit techniques (MUSCLES and augmented Daikon). CUES updated the resulting predicates to account for changing data feeds behavior. The experimental results demonstrated that it was possible to infer useful predicates over multiple loosely-synchronized data feeds with numeric attributes. It was possible to handle time lags if these lags were either fixed or the sampling interval was large compared to the difference in the lag. In addition, it was possible to infer stateful predicates—predicates over multiple observations.

1.3.4 Validation study

The *validation study* used two weather-conditions data feeds. The data was collected hourly during one year. Each data feed reported dry-bulb temperature, wet-bulb temperature, and dew-point temperature at its location. The distance between the two locations was less than 20 miles.

In the validation study two reviewers who had not previously utilized CUES interacted with CUES. Each reviewer set up a model of proper behavior for the weather data feeds. The reviewers set up the model according to a usage scenario and informal expectations that a reviewer's guide defined. The informal expectations were that the data feeds should be roughly similar. The case study used three of the tool-kit techniques (Percentile, Daikon, and MUSCLES). We did not make any changes to CUES in preparation for or during the validation study. Within an hour each reviewer successfully set up a model for the behavior of the data feeds. Analysis of detection and misclassification rates supported the usefulness of the models. Each reviewer then assessed the practicality of CUES for potential users. Both reviewers felt that CUES provides immediate and substantial benefit to non-expert users. A user quickly discovers interesting behavior by looking at the observations CUES flags as anomalous. Setting up the model for anomaly detection is quick. Further, it does not require any expertise beyond having enough understanding about the application domain to support the usage of the data feeds.

1.4 Related work

The motivation and setting of our work draw from work on homeostasis in open resource coalitions [Shaw, 2000]. Shaw recognized the potential benefits of utilizing existing information resources to cost-effectively build information systems. Shaw identified the dependability of such third-party resources as a major obstacle and suggested a self-healing approach. Our work takes a first step towards increasing the dependability of data feeds.

The software engineering research community typically primarily considers software components as information resources. The research described in this dissertation extends that view to include data feeds as first class information resources. The research treats data feeds as part of a composable system. Making data feeds more dependable supports their combination in information systems.

Many computer science sub-disciplines treat data as part of a composable system. Work most closely related to ours includes software architecture styles, such as styles of shared information systems [Shaw and Garlan, 1996], that emphasize the role of data in a system, and continuous queries or stream mining [Seshadri et al., 1996, Chen et al., 2000, Babu and Widom, 2001].

Continuous query systems deal with ways to (efficiently) query continuously arriving data streams. A continuous query engine filters and synthesizes data streams to deliver unbounded results in response to a user query. Adaptation of continuous query engines (e.g., [Avnur and Hellerstein, 2000, Madden et al., 2002]) aims to ensure efficient processing over time, such as minimizing the amount of storage and computation a continuous query requires. Though CUES deals with similar data, it focuses on generating a model of proper behavior for the data streams.

Other related work is discussed in context throughout the presentation of CUES and the case studies.

1.5 Contribution

The main contribution of this research is CUES, a method and a prototype implementation for helping users make their expectations about the behavior of data feeds precise and for checking the precise expectations to detect semantic anomalies in the data feeds.

Detecting semantic anomalies in data feeds is a first step towards increasing the dependability of information systems that incorporate data feeds. Detection raises a flag that can trigger mitigation or repair.

Other contributions are:

- This research demonstrated how CUES may assist the user in better understanding the user's own expectations about the behavior of data feeds, missing documentation, or imprecise existing specifications.
- This research showed how to exploit and adapt existing techniques from the areas of machine learning, statistics, and dynamic program analysis for the purpose of inferring anomaly detection predicates.
- This research provided a template mechanism to elicit user expectations in the form of anomaly detection predicates and to support updating these predicates.
- This research helped to clarify the need for user-centered specifications.
- In addition to the above contributions to computer science, the truck WIM case study produced a contribution in its application domain. That case study made advances towards automated cleaning of data produced by event-based monitoring systems in civil engineering.

1.6 Outline

The remainder of this dissertation is organized as follows.

Chapter 2 presents basic definitions related to data, its properties, and semantic anomalies.

Chapter 3 introduces CUES. CUES has three stages: setup, checking, and tuning. CUES has three mechanisms that implement CUES' stages. These mechanisms are the technique tool kit, the template mechanism, and the anomaly detector.

Chapter 4 first defines and scopes the technical problem that CUES addresses—building a model to detect semantic anomalies in data feeds. It then presents "background" processing that provide support to the CUES stages beyond the CUES mechanisms. It also presents other work most closely related to each of CUES' three stages.

Chapter 5 presents the case studies that utilize CUES over real-world data feeds—the truck WIM case study, the stock quotes case study, and the stock daily low case study. Two of these case studies involved users. The user in the truck WIM case study was a domain expert. The user in the stock daily low case study was an everyday (non-expert) user.

Chapter 6 provides evidence in support of our thesis claims. In addition to the three case studies, a validation study and a cost-benefit analysis support the thesis claims.

Chapter 7 concludes with a summary of the contributions of this research and a discussion of future work.

Chapter 2

Basic definitions

We define the basic terms that we use in this research. Section 2.1 provides basic definitions related to the data CUES handles. Section 2.2 defines the problems CUES detects: semantic anomalies.

2.1 Data and its properties

Data feeds are third party information resources, such as Internet services and sensor data. A *data feed* is a time-ordered vector of observations. An *observation* at time t is a tuple that contains a value for each attribute: $\mathrm{OBS}(t) = \langle a_1, ..., a_n \rangle$, where a_i is a numeric valued attribute (in this research we concentrate on numeric valued attributes).

Anomaly detection requires a model of proper behavior of the data feed, preferably in the form of specifications. *Specifications* define what a user of a data feed can rely on. However, data feeds often have incomplete and imprecise specifications.

CUES infers a model of proper behavior that may serve as a proxy for incomplete or imprecise specifications. A *model* is a set of predicates. Given a model and a tolerance (i.e., threshold or confidence), an *anomaly* is an observation that has attribute values that are different from the model by more than the tolerance.

A single data feed or multiple data feeds may be used for inferring predicates. For multiple data feeds CUES requires synchronization in the form of matching time stamps. CUES' implementation does not have a means for synchronization. Section 4.2.1 discusses how CUES handles multiple data feeds. Basically, CUES creates a single observation from multiple data feeds (it concatenates observations with identical time stamps) and possibly accommodates a time lag between the feeds.

Predicates may be *stateless* or *stateful*. Stateless predicates require a single observation for evaluation, whereas stateful predicates require a sequence of observations for evaluation.

2.2 Semantic anomalies

A *semantic anomaly* is defined with respect to a model. If a model exists, a semantic anomaly is an observation that has values that deviate from the model by more than a tolerance. This research finds a model that matches a subset of the user's expectations and captures the model as a set of predicates. A semantic anomaly is, then, an observation that has values that violate any of the predicates of the model by more than the tolerance.

Examples of problems that may result in semantic anomalies in data feeds are reporting data provided by an erroneous sensor, relying on erroneous data provided by a different data feeds or entered by a human, changing database schema, and changing the method of reporting.

In the absence of complete specifications, the same anomalous behavior sometimes indicates a failure and at other times does not. A *software failure* [Lyu, 1996] is either an outcome that violates the specifications of that software or an unexpected software behavior observed by the user. A *software fault* is the identified or hypothesized cause of the software failure. For example, an extreme value change in the value of a stock may be caused by a fault in the data. However, stocks sometimes exhibit extreme value changes. These cases cannot be distinguished without knowledge of the market, although both are anomalous.

Ideally, specifications would describe correctly and precisely all possible data feed behavior. In reality, however, this rarely happens for data feeds. Often, the model CUES checks to detect semantic anomalies, which describes precisely a subset of the user's expectations, complements existing documentation or suggests missing documentation. It enables the discovery of previously unknown or unnoticed behavior. Our case studies provide examples. The truck WIM case study suggested behavior the expert was unaware of—the expert gained insights about the system's behavior. See Section 5.1.8 for details. That case study also suggested places where the existing documentation was too restrictive or too permissive. See Section 5.1.7 for details. The stock quotes case study helped us to suggest missing documentation. See Section

5.2.4 for details.

Our definition of semantic anomalies can be viewed as a semantic extension of the usual notion of availability—a facet of dependability. The *dependability* of a computing system [Avizienis et al., 2001] is "the ability to deliver service that can justifiably be trusted. The service delivered by a system is its behavior as it is perceived by its user(s)". *Availability* [Avizienis et al., 2001] is "readiness for correct service, a measure of delivery of correct service with respect to the alternation of correct and incorrect service". Instead of using the traditional "fail-silent" (crash failures) fault model of availability, we explore using a "semantic" fault model.

Chapter 3

CUES: mechanisms

CUES—Checking User Expectations about Semantics—is a three-stage user-centered method and prototype implementation for checking data feeds that have incomplete (i.e., missing or imprecise) specifications. Figure 3.1 depicts the stages of CUES—setup, checking, and tuning—and the mechanisms these stages rely on—technique tool kit, templates, and anomaly detector.

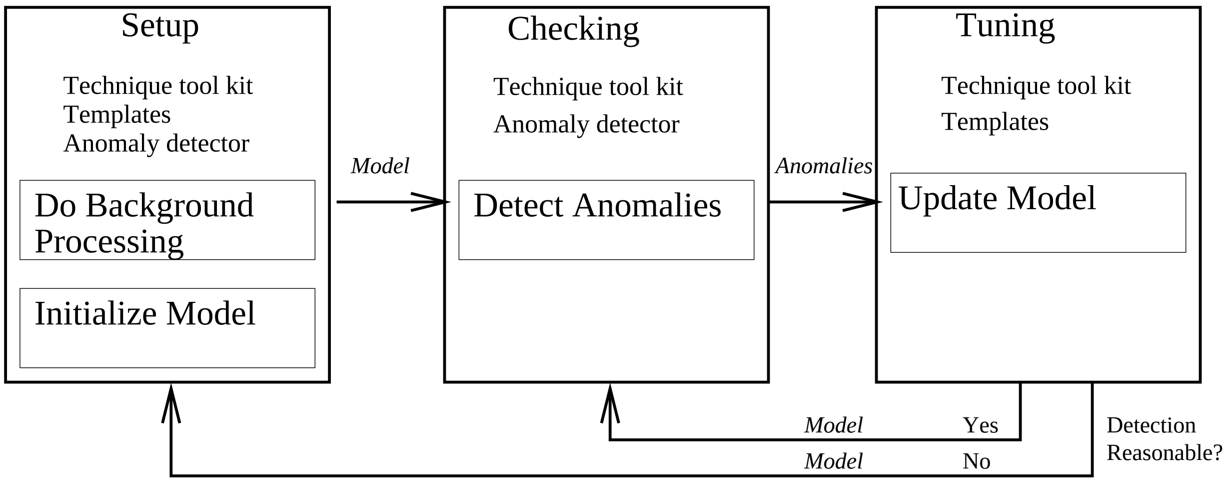

Figure 3.1: CUES: detailed stage diagram. Arrows with italics show the data flow

Section 3.1 introduces CUES. It provides an overview of the mechanisms that CUES is based on by presenting the CUES stages—the process governing the CUES mechanisms. Section 3.2 expands the introduction of CUES through a simple example, concentrating on how a user may utilize CUES. A detailed presentation of the mechanisms follows: Section 3.3 presents the technique tool kit, Section 3.4 presents the template mechanism, Section 3.5 provides details about the templates of the tool-kit techniques, and Section 3.6 presents the anomaly detector.

Chapter 4 that follows presents "background" processing that CUES performs but is not part of its three underlying mechanisms. Chapter 4 also presents other work related to CUES.

3.1 CUES stages: a process governing the mechanisms

CUES' stages can be viewed as a process that provides appropriate data to the CUES mechanisms and controls their execution. The goal of this process is to generate a model of proper behavior for data feeds and to check this model in order to detect semantic anomalies—data feed observations that differ from the model. CUES performs model generation and checking for data feeds that the user indicates. Sections 3.1.1, 3.1.2, and 3.1.3 introduce the CUES mechanisms and their role in model generation, checking, and tuning, respectively. Section 3.2 that follows complements this introduction by providing an example concentrating on the view point of a user of CUES.

3.1.1 The setup stage

The setup stage (the left box in Figure 3.1) provides semi-automated support to the user in creating the proper-behavior model. CUES automatically generates predicates over a sample of the data. These predicates describe various aspects of the data behavior. CUES lets the user choose the predicates that describe behavior that is relevant to the user's expectations. The resulting model is the set of predicates that the user selected.

The setup stage operates from the assumption that choosing is easier than generating. Templates provide the major mechanism that supports working under this assumption. The *template mechanism* enables the user to choose a model—a set of predicates—from a list of predicates CUES automatically generates. The user sorts predicates into three categories: (1) "accept": the user expects this predicate to hold over the data, (2) "update": the user expects this predicate to hold over the data but also expects some of its numeric values to change over time, and (3) "reject": the user does not expect this predicate to hold over the data (the behavior the predicate characterizes may be irrelevant to the user's expectations or the predicate may describe noise in the data).

The *technique tool kit* enables automated predicate generation. The technique tool kit supports and adapts multiple inference techniques. These techniques are existing machine learning, statistical, and program analysis techniques. In essence, templates document the predicates of the inference techniques. A *template* is a generalized predicate: it is identical to the predicate with the exception of a # sign indicating a numeric value where the predicate has a specific numeric value. Often, only a subset of the tool-kit techniques is appropriate to the problem. Further, only a subset of the predicates inferred by these techniques is relevant to the user expectations. Templates support selecting only the relevant predicates from the output of the appropriate techniques. The template mechanism applies the user's predicate classification to instantiate templates and to filter the output of the inference techniques. It includes only "accept" and "update" predicates in the resulting model.

In addition to automated predicate generation, CUES supports incorporating some forms of existing specifications. CUES supports specifications that allow discrete localized checking and adds these specifications as "accept" predicates. Section 3.4.3 provides details.

The setup stage also operates under the assumption that it is easier for users to understand their expectations about data behavior when presented with examples of data behavior. Specifically, examples of anomalous behavior together with the predicates flagging them as anomalous are especially helpful. The *anomaly detector* provides the major mechanism that supports working under this assumption. Anomaly detection is not only a first step in increasing the dependability of data feeds but it also is a means of providing immediate feedback to the user about the consequences of the model the user selected. In our research, we also employ anomaly detection in assessing the usefulness of the model the user has selected.

It is common knowledge that insights often arrive from looking at a problem from different view points. Examples of points of view are examining existing documentation of a data feed and inspecting predicates that a technique infers dynamically over the data feed. The structure of CUES supports this known benefit: making expectations precise often leads the user to take a different view point and gain additional understanding. For example, in the truck WIM case study (Section 5.1), the user gained insights about the underlying WIM system, its calibration, and the data processing. Section 5.1.8 summarizes the user's insights in that case study.

3.1.2 The checking stage

The checking stage (the middle box in Figure 3.1) automatically checks the model to detect semantic anomalies in the data feeds. The checking stage gets the model from either the setup stage or the tuning stage. Both the setup and the tuning stages utilize the template mechanism to only give the anomaly detector predicates that the user has selected: "accept" and "update" predicates. The anomaly detector evaluates these predicates on current observations—observations in the current moving window of observations (CUES infers the predicates over the previous moving window of observations). The anomaly detector warns the user if the results of evaluating the predicates are different from the values of actual observations. The technique tool kit defines a proper comparison measure for the predicates each technique can infer. The anomaly detector aggregates warnings and only reports them to the user once for every moving window of observations. These warnings do not require user confirmation.

The anomaly detector checks predicates that the tool-kit techniques dynamically infer, including predicates the user expects to change over time. CUES can treat these predicates as a proxy for missing specifications because it tunes these predicates to adjust then to changing data behavior and changing user expectations.

3.1.3 The tuning stage

The tuning stage (the right box in Figure 3.1) repeatedly tunes the model to account for changing behavior. It does this mostly automatically. The tuning stage employs the tool-kit techniques that the user has selected in the setup stage. It runs these techniques to re-infer predicates over a moving window of observations. The tuning stage employs the template mechanism to update those predicates in the model that the user expects to change over time: the "update" predicates. The tuning stage has a sanity-check heuristic (Section 4.4.2) for deciding when to request the user to intervene. This heuristic examines the number of anomalies the checking stage detects. If this number is larger than a pre-defined threshold and using a previous model does not result in a reasonable detection rate, the tuning stage takes the user back to the setup stage and asks the user to re-examine the model and to possibly change the user's classification. The tuning stage tries to reduce the user distraction by combining requests to intervene with a task that may already require the user's attention: examining the anomalies the checking stage reports over the moving window of observations.

3.2 Usage example

Following is an example showing how a user may interact with CUES. This example draws from the stock quotes case study (Section 5.2).

Imagine a casual user of stock quotes. For example, the user may invest small amounts of money in the stock market or may seek employment in companies that the user believes show promise in the stock market. The user may carry out such decisions based on automated processing. The user may utilize CUES to be automatically notified when the data this processing is based on seems anomalous.

The user indicates the data feeds to CUES. For example, the user may be downloading stock quotes for a stock ticker symbol from a freely available online service every 10 minutes during business hours of weekdays. Table 3.1 gives an example such a stock quote data feed. The data feed provides information about a specific stock (CSCO) from a specific online service [S3, 2000]. A data feed is a time ordered vector of observations. Each observation (a row in Table 3.1) contains a time stamp and a value for each attribute. This data feed has three attributes: the current value of the stock `cur`, the daily low value of the stock `dlow`, and the lowest value of the stock during the last 52 weeks `w52low`.

Date and time	`cur`	`dlow`	`w52low`
10/11/2000 15:30	52.13	48.22	32.53
10/11/2000 15:40	52.06	48.22	32.53
...	...	...	...
10/17/2000 14:20	54.00	54.00	32.56
10/17/2000 14:30	53.50	53.44	32.56

Table 3.1: Example of a stock quote data feed

During the setup stage the user creates a model of proper behavior for the data feed. This model is based on past behavior of the data feed. The checking stage will check future behavior. The main task of the user is to select predicates that match her expectations from the list of predicates that CUES generates. Her expectations may be vague and informal, such as expecting reasonable values for the current value of the stock.

To make expectations precise, the template mechanism interacts with the user. Prior to interacting with the template mechanism, the user may select data preprocessing actions, improve automated attribute selection, and run clustering algorithms or indicate manually classes in the data (details follow in Chapter 4). In the example, none of these is necessary. When interacting with the template mechanism, the user first provides basic information about the data scale. If the user supplies multiple data feeds to CUES, the user also selects the type of correlation the user expects among the feeds[1] (e.g, a linear relation). This enables the template mechanism to suggest an initial set of tool-kit techniques that may be appropriate to the problem (details in Section 3.4.4). The user may further select techniques. In the example, the user indicates that all attributes have an interval scale. The template mechanism suggests two of the tool-kit techniques.

The template mechanism repeatedly runs the selected techniques to infer predicates, presents to the user the anomalies that result from checking these predicates, and asks the user to classify predicates. This classification has three categories: "accept", "update", and "reject". The template mechanism uses the classification to initialize templates and automatically filter the output of the tool-kit techniques (details in Section 3.4). In the example, the selected techniques may infer the predicates that Table 3.2 lists. The user is now able to make her expectations precise by selecting from these predicates through classification. Table 3.2 also lists an example of possible user classification. The template corresponding to predicate number (6), for example, would be $\# \leq \texttt{dlow} \leq \#$, where # indicates a numeric value.

First technique		Second technique	
Predicate	Classification	Predicate	Classification
(1) $\texttt{cur} \geq \texttt{dlow}$	accept	(5) $44.81 \leq \texttt{cur} \leq 65.27$	reject
(2) $\texttt{dlow} \geq \texttt{w52low}$	accept	(6) $42.05 \leq \texttt{dlow} \leq 64.50$	update
(3) $\texttt{dlow} \geq 48.22$	reject	(7) $32.50 \leq \texttt{w52low} \leq 32.60$	update
(4) $\texttt{w52low} == 32.53$	reject		

Table 3.2: Example of predicate the two selected techniques infer over the data feed of Table 3.1 and of user classification for these predicates

Once a model exists, the checking stage of CUES repeatedly and automatically checks the model's "accept" and "update" predicates over observations in the current moving window of observations. It reports to the user as anomalous observations that violate any of these predicates. The checking stage outputs one report summarizing anomalies over all the observations in the moving window. The user examines the anomalies and may decide to change the classification as a result. Table 3.3 shows

[1]The user could also get arbitrary correlations by running all the CUES techniques. However, the user will them have to tune parameters for all the techniques and also inspect a potentially large number of predicates

an example of detecting an anomaly. The anomaly detector flags the observation at date 10/25/2000 time 11:10 as anomalous as a result of checking predicates number (1), (6) and (7) of Table 3.2.

time	`cur`	`dlow`	`w52low`
10/11/2000 15:30	52.13	48.22	32.53
10/11/2000 15:40	52.06	48.22	32.53
...	...	...	...
10/17/2000 14:20	54.00	54.00	32.56
10/17/2000 14:30	53.50	53.44	32.56
...	...	...	
10/25/2000 11:10	*52.38*	*106.00*	*106.00*
...	...	...	...

Table 3.3: Example of anomaly detection. Italics mark the anomalous observation. Checking the Table 3.2 predicates flags the `dlow` *and* `w52low` *values of the 10/25/2000 11:10 observation as anomalous*

The tuning stage of CUES repeatedly updates the "update" predicates in the model by automatically re-inferring predicates over each new window of observations. It provides the anomaly detector (in the checking stage) with the current model. The checking stage will check this model on the next moving window. If the checking stage flags too many observations as anomalous (e.g., the anomaly detector flags as anomalous more than 30% of the observations in the window) the tuning stage requests the user to intervene by taking the user back to the classification step of the setup stage. In the example, there may be a period of time in which there is not much activity in the stock, therefore, `dlow` is constant. As a result, the first technique may infer the predicate `dlow` == 48.22 and the user may classify this predicate as "update". As a result, the template mechanism will generate the template `dlow` == # for this predicate. This period may be followed by a period in which there are rapid changes in the value of the stock. Updating the constant to any numeric value would result in flagging many valid observations as anomalous. During the predicate re-inference, the first technique will no longer infer the above predicate (and therefore, the tuning stage will keep the existing numeric value; the tuning stage does not automatically remove or add predicates to the model). This predicate will cause the checking stage to detect too many anomalies. When presenting the anomalies to the user, the tuning stage will warn the user that the model is probably dated and request the user to examine the model. The user may decide to change the classification of this predicate to "reject". The user may also decide to classify newly inferred predicates that better describes the current behavior of the data feed (such as predicate number (6) in Table 3.2) as "update".

3.3 Technique tool kit

The technique tool kit supports and adapts multiple predicate inference techniques. The tool kit incorporates existing machine learning, statistical, and program analysis techniques. The tool kit provides default values for the parameters of each technique. It also recommends parameter values for the user to experiment with if the user chooses not to use the default values. It does so in order to ease the user's task of parameter setting. In the description of the techniques that follows we specify the default values and the values that CUES recommends. However, the user may choose any values. The tool kit also provides a measure for comparing the predicates the technique infers to actual data.

The technique tool kit currently includes the following techniques: Rectmix—a clustering technique, Mean and Percentile—simple statistical techniques, Magnum Opus—an association rules technique, Daikon—a dynamic invariant detection technique, and MUSCLES—a linear regression technique. We selected these techniques because they expose different aspects of the data and because their output is easy for a human to understand. Table 3.4 summarizes the assumptions the different techniques make and the resulting type of predicates they can infer. The assumptions include the minimal data measurement scale ([Fenton and Pfleeger, 1997]; reviewed in Figure 3.2) each technique is appropriate for. Section 3.4.4 discusses how the template mechanism utilizes the information of Table 3.4 to select from the tool kit a subset of techniques appropriate to the problem. Sections 3.3.1–3.3.5 describe each of the tool-kit techniques. Sections 3.5.1–3.5.5 present the templates corresponding to each technique—a precise documentation of the type of predicates each technique can infer. Sections 3.5.1–3.5.5 also provide additional examples of predicates: predicates each technique infers over the truck WIM data.

A person who has a good understanding of a technique may add this technique the tool kit. Section 3.3.6 provides guidelines for adding a technique to the tool kit.

3.3.1 Clustering—the Rectmix technique

The Rectmix technique [Pelleg and Moore, 2001] is a clustering algorithm that supports soft membership (a point can probabilistically belong to multiple clusters). The

Technique	Assumptions about the data	Types of predicates
Rectmix	Interval scale Correlation among multiple attributes exists in the form of different groups (classes) of values, and the values are concentrated in ranges	Hyper rectangles
Mean	Interval scale Attribute values are normally distributed	Range
Percentile	Interval scale Attribute values are somewhat centered	Range
Association rules	Nominal scale Correlation among multiple attributes exists in the form of associations	Association rules (if A then B)
Daikon	Nominal/interval scale	Equality, inequality, linear, one-of
MUSCLES	Interval scale Correlation among multiple attributes exists in the form of a linear relation, possibly over time	Multi-variate linear-regression

Table 3.4: Tool-kit techniques: assumptions and resulting predicate types

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 cluster.

Rectmix assumes clusters have a hyper-rectangle shape. It is appropriate for interval scale data.

Rectmix has two parameters: the number of rectangles (clusters) and the number of sigmas of uncertainty to allow (the same for all dimensions). The technique tool kit runs Rectmix for three to ten rectangles. The default is four rectangles. When determining whether a value fits a predicate[2] the tool kit allows one to fours sigmas difference from the rectangle. The default is two sigmas.

An example of predicates (with three clusters/rectangles) that Rectmix could infer over the stock quotes data feed example (Table 3.1) is:

- $52 \leq \texttt{cur} \leq 53 \wedge 48 \leq \texttt{dlow} \leq 48.5 \wedge 32 \leq \texttt{w52low} \leq 32.6$
- $53 \leq \texttt{cur} \leq 54.5 \wedge 53 \leq \texttt{dlow} \leq 54.5 \wedge 32 \leq \texttt{w52low} \leq 32.6$

[2]CUES implements this by adjusting the predicates during inference. The effect is identical to soft comparisons during checking.

- *Nominal measurement* assigns items to categories. It is not possible to quantify or rank order these categories. An example is the lane attribute in the truck WIM case study. Other examples are gender and race. Categorical variables are measured on a nominal scale.

- *Ordinal measurement* assigns higher numbers to items representing higher values. However, the intervals between the numbers cannot quantify the difference between the numbers. An example is a three point rating scale for customer satisfaction.

- *Interval measurement* allows not only to rank order items but also to quantify and compare the size of differences between the items. However, interval scales do not have a "true" zero point and therefore, it is not possible to take the ratio of two interval scale items. An example is the Fahrenheit scale for temperature.

- *Ratio measurement* is similar to interval measurement with the addition of the existence of an identifiable absolute zero point. An example is the Kelvin scale of temperature. Most statistical data analysis procedures do not distinguish between the interval and ratio properties of the measurement scales. CUES follows this common approach.

Figure 3.2: Measurement scales

- $52 \leq \texttt{cur} \leq 55 \wedge 48 \leq \texttt{dlow} \leq 54.5 \wedge 32 \leq \texttt{w52low} \leq 32.6$

3.3.2 The Mean and Percentile techniques

Mean and Percentile estimate an interval for the values of an attribute (Rectmix also estimates intervals but in multiple dimensions, therefore it finds a correlation among attributes). Mean estimates the mean and standard deviation of an attribute distribution from the data and expects values to be within a certain number of standard deviations of the mean.

Mean assumes the data is normally distributed.[3] It is inappropriate when extreme values exist or when the distribution is highly skewed (one of its tails is longer than the other). Mean is appropriate for interval scale data.

[3]The CUES implementation could include a test for normality, to decide between using Mean and Percentile.

Mean has one parameter: the number of standard deviations to allow. The technique tool kit can run Mean with any number of standard deviations but recommends to the user two to ten standard deviations. The default is five standard deviations.

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 only assumes that the distribution values are somewhat centered and is insensitive to extreme values. Percentile is appropriate for interval scale data.

Percentile has two parameters: x and y. The technique tool kit runs Percentile with percentiles x=25 or x=10. The default is x=25. When determining whether a value fits a predicate the tool kit allows ranges that are either y=25% or y=50% larger than the inter-percentile ranges. The default is y=50%. These value recommendations are an arbitrary heuristic that has not been theoretically justified. As is true for all of the CUES recommendations, the user may choose to ignore these recommendations by choosing different values.

An example of predicates that Mean or Percentile could infer over the stock quotes data feed example (Table 3.1) is:

- $44.81 \leq \texttt{cur} \leq 65.27$
- $42.05 \leq \texttt{dlow} \leq 64.50$
- $32.50 \leq \texttt{w52low} \leq 32.60$

3.3.3 Association rules—the Magnum Opus technique

The Association Rules technique [Agrawal et al., 1993] finds probabilistic rules in an 'if then' form. An association identifies dependent attribute values: it looks for a combination of attribute values that occur together with frequency greater than could be be expected if the values were independent of one another.

Association rules are appropriate for nominal scale data (categorical valued attributes) so numeric data should first be divided into bins.

The tool-kit incorporates a specific implementation [Magnum Opus, 2002] of Association rules. An example of a predicate that Association rules could infer over the stock quotes data feed example (Table 3.1) is[4]:

[4]Provided the data has been divided into bins first; Section 4.2.1 provides details about data preprocessing.

- if `cur` $> 52 \wedge$ `dlow` $== 48.22$ then `w52low` $== 32.53$

3.3.4 Dynamic invariant detection—the Daikon technique

The Daikon technique [Ernst et al., 2001] dynamically discovers likely program invariants over program execution traces by checking whether pre-defined relations hold. Daikon, being intended for program analysis, is capable of generating a potentially rich syntax. A list of all the invariants that Daikon detects appears under the "Invariant List" Section of the Daikon manual [Daikon Invariants, 2004]. Some of the most interesting predicates for CUES' purposes are:

1. an equality between 2 attributes or between an attribute and a numeric value; appropriate for interval scale data
2. an inequality ($\neq, \leq, <, >, \geq$) between 2 attributes or between an attribute and a numeric value; appropriate for interval scale data
3. a linear relation among 2 or 3 attributes; appropriate for interval scale data
4. a 'one of' relation between an attribute and a set of values; appropriate for nominal scale data

The tool kit maps Daikon's program points and variables to CUES' observations and attributes, respectively. Daikon assumes the data is clean, but data feeds are noisy. Therefore, the tool kit applies voting over Daikon's output: the tool kit runs Daikon on multiple subsets of the data and selects only the invariants that appear in multiple subsets. As a result, augmented Daikon has one parameter: the number of subsets to take into account in the voting. The tool kit runs Daikon over three to five subsets. The default is three.

An example of predicates that Daikon could infer over the stock quotes data feed example (Table 3.1) is:

- `cur` $\geq$ `dlow`
- `dlow` $\geq$ `w52low`
- `w52low` $== 32.53$

3.3.5 Linear regression—the MUSCLES technique

The MUSCLES technique [Yi et al., 2000] is a multi-variate linear regression technique for time sequences. MUSCLES estimates the current value of one of the sequences as a linear combination of past values of the estimated sequence and past and current values of the other sequences.

MUSCLES assumes the data sequences are co-evolving over time and this correlation is linear. It is appropriate for interval scale data.

MUSCLES has several parameters:

1. window size
2. sequence to predict (estimate)
3. number of observations to predict
4. number of observations for training
5. forgetting factor between 0 and 1

The important parameters for CUES' purposes are the size of the window to use for past observations, the sequence to predict, and the number of observations to use for training. The technique tool kit uses the MUSCLES defaults for the other parameters. CUES recommends a window size. Section 4.2.2 provides details about selecting a size for the technique window. The CUES manual recommends to the user estimating the data feed that has the most lag compared to the other data feeds (see the stocks daily low case study, Section 5.3.2, for an example). Finding a good number of observations to use for training is the same problem as finding a good size for the CUES moving window of observations. Section 4.2.2 discusses selecting a moving window size—a problem that is relevant to all the tool-kit techniques (because CUES always infers and checks predicates over a moving window of observations). When determining whether a value fits a predicate the tool kit allows a difference of two to ten standard deviations (of the estimated data feed). The default is a difference of five standard deviations.

An example of a predicate that MUSCLES could infer over the stock quotes data feed example (Table 3.1) is[5] (window size 1, estimating `w52low`):

- $\texttt{w52low}(t) == \texttt{w52low}(t-1)$

[5]The MUSCLES version the tool kit incorporates does not select attributes.

3.3.6 Adding a technique to the tool kit

A user who is capable of providing the following information may add a technique to the tool kit. The user should add the following information to the code that implements CUES and add appropriate documentation to the CUES' manual.

1. Provide the code implementing the new technique.

2. Specify technique augmentation, if needed. An example is augmenting Daikon with voting to support noisy data (Section 3.3.4).

3. Specify technique parameters, including default values and values the user may select when interacting with the template mechanism. Section 4.2.2 discusses parameter setting.

4. Provide templates for the predicates the technique infers. Section 3.5 lists templates for the current tool-kit techniques.

5. Provide an appropriate question for initial technique recommendation by the template mechanism; Section 3.4.4 lists the questions for the current techniques in the toolkit.

6. Provide a procedure for computing a tolerance for comparing numeric values. Comparing numeric values is necessary when comparing templates to predicates the tool-kit techniques re-infer and when checking for anomalies. For example, for the Rectmix technique (Section 3.3.1), the tolerance is the selected number of the sigma of each dimension.

In addition to documenting the above additions to the implementation of CUES, the user should specify the following information in CUES' user manual:

1. Specify the assumptions the technique makes about the data. Table 3.4 lists assumptions the current tool-kit techniques make.

2. Specify recommended or needed data preprocessing. Section 4.2.1 discusses preprocessing that CUES supports. An example is dividing numeric valued attributes into bins for Association rules (Section 3.3.3).

3.4 The template mechanism

We characterize a predicate inference technique by the types of predicates it can infer. Templates capture the form of these predicates. For example, an inference technique may find a probable range for the values of a given attribute, e.g., the `dlow` attribute. The corresponding template would be $\# \leq$`dlow`$\leq \#$, where # is a numeric value. Section 3.5 states the templates of the techniques currently in the tool kit.

Figure 3.3 summarizes the major steps of the template mechanism. The template mechanism enables the user to choose from the predicates the tool-kit techniques infer by asking the user to classify these predicates. It then applies this classification to create an initial model of proper behavior and to automatically filter the future output of the techniques.

1. Recommend appropriate techniques for the problem from the tool kit (details follow in Section 3.4.4).
2. Ask the user to select techniques.
3. Run the selected subset of techniques from the tool kit to infer predicates over subsets of the data.
4. Ask the user to classify each predicate as either "accept", "update", or "reject".
5. Instantiate templates based on the classification.
6. Apply the instantiated templates to filter the output (i.e., predicates) of the tool kit techniques.
7. Give the filtered output to the anomaly detector and present to the user the resulting anomalies along with the predicates that triggered these anomalies. Allow the user to change the predicate classification.
8. Goto 3 or terminate when the user is happy with the classification.

Figure 3.3: Outline of the template mechanism procedure

The main goal of the template mechanism is to support:

1. creating an initial model of proper behavior. Section 3.4.1 discusses the template mechanism and its role during CUES' setup stage.

2. automated updating of the model of proper behavior. Section 3.4.2 discusses the role of the template mechanism during CUES' tuning stage.

Section 3.4.3 discusses how to incorporate existing knowledge with templates. Section 3.4.4 explains how the template mechanism recommends to the user an initial set of tool-kit techniques appropriate to the data and problem.

3.4.1 The template mechanism in setup

The operation of the template mechanism during the setup stage of CUES follows the steps that Figure 3.3 outlines.

To start utilizing CUES, the user supplies to CUES data sufficient for several iterations with the template mechanism. Each iteration runs over a subset of the data because CUES runs over a moving window of observations. Section 4.2.2 discusses selecting the size of this moving window. The user also provides CUES with basic information about the data and about the user's expectations. Section 3.4.4 lists this information and discusses how CUES utilizes it to propose an initial set of tool-kit techniques. It also explains how the template mechanism helps the user to further select from this subset.

The template mechanism repeatedly runs the selected techniques over a moving window of observations to infer predicates. It presents predicates true to any window along with observations that violate these predicate (anomalies) to the user and asks the user to classify the predicates. The user may classify a predicate as either "accept", "update", or "reject".

An inferred predicate is a complete instantiation of a template, but the classification may make the instantiation partial by rendering the instantiation of all the numeric values of one or more attributes void. For example, a technique may infer the predicate $52 \leq \texttt{cur} \leq 55 \wedge 48 \leq \texttt{dlow} \leq 55$. This is a complete instantiation of the template $\# \leq A_1 \leq \# \wedge \# \leq A_2 \leq \#$. A partial instantiation is one of

1. $52 \leq \texttt{cur} \leq 55 \wedge \# \leq \texttt{dlow} \leq \#$,
2. $\# \leq \texttt{cur} \leq \# \wedge 48 \leq \texttt{dlow} \leq 55$, or
3. (default) $\# \leq \texttt{cur} \leq \# \wedge \# \leq \texttt{dlow} \leq \#$.

The template mechanism includes the complete instantiation for "accept" predicates and the partial instantiation that the user chooses for "update" and "reject" predicates. The default partial instantiation is the one with only the template attributes instantiated (e.g., item 3).

The template mechanism treats the predicate inference techniques as black boxes and uses the instantiated templates to filter the predicates a technique infers. It constructs and updates the model of proper behavior from predicates that match instantiated templates of "accept" and "update" predicates. It will never present the user or the anomaly detector with predicates that match templates of "reject" predicates.

3.4.2 The template mechanism in tuning

CUES applies the tool-kit techniques that the user has selected in the setup stage to re-infer predicates over every data subset of the moving window of observations. It then utilizes only re-inferred predicates that match "update" templates. If the techniques do not re-infer a predicate matching the "update" template over the current window, it uses the most recently updated predicate. It ignores predicates that match "reject" templates. It checks "accept" templates on all the data subsets of the moving window ("accept" templates are predicates because they are fully instantiated). Therefore, CUES automatically tunes only the model parameters. CUES does not automatically add new predicates or remove existing predicates.

When the sanity-check heuristic indicates that the model is performing poorly for anomaly detection (i.e., the model identifies too many observations as anomalies over the current moving window of observations; Section 4.4.2 provides details about the sanity-check heuristic), CUES requests the user to intervene and takes the user back to the setup stage (Figure 3.1 depicts this). Unless the expected data behavior changes greatly, the user can continue to use the tool-kit techniques that the user has initially selected. CUES asks the user to re-examine the re-inferred predicates, and possibly change existing template classification or add new templates to the model (there may be newly inferred predicates that the user did not see in the initial setup stage).

The stock daily low case study (Section 5.3) provides an example of model tuning. CUES automatically re-inferred the coefficients of the MUSCLES linear regression predicates. It checked the Daikon predicates that the user classified as "accept" over all the data subsets. In the stock quotes case study (Section 5.2) the template mechanism did not yet exist. Therefore, CUES checked all the predicates that were inferred over the moving window of observations. This included predicates that would have been rejected had the template mechanism existed. Including these predicates in the model impaired the model performance for anomaly detection. To improve detection, CUES used a voting heuristic that required additional (redundant) data feeds.

When automatically tuning "update" predicates by re-inference, it may be the case that a predicate is not re-inferred over a data subset. This may happen because not all subsets contain the same data and because some subsets may contain more noise than others. For example, in the truck WIM case study (Section 5.1), the "update" templates of the Rectmix predicates included fixed numeric values for the number of axles. However, the values of the axles attribute were not identical in all subsets.

If a predicate corresponding to an "update" template is not re-inferred over a subset, CUES checks the most recently updated predicate instead. It warns the user about this when it alerts the user about the detected anomalies. This may indicate that an aspect of the behavior that the user thought was invariant is actually changing over time. The sanity check heuristic is likely to alert the user if the predicate used was not fit for the data subset.

3.4.3 Combining existing knowledge

In addition to supporting predicate inference, the template mechanism supports incorporating existing knowledge. It supports the addition of restrictions to existing templates and the addition of user-defined templates.

Adding restrictions to predicates

Often, there are intrinsic periods in the data. For example, the daily low value of a stock ticker is re-set every day and the traffic patterns of trucks traveling on a highway are likely to differ between weekdays and weekends.

The template mechanism, therefore, allows the user to choose restrictions or add restrictions. The template mechanism can then either augment existing predicates with these restrictions (infer-then-restrict) or apply these restrictions to pre-process the data (restrict-then-infer). To select such a restriction, the user need only have basic domain knowledge. Further, this kind of basic knowledge generalizes across domains.

The template mechanism currently supports daily, weekly, week-days, and weekend time-period restrictions. The data needs to include a `date` attribute in order to enable these restrictions (the day-of-week can be calculated from the date). For example, a tool-kit technique may infer a predicate about the daily low value of a stock ticker stating that: $\texttt{dlow}(t) \leq \texttt{dlow}(t-1)$. The user may add a "daily" time-period restriction. As a result, the template mechanism will create the augmented predicate: $\texttt{dlow}(t) \leq \texttt{dlow}(t-1) \wedge \texttt{date}(t) == \texttt{date}(t-1)$. CUES will check the augmented predicate to detect anomalies. The stock daily low case study (Section 5.3) provides an example of CUES' infer-then-restrict approach.

CUES also supports preprocessing to include only data conforming to the time-period selected by the user. CUES will then infer predicates only over data that belongs to the same time period. For example, a user may select a weekdays restriction for a truck WIM data. CUES will then infer predicates only over observations that correspond to weekdays and will ignore weekend observations. The stock quotes case study (Section 5.2) provides an example of CUES' restrict-then-infer approach: the data feeds only included observations from weekdays.

Both the infer-then-restrict and the restrict-then-infer approaches are useful. The restrict-then-infer approach requires enough observations in the selected period of time to support inference. However, it has the advantage of automatically inferring predicates that hold over the restricted period of time—the user may not think in advance of these predicates. Often, there are not enough observations in the restricted time period. It is then useful to allow the user to restrict predicates that the user knows should only hold selectively.

Supporting existing specifications

CUES supports adding limited forms of user defined predicates and existing specifications. CUES supports predicates and specifications that can be expressed in terms of existing templates with no additional user effort. The current syntax CUES supports is the superset of all the predicates the various tool-kit techniques can infer (Table 3.4 provides a summary)—a fairly general syntax. The user may add existing specifications that conform to this syntax as predicates. The template mechanism treats user predicates as "accept" predicates. For example, when interacting with the template mechanism, the user may realize that some values should not occur (e.g., `dlow` should be at least zero). The user may add the predicate $\texttt{dlow} \geq 0$.

The user may add existing specifications whose syntax allows stateless checking or checking with reference to previous observations in the current moving window—

discrete localized checking. To add specifications whose syntax conforms to these restrictions but is not already part of the existing templates, the user should add an appropriate null-inferrer technique to the technique tool kit. Section 3.3.6 provides a list of information the user needs to supply when adding a technique to the tool kit.

CUES takes a 'single-pass' approach. It is also possible to use the user-defined predicates to pre-process the data and then re-run the template mechanism on the cleaner data.

3.4.4 Selecting techniques

Different techniques make different assumptions and often use different vocabularies. Therefore, it may be useful to apply multiple techniques to the same data. However, a large number of techniques is likely to burden the human. In addition, data characteristics vary and may make some techniques more or less appropriate. 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.

Technique selection is done via template selection: if all the predicates associated with a technique are eliminated this technique is eliminated for the data of interest.

Template selection has two major stages. First, the template mechanism asks the user simple questions about the data measurement scale and about the correlation the user expects among the attributes of the data. It then recommends to the user tool-kit techniques that are likely to work well for the data. Details follow below.

Then, the user interacts with the template mechanism to further select templates. Different users may have different dependencies on the data and therefore different expectations. In addition, the output of one technique may be largely redundant with the output of another for the data of interest. The template mechanism eliminates techniques that are not relevant for this user and data: it will not employ a predicate inference technique if the user rejects all the predicates that are associated with this technique. Sections 3.5.1–3.5.5 provide an example of technique selection by the user for the truck weigh-in-motion data feed through interactions with the template mechanism.

The syntax of existing specifications may also provide guidance to the user in choosing techniques. The user may prefer techniques that can express at least parts of the existing specifications. For example, in the truck WIM case study (Section

5.1), the existing documentation included ranges. The techniques the user selected could express various forms of ranges.

Initial technique recommendation

Each technique is likely to be useful only for data with certain characteristics. This provides an initial technique filtering criterion. CUES utilizes the measurement scales of the data and the measurement scales that a technique supports for initial template selection. If a technique performs computations that are inappropriate to all the attributes of a data feed the technique should not be applied to that data feed.

Table 3.4 lists the minimal measurement scale that each of the tool-kit techniques requires for its input attributes (Daikon's predicates vary with respect to the measurement scale they are appropriate for; Section 3.3.4 provides details).

The template mechanism asks the user the following questions in order to determine the data measurement scale:

1. Is it meaningful to compare the size of differences between items? Yes: scale is interval. No: ask next question.

2. Is it true that a higher number means a higher value? Yes: scale is ordinal. No: scale is nominal.

The template mechanism provides guidance regarding the applicability of each of the tool-kit techniques, based on the assumptions each technique makes about the data. Table 3.4 lists the major assumptions the tool-kit techniques make.

Predicates over single attributes are always in the set of the user's expectations. Predicates over multiple attributes are only selectively in this set because they depend on the user's expectations for correlations. Therefore, CUES always recommends trying techniques that infer predicates over single attributes: Daikon and/or Percentile. For selecting techniques that infer predicates over multiple attributes (often from multiple data feeds) CUES applies the information in Table 3.4. To do so it utilizes basic domain knowledge about the correlation the user expects among the attributes/feeds. It gets this information by asking the following questions:

1. Is the expected correlation an inequality among attributes or a simple linear relation (involving no more than 3 attributes)? Yes: try Daikon.

2. Is the expected correlation linear, possibly involving correlations over time? Yes: try MUSCLES.

3. Are ranges meaningful? Yes: try Rectmix.

4. Is it meaningful to divide attribute values into bins and look at associations among these bins? Yes: try Association rules.

3.5 Templates for the tool-kit techniques

Sections 3.5.1–3.5.5 describe templates corresponding to the tool-kit techniques. These Sections also discuss strengths and weaknesses of each technique and provide an example of technique selection. The example is based on the truck WIM case study (Section 5.1). It describes the predicates the different techniques infer over the truck WIM data feed. The attributes it uses are the length of a truck `length`, a measure of the weight on an axle `ESAL`, the number of axles of the truck `axles`, the gross weight of the truck `weight`, and the speed the truck was traveling in when it crossed the WIM scale `speed`. The user's imprecise expectations in that case study could be summarized as: (1) trucks should be physically feasible and (2) trucks in the same vehicle class should be similar.

The template mechanism helped the user to make her expectations precise, by suggesting an initial subset of techniques (Daikon, Percentile, and Rectmix) and by letting the user select from the predicates the tool-kit techniques inferred. The technique-selection example shows predicates each of the tool-kit techniques infer over the WIM data, not just the predicates inferred by the recommended subset of techniques. The user selected predicates from two techniques—Rectmix and Percentile—because these predicates best described behavior that was relevant to her expectations. In addition, the syntax of the predicates outputed by these techniques matched the syntax of existing documentation of the WIM system. For this data feed and user, other techniques inferred predicates that were either less precise or redundant with respect to the predicates inferred by the techniques the user selected. Because Rectmix and Percentile differ, it is useful to apply both: Rectmix finds correlations among common attribute values whereas Percentile finds common values for a single attribute.

3.5.1 Rectmix templates

Rectmix (Section 3.3.1) 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 (i.e., dimensions).

Though clusters rarely have a hyper-rectangle shape in reality, Rectmix has the significant advantage of inferring predicates that are easy to understand: a hyper-rectangle is simply a conjunction of ranges, one for each attribute.

Table 3.5 gives an example of user classification for predicates that Rectmix infers over a subset of the WIM data. The corresponding templates have numeric values in one dimension—the axle attribute—because the user chose to void the other attribute values. For example, the template for the first predicate is $\# \leq \texttt{length} \leq \# \wedge \# \leq \texttt{ESAL} \leq \# \wedge 3 \leq \texttt{axles} \leq 3 \wedge \# \leq \texttt{weight} \leq \#$. Figure 3.4 depicts three of the four dimensions of the Table 3.5 clusters (i.e., hyper rectangles): length (feet; x-axis), weight (kilo pounds; y-axis), and number of axles (shade and shape). For each rectangle it depicts the bounding rectangle (the result of adding the selected number of sigmas to each dimension) from Table 3.5.

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

Table 3.5: Example of Rectmix predicates classification

3.5.2 Mean and Percentile templates

Mean and Percentile (Section 3.3.2) have a single template: $\#\leq A \leq\#$.

Mean and Percentile produce the same kind of predicates: a probable range for the values of each attribute. However, Percentile makes fewer assumptions about the data distribution.

Table 3.6 gives an example of user classification and resulting instantiated templates for predicates 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).

3.5.3 Association Rules templates

Association rules (Section 3.3.3) 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.

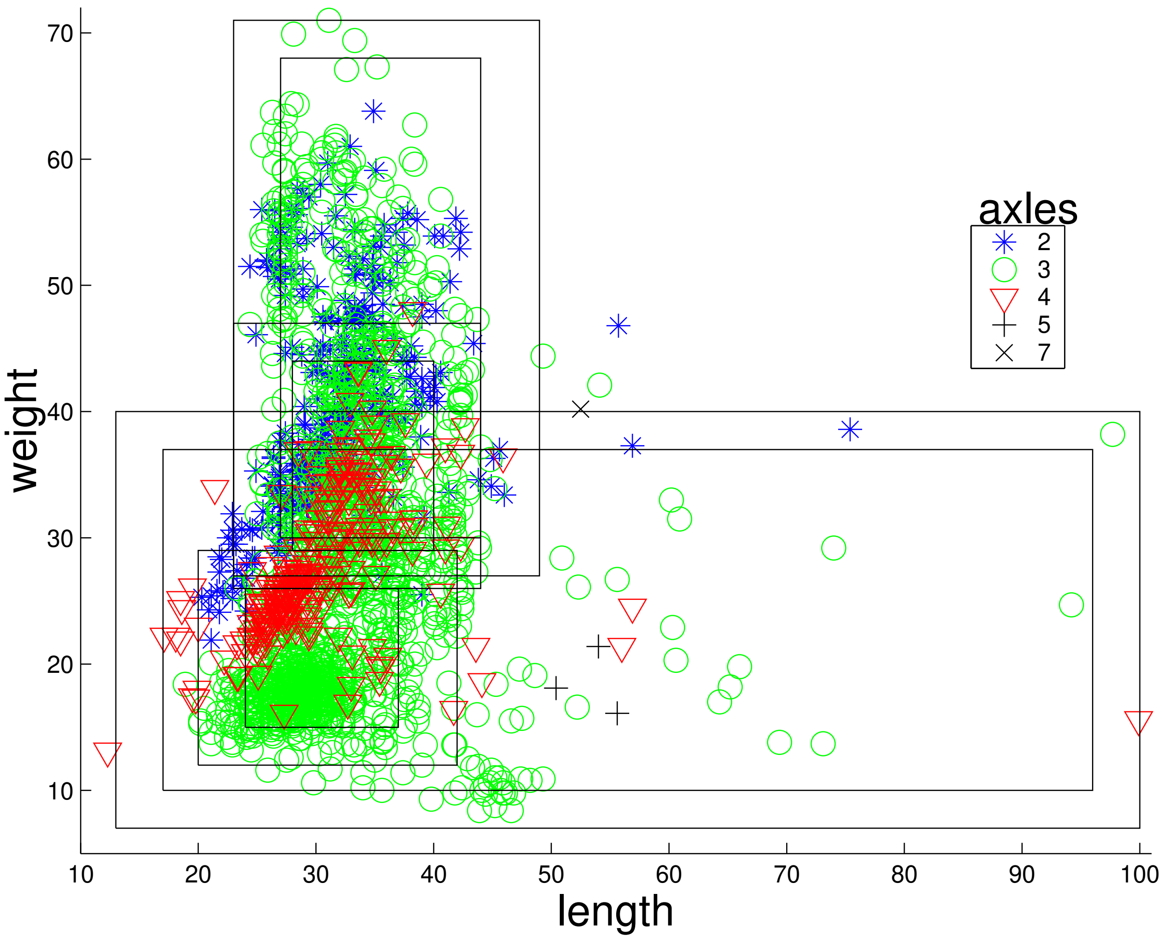

Figure 3.4: The rectangles of Table 3.5 in three dimensions

The rules reflect correlations among attributes but these correlations may not reflect accurately cause and effect relations. 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.

For the WIM data, association rules work rather well. An example of the rules they produce is 'if `length` $< 34.8 \wedge$ `ESAL` $< 0.11 \wedge 2 \leq$ `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: `ESAL` is calculated based on the other attribute values). In general, Rectmix performs better over the WIM data.

Class	Predicate	Template
Update	$40 \leq$ `speed` ≤ 88	$\# \leq$ `speed` $\leq \#$
Update	$17 \leq$ `length` ≤ 39	$\# \leq$ `length` $\leq \#$
Reject	$.06 \leq$ `ESAL` $\leq .9$	$\# \leq$ `ESAL` $\leq \#$
Update	$3 \leq$ `axles` ≤ 3	$\# \leq$ `axles` $\leq \#$
Update	$12 \leq$ `weight` ≤ 49	$\# \leq$ `weight` $\leq \#$

Table 3.6: Example of Percentile predicates with user classification and instantiated templates

3.5.4 Daikon templates

Daikon (Section 3.3.4) has templates for each of its pre-defined relations. The template mechanism currently supports a subset of Daikon's predicates—the predicates most relevant for the purpose of anomaly detection over data feeds. These templates are (Items correspond to the predicates listed in Section 3.3.4):

1. $A_i == A_j$, $A_i == \#$
2. $A_i \neq A_j$, $A_i \neq \#$, $A_i \leq A_j$, $A_i \leq \#$, $A_i < A_j$, $A_i < \#$, $A_i > A_j$, $A_i > \#$, $A_i \geq A_j$, $A_i \geq \#$
3. $A_i = \#A_j + \#$, $A_i = \#A_j + \#A_k + \#$
4. $A_i \in \{\#\}$

Daikon is very effective when strong correlations that can be described by its pre-defined relations exist.

Because the WIM data has mostly statistical correlations, the only useful predicates Daikon outputs are of the form `axles` $\in \{\#\}$. However, Rectmix and Percentile produce similar predicates.

3.5.5 MUSCLES templates

MUSCLES (Section 3.3.5) has a single template:

$$\begin{aligned}\hat{A}_1(t) &= \#A_1(t-1) + \ldots + \#A_1(t-w) \\ &+ \#A_2(t) + \#A_2(t-1) + \ldots + \#A_2(t-w)\end{aligned}$$

$$
\begin{aligned}
&+ \quad \ldots \\
&+ \quad \#A_k(t) + \#A_k(t-1) + \ldots + \#A_k(t-w)
\end{aligned}
$$

where a hat over an attribute ($\hat{A}$) indicates an estimated value, k is the number of sequences, w is the window size, and t is the current time tick.

The predicate MUSCLES infers is simply a linear relation. Often, a subset of the attributes have coefficients that are significantly larger than the coefficients of the other attributes. In these cases, the user may ignore attributes with small coefficients (because their influence on the estimate is negligible). The user may interpret the MUSCLES predicate as stating that the value of the estimated attribute is mostly influenced by the values of the attributes with the large coefficients. Moreover, this correlation takes the form described by the coefficients.

To a first order approximation, there are no temporal changes in the WIM data, so MUSCLES is not an appropriate technique for that data.

3.6 Anomaly detector

Section 2.1 defines an anomaly. We repeat that definition here: given a model and a tolerance (i.e., threshold or confidence), an anomaly is an observation that has attribute values that are different from the model by more than the tolerance.

It follows from the definition of an anomaly that, given a model of proper behavior and a tolerance, detecting an anomaly is straightforward:

1. Evaluate the model predicates over attributes in current data feed observations.
2. Warn the user about an anomaly when attribute values violate any of the predicates by more than the tolerance.

CUES assists the user in creating the model of proper behavior: the user selects an initial model during the setup stage. CUES then repeatedly updates the model in the tuning stage. The template mechanism (Section 3.4) plays an important role in both initial model creation and updating the model. The tolerance is part of the information the technique tool kit (Section 3.3) supplies for each technique. CUES then adds the tolerance to the predicates it infers. The checking code evaluates each predicate over each observation (for stateless predicates) or over the N recent observations (for stateful predicates with a window size N) in the current moving window. It reports as anomalous observations that violate any of the predicates.

Chapter 4

CUES: scope, "background" processing, and related work

The CUES stages—setup, checking, and tuning—are a process governing the execution of the CUES mechanisms—technique tool kit, templates, and anomaly detector—as Section 3.1 explains. Chapter 3 concentrates on the CUES mechanisms. CUES' stages also include "background" processing. This background processing provides support to the stages beyond the CUES mechanisms. It helps in providing the appropriate data to the mechanisms, in reporting their results, and in controlling their execution. Section 4.1 captures the technical problem that CUES addresses and scopes it. Sections 4.2, 4.3, and 4.4 present background processing that CUES may perform during the setup, checking, and tuning stages, respectively. These Sections also present other work most closely related to each of the stages.

4.1 Scope

The main technical problem that CUES addresses is: given data feeds, with numeric valued attributes, build a model to detect semantic anomalies in these data feeds. In our setting, if a model exists then detecting semantic anomalies is straightforward.

Section 4.1.1 discusses the assumptions CUES makes about its users. Section 4.1.2 discusses characteristics of the data that CUES is appropriate for.

4.1.1 Users

CUES relies on the user. It provides automated assistance to users in creating the above-mentioned model. It does so by helping users make their expectations for data feeds behavior precise.

CUES assumes the user may be a *secondary user*. Secondary users are users who may use a data feed for purposes other than those originally envisioned by the data feed providers. An example is using river gauges data feeds for recreational purposes rather than for flood prediction. Secondary users should have sufficient domain knowledge to select predicates that express their expectations for the behavior of the data feeds they are using. However, they cannot be expected to be experts in modeling techniques, such as time series analysis or data mining. They also cannot be expected to have the expertise required to model the underlying phenomena. For example, a river gauges user does not need to be a hydrologist in order to be able to decide whether the river conditions are fit for canoeing.

CUES incorporates existing predicate inference techniques to help in making user expectations precise. CUES incorporates predicate inference techniques that output predicates that are easy for a human to understand. Examples of such predicates are linear relations and ranges. It cannot incorporate techniques that require much expertise to set up or to interpret, such as many time series analysis or digital signal processing techniques. This thesis experiments with a few existing inference techniques (the tool kit techniques; see Section 3.3).

4.1.2 Data characteristics

CUES is appropriate for everyday information resources: resources whose usage can tolerate recognition and repair of faults, rather than rely solely on prevention.

This thesis concentrates on the benefits achievable by using limited domain knowledge: knowledge of secondary users about the application domain. CUES can be viewed as an example of the 80–20 engineering rule. CUES is semi-automated with limited user effort. It provides much benefit for this effort. It helps the user to create a simple yet precise model from the user's expectations. However, creating a precise model that describes entirely the user's expectations often requires much user effort because some expectations are domain knowledge intense. This is beyond the scope of CUES.

CUES can model simple correlations among attributes of a single data feed or among attributes of multiple synchronized data feeds. For example, it can find that the daily value of a stock should always be no more than its daily high value. However, it cannot model the underlying phenomenon, such as stock market behavior. This is a result of the predicates that CUES can infer.

4.2 The setup stage

Prior to executing the template mechanism, CUES may need to perform data preprocessing and to set parameters. Section 4.2.1 lists possible data preprocessing. Section 4.2.2 discusses parameter setting. Section 4.2.3 summarizes other work most closely related to the setup stage.

4.2.1 Data preprocessing

CUES may need to pre-process the data before the template mechanism interacts with the user. This includes handling multiple data feeds, performing other data transformations, selecting attributes, and clustering. Most preprocessing actions are automated—the user only needs to select from the preprocessing actions. User participation may be necessary for attribute selection and clustering.

Handling multiple feeds

CUES supports predicate inference over multiple data feeds. The data feeds may be loosely synchronized, as long as the resulting time lags are either fixed or small compared to the sampling interval. By loosely we mean that it is possible to match observations of multiple feeds so that the time stamp of this *united observation* is reasonably accurate for all the feeds. The user needs to do this matching before utilizing CUES. The stocks daily low case study (Section 5.3) provides an example of such loosely synchronized data feeds: the user sampled the data feeds at identical time intervals and matched observations according to the time in which they were sampled. Therefore, each united observation included the daily low value at time t from each of the three data feeds that provided the daily low value of the same stock ticker symbol.

If a time lag in the data exists, CUES creates a *mega observation* for techniques that infer stateless predicates (all the tool-kit techniques except MUSCLES). This results in enabling these techniques to infer stateful predicates without changing the techniques. Techniques that infer stateful predicates already have a built-in ability to take into account multiple observations. A mega observation consists of the values of the attributes of all united observations in a time window (so it includes multiple united observations). Section 4.2.2 discusses how to find the size of this time window. For example, in the daily low case study, each mega observation in a time window of size one included the current daily low value (value at time t) from each of the feeds and the previous daily low value (value at time $t-1$) from each of the feeds.

Data transformations

Data transformations are straightforward and mostly automated. CUES provides automation support for two transformations: normalizing each numeric valued attribute to have mean zero and standard deviation one and dividing data into bins. Both transformations are common and supported by many existing data exploration tools. CUES uses the Auton lab [Auton, 2003] database processing and analysis software (SPRAT).

Normalizing Normalizing the values of an attribute is preformed by subtracting the estimated mean from each value of the attribute and dividing by the estimated standard deviation. The estimated mean $\bar{X}$ is the sample mean: $\bar{X} = \frac{1}{n}\Sigma_{i=1}^{n} X_i$, where n is the sample size. The estimated standard deviation σ' is the square root of the sample variance: ${\sigma'}^2 = \frac{1}{n-1}\Sigma_{i=1}^{n}(X_i - \bar{X})^2$.

Normalizing is necessary for techniques that assume similarly scaled attributes (K-means, PCA, and MUSCLES; a description of K-means and PCA follows in the discussion of clustering) and data with differently scaled attributes (e.g., the WIM data).

Binning There are multiple ways for dividing the values of an attribute into bins. SPRAT supports dividing values into bins of equal width, creating bins of possibly non-equal width but making sure the number of observations in the various bins are not wildly skewed, reading in numbers from the user on where to put the breaks between bins, and rounding all the real numbers to their nearest integer value and making one bin per integer. The user must select one of the above ways of dividing the data into bins and must indicate the number of bins for all but the last binning method.

Dividing data into bins is necessary for techniques that require categorical data

(Association rules) and data with numeric valued attributes.

Attribute selection and clustering

Attribute selection is useful for techniques that infer multi-dimensional predicates (all tool-kit techniques except Mean and Percentile) both because these techniques tend to work better with fewer 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 (according to a classification criterion that is relevant to the user's expectations and therefore the user can reason about), 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. The truck WIM case study (Section 5.1) provides an example of data with different classes. Many attribute values are similar for trucks of the same vehicle type but different across types. The data behavior the user expects depends on the type. Therefore, CUES interacts with the user separately for each type.

The measurement scales of attributes serve for initial attribute selection: each tool-kit technique runs only on attributes with an appropriate measurement scale. Section 3.4.4 explains how CUES utilizes measurement scales. The user may select attributes either manually or with the aid of automated attribute selection techniques. An example of an automated technique that was useful in the truck WIM case study follows.

Clustering algorithms automatically assist in finding classes in the data. Examples are Rectmix, K-means, and Principle Component Analysis (PCA). Rectmix is part of CUES' technique tool kit (Section 3.3). It serves not only for predicate inference but may also provide automated assistance to the user in finding classes in the data. A description of K-means and a description of PCA together with an example of finding classes using PCA follows.

K-means K-means [Duda et al., 2000] is a clustering algorithm. It partitions points into distinct clusters. K-means assumes attributes are similarly scaled (it applies the Euclidean distance metric) and clusters have a hyper-sphere shape. It is appropriate for interval scale data. K-means has one parameter: the number of clusters. The X-means algorithm [Pelleg and Moore, 2000] is an extension of the K-means algorithm that automatically finds the best number of clusters according to the information criterion X-means uses (BIC—Bayesian Information Criterion).

PCA PCA [Jolliffe, 1986] 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, CUES runs PCA on the data and plots the observations along the first two principle components. To check for clusters, CUES plots 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).

4.2.2 Parameter setting

CUES utilizes the tool-kit techniques to infer predicates over a moving window of observations. These techniques often have parameters whose values affect the performance of the techniques over a specific data feed. In addition, the size of CUES' moving window determines the size of the training data that CUES provides to these techniques. Therefore, the size of the moving window may also affect the performance of the techniques.

Selecting technique parameter values

The technique tool kit (Section 3.3) provides default parameter values for each of its techniques. Alternatively, the user may set parameter values. The template mechanism assists the user in determining parameter values by running each technique with several values for each parameter and letting the user select the combination that best reflects the user's expectations. The complete list of technique parameters, including the default CUES provides, appears with the description of the tool-kit techniques in Section 3.3. For example, the template mechanism runs Rectmix (Section 3.3.1) 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 (Section 3.3.2) 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%.

Selecting a time window size

CUES supports stateful predicates for multiple data feeds with a time lag, by providing the tool-kit techniques input data that has multiple observations (Section 4.2.1 discusses CUES' preprocessing to create this data). To find the number of observations, CUES looks for a window that includes observations over a period of time similar to the time lag.

It is common practice to apply information criteria for finding a window size (e.g., [Box et al., 1994]). Finding a window size is an example of model selection: choosing between a range of hypotheses that explain the situation. Information criteria are a way of selecting models with a lot of explanatory power but without an excessive number of parameters [Information Criteria, 2004]. Information criteria achieve this by penalizing the log likelihood[1] of the models with a term that tries to capture the model complexity. Typically this is measured as: $-2\ln(L) + A * K$, where L is the likelihood of the model, K is the number of free parameters (the model order), and A is a metric-specific term. The smaller the result of this calculation, the better. For example, in the Bayesian Information Criteria (BIC) the metric-specific term is $\ln(N)$, where N is the number of observations (the sample size), so BIC is: $-2\ln(L) + \ln(N) * K$.

CUES uses an existing time series analysis package [TSA, 2003], supporting many information criteria, including: Final Prediction Error (FPE [Kay, 1988]), Akaike Information Criterion (AIC [Marple, 1987]), Bayesian Information Criterion (BIC [Wei, 1990]), Schwartz's Bayesian Criterion (SBC [Wei, 1990]), and Minimal Description length Criterion (MDL [Marple, 1987]). CUES' default is BIC. The user may select a different criterion or may manually enter a different window size. Often, manual investigation results in decreasing the size of the window in order to make the resulting predicates easier to interpret. For example, in the stocks daily low case study, CUES recommended a window of size 9 but the user chose to decrease the

[1] Likelihood is the probability of observing a particular data value D, given a model M and a set of values O for the parameters of that model: $P(D \mid O, M)$.

window size to 1 (details in Section 5.3.2; for the stock daily low data, the MUSCLES coefficients were either very small or canceled each other in all but the current and previous observations.)

Selecting a moving window size

CUES recommends to the user an initial size for the moving window, according to the minimal statistical requirements of a technique. A common rule of thumb and the one CUES applies is that techniques that do statistical inference should get at least 30 observations per attribute, and preferably around 100 or more. Daikon should get around 10 observations per attribute (this can be controlled within Daikon). CUES' defaults are 30 observations for Daikon (voting over 3 subsets with 10 observations each) and 100 observations for the other tool-kit techniques.

The user may change the default window size for each technique by manual investigation. Empirically, it has been the case that the same window size worked well for multiple techniques for the same data feed. The template mechanism helps the user to decide on a reasonable window size: it enables the user to experiment with various window sizes, see the predicates that are inferred, and understand their consequences by looking at the anomalies resulting from checking these predicates.

4.2.3 Related work

Work most closely related to CUES' setup stage includes approaches that either perform dynamic analysis based on observable behavior or have a similar emphasis on users and their intent. The difficulties of creating a model for anomaly detection can be viewed as an instance of Kolmogorov complexity.

Inferring characteristics from behavior

The CUES approach of inferring the characteristics of a data feed from its behavior is similar to work in the areas of program analysis [Ernst et al., 2001, Engler et al., 2001, Ammons et al., 2002] and testing [Dickinson et al., 2001]. However, these naturally have a different domain, and they often concentrate on a single technique. Daikon [Ernst et al., 2001] dynamically discovers likely program invariants from program executions. CUES incorporates Daikon in the technique tool kit. "Bugs as deviant behavior" [Engler et al., 2001] infers beliefs from source code, so it is inappropriate for data. "Specifications mining" [Ammons et al., 2002] uses a machine learning approach for discovering specifications of interface protocols. However, it uses techniques specific to code. "Observation-based testing" [Dickinson et al., 2001] uses clustering and visualization techniques over program execution profiles to identify unusual executions. CUES has similar techniques in the tool kit.

Emphasizing users role

CUES takes a user-centered view of dependability: to increase the dependability of data feeds CUES provides assistance to users in making their expectations about data feed behavior precise. Other work has a similar emphasis on the role of users and their intent.

Behavioral subtesting [McCamant and Ernst, 2003] aims to predict problems caused by component upgrades and is based on observing that the way the software is being used greatly influences effective prediction. CUES is based on a similar observation about the importance of usage. Operational profiles capture program behavior as mathematical predicates. The syntax of CUES' precise expectations is similar. Behavioral subtesting requires source code or binaries, as well as cooperation from the software providers. CUES only requires observable output (a data feed).

Aura's model-based adaptation approach [Sousa and Garlan, 2002] recognizes the importance of modeling the user's intent. However, Aura concentrates on mobile users and therefore on enabling a computing task within the given computing resources. Aura assumes the service/operating system provides hooks for the adaptation framework whereas CUES assumes the user has no control over the data feed.

Langley [Langley, 2000] 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. CUES' template mechanism can be viewed as a way to help users to do this. However, its goal is eliciting users' expectations instead of finding new scientific knowledge.

Lapis [Miller and Myers, 2001], a tool for lightweight structured text processing, includes an outlier finder as a way to focus human attention where human judgment is needed. CUES utilizes anomaly detection in a similar way, though the domain is different.

Kolmogorov complexity

Ideally, our problem definition would be: given a data feed, find the generative model of the data. This minimal description pattern would be the best compression of seen data: it would not only describe all possible behavior but also describe it in the most succinct way. If this generative model is predictive (i.e., future data is also generated from the same patterns) then anomalies would be deviations from this model. Our work assumes that future data is generated from patterns similar to those generating the seen data.

Finding the minimal description pattern is the Kolmogorov complexity: the complexity of a string of data as defined by the length of the shortest universal Turing machine program for computing the string [Kolmogorov, 1965]. This definition is universal—it is computer independent. Thus, this complexity is the minimal description length. Formally [Cover and Thomas, 1991], let x be a finite length binary string, U a universal computer, $l(x)$ the length of x, and $U(p)$ the output of the computer U when presented with a program p. The Kolmogorov complexity $K_u(x)$ of a string x with respect to a universal computer U is defined as $K_u(x) = min_{p:U(p)=x} l(p)$, the minimum length over all programs that print x and halt. Thus $K_u(x)$ is the shortest description length of x over all descriptions interpreted by computer U.

Kolmogorov complexity is non-computable—it has been reduced to the halting problem. The only way to find the shortest program in general is to try all short programs and see which one can output the desired string. However, at any given time, some of these programs may not have halted, and there is no effective way to tell whether they will halt and what their output will be.

Therefore, in general, we cannot expect to find the best (generative or minimal description) model for a data feed, or will be unable to know we found it. Because of the non-computability of the problem, we look for a collection of useful (good enough) techniques, rather than for the best one. The problem definition becomes: given a data feed, find patterns that are good enough in compressing the data.

4.3 The checking stage

CUES continuously checks observations in the current moving window of observations. It evaluates the predicates in the model over each observation (or over the last n observations for stateful predicates). If the result is different from the actual values by more than the threshold it flags the observation as anomalous. Section 4.3.1

provides details about the way CUES reports the anomalies to the user.

When reporting anomalies—the result of using the model for anomaly detection—and when updating the model (Section 4.4.2), CUES calculates the detection rate. The detection rate is an objective measure for the performance of the model. In our assessment of the usefulness of the model for anomaly detection we also calculate, together with the user, the misclassification rate. The misclassification rate is a subjective measure for the performance of the model. Section 4.3.2 defines the detection rate and the misclassification rate. Section 4.3.3 presents other work most closely relate to the checking stage of CUES.

4.3.1 Reporting anomalies

To minimize user distraction, CUES checks all the observations in the moving window, aggregates warnings, and warns the user about all anomalous values at once. The number of these warnings varies greatly and depends on the data feed and on the observations in the particular moving window. For example, in the truck weigh-in-motion case study (Section 5.1), the average detection rate was less than 10%. Occasionally, the observations in a particular moving window contained a lot of anomalies, resulting in a detection rate that was close to 50%(!). In the validation study (Section 6.2), the average detection rate (of the model reviewer A selected) was 0.1%. The detection rate was consistent (and low) over all the windows. CUES provides the user with a quantitative summary, a list of anomalous observations along with the predicates that flagged them as anomalous, and a graphical presentation of the data and anomalies.

The detection rate provides a quantitative summary of the detection. Section 4.3.2 that follows defines the detection rate. The graphical presentation shows attribute values for all the observations in the moving window of observations and marks anomalies with a distinct marker symbol. Figure 4.1 provides an example from the stock daily low case study (Section 5.3). The X-axis is the sequential observation index (from one to the number of observations in the window). The Y-axis is the value of an attribute. The values of each attribute (daily low) are indicated with the same color and marker symbol. An anomalous value is marked by a black asterisk. The two plots depict the anomalies detected by checking predicates that two different tool-kit techniques infer (MUSCLES in the top plot, Daikon in the bottom plot).

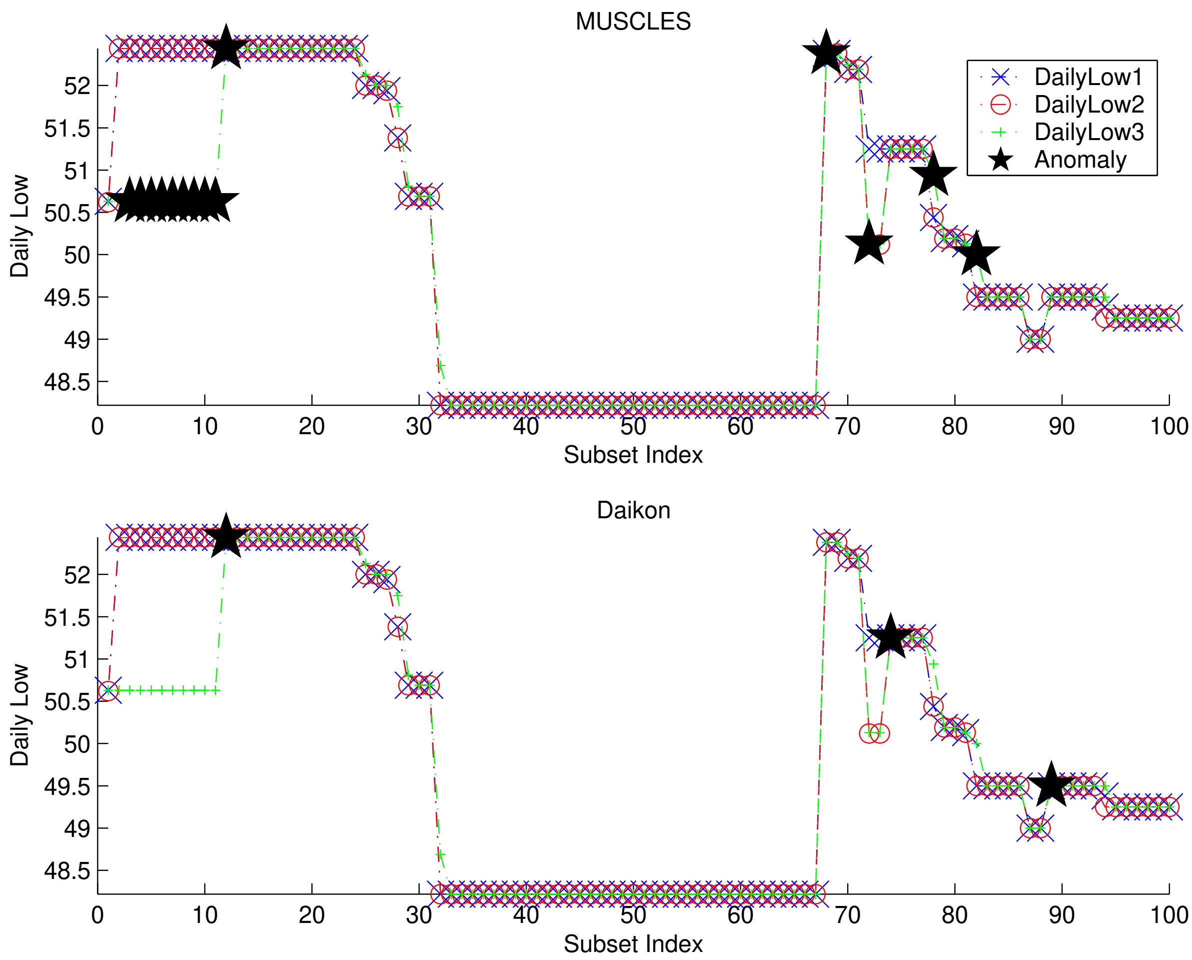

Figure 4.1: An example of highlighting anomalies detected by checking MUSCLES predicates (top) and Daikon predicates (bottom)

4.3.2 Detection rate and Misclassification rate

Detection rate and *misclassification rate* quantify the detection and the usefulness, respectively, of the proper behavior model for anomaly detection.

Figure 4.2 depicts the parts constituting the result space of detection. By using the model for anomaly detection we get:

- True Positives (TP): correctly detected anomalous data (both anomalies that indicate failures and anomalies due to uncommon yet normal behavior; see discussion in Section 2.2)
- False Positives (FP): normal data falsely detected as anomalous
- False Negatives (FN): undetected anomalous data

- True Negatives (TN): correctly detected normal data
- Normal (Nor): Nor=TN+FP; all data that is actually anomaly-free
- Abnormal (Ab): Ab=TP+FN; all data with actual anomalies

To normalize the results to the range $[0, 1]$, we divide TP and FN by Ab, and TN and FP by Nor. Diagonal quantities in Figure 4.2 are dependent—they are 1-complements. We need only display the independent measures. We chose the first row: *TP rate* and *FP rate*. These are identical to one minus the type I error rate (FN/Ab) and the type II error rate (FP/Nor) defined in [Runeson et al., 2001]. When Ab=0 (entailing TP=0,FN=0) we define TP/Ab=1 and FN/Ab=0, to maintain the complement relation. Similarly, when Nor=0 we define TN/Nor=1 and FP/Nor=0.

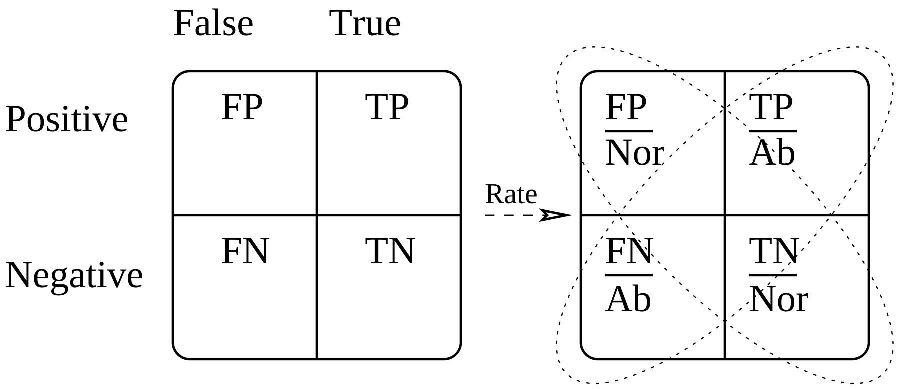

Figure 4.2: Result space. Right shows how we normalize the results. The ellipses mark 1-complements.

Determining the above measures is subjective because we do not have independent information on correctness. Moreover, it is subjective even if documentation for the data feed 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 the tool-kit inference techniques cannot express.

We seek accurate anomaly detection techniques, those with low FP,FN and high TP,TN. There is a tradeoff between FP and FN. To achieve a FP rate of 0 (no false positives) a technique could mark all attributes as normal. Similarly, to achieve a FN rate of 0 (no false negatives) a technique could mark all attributes as anomalous. In general, changing the model to include predicates that produce a better FP rate often results in a worse FN rate, and vice versa.

The detection rate $\frac{TP+FP}{Nor+Ab}$ 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. (notice that this measure only uses the model, it does not require an independent distinction between normal and anomalous).

The overall misclassification $\frac{FP+FN}{Nor+Ab} = \frac{Ab+FP-TP}{Nor+Ab}$ [Runeson et al., 2001] (lower is better) summarizes the TP rate and FP rate. It is a subjective measure (notice that this measure requires a distinction between normal and anomalous). We concentrate on whether the user cares about the anomalies the model detects, not on whether the model detects all the anomalies.

The information retrieval community often uses the measures of precision and recall to quantify the performance of a document retrieval system. In our result space, precision would be $\frac{TP}{TP+FP}$ (the number of true anomalies the system detects out of the total number of anomalies the system detects) and recall would be $\frac{TP}{TP+FN}$ (the number of true anomalies the system detects out of the total number of true anomalies in the data). Both these measures are subjective. We prefer the measure of misclassification rate because, though it relies on making the same judgments about the data, it stresses the performance aspect that results in a burden on the user (dealing with false alarms). The measure of detection rate does not require making these judgments (it is objective, given a model).

4.3.3 Related work

Increasing dependability includes prevention and detection/mitigation of problems. CUES' checking stage concentrates on detecting semantic problems by the client of a data feed. In general, prevention and detection are complementary as complete prevention is rare.

Preventing problems

Various approaches address preventing connectivity, syntax and form, and semantic problems in online data feeds.

CUES assumes the proprietor of a data feed may not be aware of its consumers. In addition, CUES recognizes the actual data behavior might differ from predefined semantics. The Semantic Web [W3C. Semantic Web, 2001, Berners-Lee et al., 2001] is a grand vision for a comprehensive solution to syntax/form and semantic failures. It is an extension of the current web that recognizes the distributed nature of services and content, and requires many additions to the current Web. It can enable one not only to use content from diverse sources but also to share the results with other programs (possibly written by other people). The Semantic Web uses XML to express structure, RDF (Resource Description Framework) to express meaning, and ontologies to enable agreement regarding meaning. The ontologies provide not only a taxonomy and a set of inference rules but also equivalence relations (to address the issue of using different terms to describe the same concept).

Our work is concerned with data quality. Our emphasis is on measuring and increasing the quality of the data by the consumer of this data. However, most of the research related to data quality is concerned with measuring and increasing data quality by the producer of the data: providing data quality dimensions for reasoning about data quality in the design of information systems [Wand and Wang, 1996], identifying the importance of data consumers when defining data quality [Strong et al., 1997], and discussing different perspectives of data quality for information products and businesses [Lee et al., 2000].

Web Services promote the same new development paradigm we do: building applications using as elements services available on the Web. Web Services use XML-based protocols that support discovery of services [Agrawal et al., 2001] and automatic information exchange and integration. This is mainly related to connectivity and syntax/form failures, addressing the interface problem.

Examples of Web Services include Microsoft's .NET, IBM's dynamic e-business (e.g., WebSphere Application Server, Web services Toolkit), and Oracle's dynamic services. Microsoft .NET is based on SOAP (Simple Object Access Protocol), XML (Extensible Markup Language), and CLR (Common Language Runtime). IBM bases its solution on UDDI (Universal Description, Discovery, and Integration), SOAP, and WSDL (Web Services Description Language), all XML-based technologies that have been advanced as Internet standards. To achieve dynamic integration there is a need to enable not only connection between applications (e.g., via the Web Services standards SOAP, WSDL and UDDI) but also to support transactions [IBM Web Services, 2001].

The protocols that Web Services are based on define allowable syntax and its semantics, resulting in automated support for identifying and possibly preventing aberrations in the data. XML (Extensible Markup Language) [W3C XML, 2001] is the universal format for structured documents and data on the Web. SOAP (Simple Object Access Protocol) [W3C SOAP, 2001] provides a mechanism for exchanging structured and typed information in a decentralized, distributed environment using XML. SOAP defines a mechanism for expressing application semantics by providing a modular packaging model and encoding mechanisms for encoding data within modules. SOAP can potentially be used in combination with a variety of other protocols; it is often used in combination with HTTP and HTTP Extension Framework. UDDI (Internet-based Universal Description, Discovery, and Integration) [UDDI, 2001, UDDI, 2000] is a building block for dynamically finding services and transacting between services. UDDI provides four core types of information: business information, service information, binding information, and information about specifications for services. The latter is a mechanism for finding and matching a given specification. It is not the specification itself. It is meta data about a specification (called "tModels" and includes name, publishing organization, and URL pointers to the actual specifications). UDDI uses a "retry or fail" approach to recover after a remote Web service failure. WSDL (Web Services Description Language) [W3C WSDL, 2001] is a descriptor standard used to describe service-offerings to the UDDI registry in a standard format. WSDL defines an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages.

Detecting/Mitigating problems

Various approaches address detecting and mitigating connectivity, syntax and form, and semantic problems in online data feeds. Detection/mitigation is necessary for all problem types. Solutions exist for detecting and mitigating specific connectivity [GoZilla, 2001, Alexa, 2001, Google, 2001, Akamai, 2004]. Solutions also exist for detecting and mitigating syntax and form [Knoblock et al., 1999, Bauer and Dengler, 1999, Kushmerick, 1999, Cypher, 1993] problems. However, solutions to detecting and mitigating semantic problems are scarce and either require having domain knowledge [Raman and Hellerstein, 2001, Knoblock et al., 1999, Bauer and Dengler, 1999, Kushmerick, 1999] or providing a specific technique [Gritbot, 2002].

Outlier or anomaly detection is part of data mining. CUES provides a framework to allow users to incorporate their favorite outlier-finding technique in CUES. Users may do so by adding the technique to CUES' technique tool kit.

Many outlier identification methods define a measure of "outlier-ness" to support automated identification. Papadimitriou et al. [Papadimitriou et al., 2003b] classify these methods into the following categories.

- Distribution-based approaches: methods based on a standard distribution model.
- Depth-based approaches: methods based on computations related to convex halls [Johnson et al., 1998].

- Clustering approaches: methods based on clustering algorithms [Duda et al., 2000].
- Distance-based approaches: proposed by Knorr et al. in a large body of work, an example of which is [Knorr et al., 2000]. Other distance-based methods include [Ramaswamy et al., 2000] and [Stephen Bay, 2003]. These methods define an outlier based on single global criterion .
- Density-based approaches: proposed by Breunig et al. [Breunig et al., 2000] as the Local Outlier Factor (LOF).

Other methods for evaluating outlier-ness include the Local Correlation Integral (LOCI) [Papadimitriou et al., 2003b]. In addition, there exist various tools for finding anomalies in a dataset to prepare it for data mining. An example is GritBot [Gritbot, 2002]. Currently, CUES' technique tool kit includes distribution-based approaches, a clustering approach, and a distance-based approach. To make the syntax of the predicates the tool-kit techniques infer richer, it may be a good strategy to explore techniques in the other categories.

Outlier-finding methods that make allowances for the intentions of users include interactive methods such as Potter's wheel [Raman and Hellerstein, 2001] and Outlier By Example (OBE) [Zhu et al., 2004], a method that relies on examples from the user. Potter's wheel is an interactive data cleaning system that includes transformations (addressing syntax/form problems) and a discrepancy detector. The users need to define domains and algorithms to enforce domain constraints for the discrepancy detector. The system automatically infers appropriate structure for columns in the data and applies the appropriate domain algorithm to detect anomalies. Outlier By Example (OBE) finds outliers in large data bases. OBE makes the observation that the notion of what an outlier is varies among users, problem domains, and datasets. OBE assumes the users are experts in the problem domain, not in anomaly detection. They may have examples of outlier that describe their intentions but want to find more objects similar to these examples. OBE can discover outliers at a scale indicated by examples the user provides.

There exist many domain-specific algorithms for detecting outliers (anomalies). Domains that have attracted much attention recently include intrusion detection [Sequeira and Zaki, 2002, Hofmeyr and Forrest, 2000, Lane and Brodley, 1998] and disease outbreak identification [Wong et al., 2003].

Information integration research provides data integration techniques to answer user questions based on information obtained from many different sources. We explore the challenge of dependability assessment, preservation, and enhancement of a future with integration systems in place. In this future one will [Levy et al., 1998] “use the data to automate some of the tasks we routinely perform with the data”and “help users rapidly construct and extend their own web-based applications out of the huge quantity of data available online”.

Data scrubbing (aka data cleansing) is often concerned with enterprise customer data [Trillium, 2001, FirstLogic, 2001]. As a result, tools for data scrubbing are often domain specific: for example, tools for correcting, standardizing, and verifying address elements [FirstLogic, 2001]. A main concern in data cleansing is data integration. Tools often have a component of data discovery that addresses a syntax and form problem (parsing records that have varying formats) [Paladyne, 2001]. There are also data cleansing products that operate according to a set of rules [Datanomic, 2001].

Failure detection is an aspect of fault tolerance. Fault tolerance approaches to failure detection often use state-space models [Villemeur, 1992]. These require specifications of normal, broken, and degraded states and of transitions between states. In [Raz and Shaw, 2000] we noted such models are difficult to work with when the specifications are inaccurate and suggested an alternative incremental-improvement model. CUES’ approach to anomaly detection follows this model. Using the precise expectations as specification proxies overcomes the limitation of requiring precise definitions of states and transitions.

Our long term vision is to increase the dependability of information systems through self healing [Shaw, 2000]. The work of Forrest et al. [Hofmeyr and Forrest, 2000] and IBM’s “autonomic computing” [IBM autonomic computing, 2001] has a similar vision. Forrest et al. concentrate on comparing sequences of events as the basic detection method, and have demonstrated it for the security domain (intrusion detection). The most important step in their approach is a domain specific encoding of a characteristic (peptide, in terms of the human immune system analogy they use) to monitor. IBM’s eLiza [IBM eLiza, 2001] provides self-managing servers, dealing mostly with connectivity failures.

BEAM [Mackey et al., 2001] is an end-to-end method for data analysis intended for real-time fault detection and characterization. It was originally intended for the domain of deep space probes—a domain that is mission critical. BEAM separates the two major forces of sensor data: a deterministic one, for which it uses theoretical dynamical models describing the phenomena associated with the primary function of the system and expert domain knowledge (in the form of a rule base), and a stochastic one, for which it uses a purely signal-based analysis. BEAM has a similar approach to CUES’ in its anomaly detection treatment of the stochastic data. It concentrates on a particular time-series analysis technique in its System Invariant Estimator (SIE) [Mackey, 2001]. However, it also experiments with anomaly detector fusion [Brotherton and Mackey, 2001]. The motivation for such fusion is similar to CUES' motivation in using multiple techniques for invariant inference.

4.4 The tuning stage

CUES updates the model to account for changing data behavior and user expectations. It utilizes the tool-kit techniques that the user selected in the setup stage to re-infer predicates over each new moving window of observations. Section 4.4.1 discusses the role of the moving window in tuning. The template mechanism plays an important role in tuning (Section 3.4.2). CUES always adds to the re-inferred predicates the predicates that the user marked as "accept". It gives the anomaly detector the "accept" predicates and predicates that match "update" templates. CUES applies a sanity-check heuristic to decide whether the automated re-inference resulted in a reasonable model or the user should intervene. Section 4.4.2 introduces the sanity-check heuristic.

The challenging problem of adjusting to changes in the data is also known as *learning in the presence of concept drift* by the machine learning research community. Section 4.4.3 summarizes related work.

4.4.1 The moving window of observations

CUES tunes predicates by re-inferring the predicates over a moving window of observations. The size of this moving window is important: it should be small enough for prompt updates yet big enough to include representative data to enable adjusting to trends. Section 4.2.2 discusses how CUES assists the user in setting the size of the moving window.

Updating predicates is a challenging task: there is a tradeoff between the desired quick adjustment to changes in the behavior of the data and the undesired adjustment to noise. Often, it is very hard or even impossible to automatically distinguish between changes in the data that are due to underlying trends and changes that are due to noise. The size of the moving window often controls this tradeoff. A larger window is likely to contain enough data to overcome occasional noise. However, it is also likely to include behavior that has since changed thus slowing the desired adjustment. A smaller window contains more recent behavior so adjustment to changes is quick.

However, noisy observations will have greater effect on the resulting predicates.

A simple and effective way to alleviate adjusting to noise in CUES is to add "accept" predicates. CUES checks these predicates on every moving window of observations, even if the techniques do not re-infer these predicates. Therefore, these predicates tend to keep the model relevant even if the data is very noisy. In our experience, it is often the case that, after examining the list of predicates that CUES infers (using the tool-kit technique), the user realizes that invariants exist and classifies some predicates as "accept" predicates. This is probably because CUES deals with user expectations about the *semantics* of the data.

4.4.2 The sanity-check heuristic

CUES has a sanity-check heuristic for determining whether the model it automatically re-infers is reasonable or the user should provide assistance. This heuristic examines the detection rate that is the result of checking the current model. If the detection rate is higher than a pre-defined percentage (the default is 30% but the user may change this; in our experience, if the model is not reasonable the detection rate tends to be extremely high (often over 50%)) then heuristic tries to use a previous model instead. If the resulting detection rate is lower than the threshold, the tuning stage continues to operate automatically on the next data subset. If the resulting detection rate is too high when using each of the previous five models in turn (again, this parameter may be changed by the user) then CUES suggests user intervention when it alerts the user about these anomalies. These anomalies may be due to problems in the model rather than problems in the data.

If the re-inference does not work well for particular data feeds (the detection rate is consistently too high) this may be an indication that the data has characteristics that CUES cannot handle. Likely characteristics are a time lag that is changing or modes in the data. An example of such data is the river gauges data (Section 7.2.1).

4.4.3 Related work

Changing data behavior is termed concept drift by the machine learning community (e.g., [Schlimmer and Granger, 1986]). The task of tuning to account for concept drift is related to incremental or on-line concept learning.

Usually, the problem of learning in the presence of concept drift is phrased as a classification problem in a feedback system: the learner gets a stream of objects, one at a time. After classifying each object it gets the correct classification. The learner uses the current knowledge to classify each incoming object. Examples of approaches to solving this problem are STAGGER [Schlimmer and Granger, 1986], FLORA [Widmer and Kubat, 1996], and CVFDT [Hulten et al., 2001a]. Dealing with concept drift is often part of the semantics of the learning algorithm: the algorithm refines the concept either by making it more specific or by making it more general. CUES does not get classification information—it does not know whether an observation is anomalous or normal. Further, it may not be feasible or even possible to supply this information. CUES utilizes existing un-supervised learning techniques and program analysis techniques. CUES could incorporate techniques that can handle concept drift, possibly skipping the tuning stage when applying such techniques. The techniques that are currently in CUES' technique tool kit do not handle concept drift.

Research dealing with concept drift often couples the problem with hidden contexts (e.g., [Widmer and Kubat, 1996]): changes in the context that are not part of the input data may induce changes in the target concept. Approaches to dealing with concept drift and hidden context often keep only a window of current examples (or use a forgetting factor) and use changes in the detection rate to suggest context switches. CUES utilizes similar methods.

Adapting to changes in data streams is the task of adaptive continuous queries (e.g., [Avnur and Hellerstein, 2000, Madden et al., 2002]; Section 1.4 briefly discusses continuous query systems). Adaptive continuous queries deal with query optimization when conditions change while a query is running. CUES deals with keeping predicates relevant when data behavior changes, not with optimizing the predicate inference in the presence of such changing behavior.

Chapter 5

Case studies

Three case studies with real-world data feeds provide empirical evidence in support of the usefulness of CUES. Each case study concentrates on one of the stages of CUES and utilizes the relevant mechanisms (Chapter 3 describes the mechanisms underlying CUES as well as how CUES utilizes these mechanisms in its three stages). Section 5.1 presents the truck WIM (Weigh-In-Motion) case study [Raz et al., 2004a, Raz et al., 2004b, Raz et al., 2003]. That case study concentrated on the setup stage of CUES. Section 5.2 presents the stock quotes case study [Raz et al., 2002]. That case study concentrated on the checking stage of CUES. Section 5.3 presents the stock daily low case study. That case study concentrated on the tuning stage of CUES.

5.1 The truck WIM case study

In the truck WIM case study, a domain expert interacted with the template mechanism to create a model of proper behavior for the WIM data feed from her informal expectations. Section 5.1.1 states the hypothesis the WIM case study explores. Section 5.1.2 describes the WIM data feed. Section 5.1.3 presents possible faults in the WIM data feed, concentrating on the user's expectations for the data feed behavior.

CUES successfully turned the user's vague expectations into precise predicates. Section 5.1.4 describes the methodology of the case study and Section 5.1.5 details the precise predicates and the detection rate that results from checking these predicates to detect anomalies. We also compared the resulting model to existing documentation of the data feed as Section 5.1.7 details. We show that the template mechanism is effective, as measured by the detection and misclassification rates and the insights the user gains. Section 5.1.6 summarizes the misclassification rate and Section 5.1.8

summarizes the user insights. CUES enabled quick detection of problems that had taken the data providers months to detect independently. Section 5.1.9 discusses this. The tool kit techniques that the user selected for this case study were Rectmix and Percentile. Section 5.1.10 compares the two techniques. Finally, Section 5.1.11 summarizes the case study results.

5.1.1 Case study hypothesis

The case study explores the hypothesis that the template mechanism is effective in eliciting precise user expectations and that the resulting precise expectations are a "good enough" engineering approximation to missing specifications, for the purpose of semantic anomaly detection.

The case study supports the hypothesis by showing that

1. The precise expectations are useful in detecting semantic anomalies in the WIM data, and
2. The user gains 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 data feeds, but it also helps users understand the consequences of their model selection and it draws their attention to interesting data behavior.

5.1.2 Weigh-In-Motion data

The data we use in our case study 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 Mn/ROAD research division operates a two-lane mainline test road equipped with a weigh-in-motion (WIM) scale [Minnesota Department of transportation, 2002]. The WIM scale is embedded in the pavement to observe passing vehicles. The WIM scale measures the individual axle spacings and weights, speed, length, and gross weight for each vehicle that passes over the scale. In addition, the time when the vehicle crosses the scale, the lane in which the vehicle was traveling, and any error codes generated by the WIM scale are recorded. Software then calculates a classification number and the equivalent standard axle loads (ESALs) for each vehicle

[American Association of State Highway and Transportation Officials, 1986]. ESALs are a measure of load, a dimensionless quantity that describes the usage of a pavement surface; an ESAL value of 1.0 is a standard truck. Personal vehicles are recorded by the WIM scale but are not imported into its database.

Roughly one million vehicles are added to the data set per year. The number of observations the system collects varies by vehicle type. For example, it takes the system roughly two days to collect two thousand observations of the most common vehicle type (type 9) but roughly three weeks to collect the same number of observations for vehicles of type 4 or of type 6.

CUES treats the Mn/ROAD WIM data feed as a time-stamped sequence of observations. Each observation has attribute values for a single truck: date and time (accurate to the millisecond), vehicle type (one of ten classes), lane (one of two classes), speed (mph), error code (one of twenty five classes), length (feet), ESAL, number of axles, and weight (kilo-pounds).

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.

WIM data has been used extensively for analysis of transportation design issues. This includes Mn/ROAD research projects, such as pavement performance, preventive maintenance, and low volume road design [Minnesota Road Research Section, 2003]. It also includes research projects and analyses at other states [Beshears et al., 1998, Najafi and Blackadar, 1998, Lee and Souny-Slitine, 1998, Clayton et al., 2002]. There is also research examining additional applications of WIM data, for example, real time enforcement of truck weight restrictions [Andrle et al., 2002].

However, little work addresses data quality issues. Assessing data quality is complementary to analyzing the data. Vacuum [Buchheit, 2002, Buchheit et al., 2002, Buchheit et al., 2003] was applied to the Mn/ROAD data for assessing the data quality and cleaning the data. However, that analysis concentrated on aggregated data (e.g., the daily sum of ESALs). Our analysis concentrates on individual observations and is complementary to the analysis of aggregated data. CUES can support various existing techniques. The kind of anomalies CUES detects are vehicles that do not seem to belong to their assigned class and vehicles that have attribute values that are improbable (e.g., too low or too high). Such anomalies may explain anomalies in the aggregated data.

5.1.3 Possible data faults and user expectations

The WIM data feed is produced by an event-based monitoring system. The sensor data is further processed by multiple algorithms, written by different contractors. Many things can go wrong in this process. For example, there may be problems in the physical calibration of the WIM scale, inaccurate sensing, improper processing done by software, or undesirable interactions among the multiple algorithms processing the data.

These problems may cause the WIM data feed to behave in a way that violates the user's expectations. These expectations can be summarized as: (1) vehicles in the same class should be similar and (2) vehicles should be physically probable.

5.1.4 Methodology

The user begins by looking for clusters and selecting attributes, as Section 4.2.1 describes. The user chose to cluster the data by vehicle type, so the template mechanism interacts with the user for each vehicle type separately. The user chose a subset of the attributes, so the template mechanism inputs only the attributes the user selected 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, together with the user, 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). The existing documentation defines these types as follows: type 9 vehicles are five-axle single trailer trucks, type 6 vehicles are three-axle single unit trucks, and type 4 vehicles are buses. On the basis of preliminary analysis, the vehicle types seem similar enough that CUES can use the same techniques over them. The user first creates a model, including selecting techniques and setting parameters, for two of the types (4 and 6). The user then selects predicates for the third type (9) using the techniques and parameters the user chose for the first two types. This results in detecting anomalies the user cares about, supporting the preliminary analysis regarding the similarity of the vehicle types with respect to the tool-kit techniques.

The user in the truck WIM case study is a domain expert. This case study concentrate on CUES' setup stage. Therefore, the template mechanism allows the user to select any classification during setup ("accept", "update", or "reject") but only accept or reject predicates on the last setup iteration (CUES does not update predicates in this case study).

The anomaly detector checks the model the expert has set up. The anomaly detector runs over subsets of the data. Each data subset has two thousand consecutive (sorted by time) observations.

To analyze the model, we determine, together with the expert, the resulting detection rate and misclassification rate. Section 4.3.2 defines and discusses these measures. The detection rate quantifies the number of 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 the case study, we work with the expert to analyze 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.

To help in inspecting the resulting anomalies, the anomaly detector orders the reports according to the measure a technique is using. It can then output either all the anomalies (and the user can examine as many as the user chooses) or only the most significant anomalies. In the case study, the anomaly detector outputs all the anomalies.

5.1.5 Detection rate

CUES detects anomalies over the WIM data feed using the model the expert has set up. Tables 5.1, 5.2, and 5.3 list the Rectmix model the expert has set up (predicates inferred by Rectmix) for vehicle types 4, 6, and 9, respectively. Tables 5.4, 5.5, and 5.6 list the Percentile model the expert has set up (predicates inferred by Percentile) for vehicle types 4, 6, and 9, respectively. These models consist of "update" and "accept" predicates from the final setup stage. For example, for vehicle type 6, Table 5.2 consists of the "update" predicates from Table 3.5—the final setup classification for Rectmix predicates (Figure 3.4 depicts the corresponding rectangles) and Table 5.5 consists of the "update" predicates from Table 3.6—the final setup classification for Percentile predicates.

CUES checks the model to detect semantic anomalies and computes the resulting detection rate. We present plots for one vehicle type—type 6. The plots for type 4

Rectangle	Length ∧	ESAL∧	Axles ∧	Weight
1	32–43	0–.42	2–2	11–22
2	32–45	.1–1.2	2–2	18–29
3	31–49	0–1	3–4	21–43

Table 5.1: Rectmix predicates the expert chose for type 4

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 5.2: Rectmix predicates the expert chose for type 6

and type 9 vehicles are similar except as indicated in the analysis that follows.

Figure 5.1 depicts a count of anomalous attributes flagged by the Rectmix predicates the expert chose for vehicle type 6. Similarly, Figure 5.2 depicts a count of anomalous attributes flagged by the Percentile predicates for vehicle type 6. Data subsets are time ordered; each has two thousand observations. The y-axis in the plots of Figure 5.1 and Figure 5.2 gives the total number of anomalies in one of the subsets, according to the criterion the plot specifies, e.g., length anomalies. Notice that the y-axis scale differs among plots. The x-axis is the sequential subset index.

Figure 5.1's left-most plot summarizes the total number of anomalous attributes, out of eight thousand attribute observations (four attributes times two thousand observations for each). The other plots show the break-down of this total by attribute, out of two thousand observations.

The first column in Figure 5.2 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 smaller or larger, respectively, than the range bounds. All are out of two thousand observations.

Table 5.7 summarizes the average detection rate over the subsets of each vehicle type. It gives the detection rate over all (eight thousand observations of) attributes and a break-down by attribute (out of two thousand observations).

It is interesting to notice that the predicates 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. To

Rectangle	Length ∧	ESAL ∧	Weight
1	50–78	.1–2.2	37–77
2	51–77	0–.1	11–34
3	50–77	0–.2	30–41
4	52–78	2.4–6.3	74–101

Table 5.3: Rectmix predicates the expert chose for type 9 (Axles is always 5)

Predicate
$45 \leq$speed≤ 85
$23 \leq$length≤ 52
$2 \leq$axles≤ 3
$13 \leq$weight≤ 40

Table 5.4: Percentile predicates the expert chose for vehicle type 4

unify the models, possibly also with existing specifications, the user needs to invest additional effort. For example, the user should consolidate differences among predicates and decide when an inferred model describes the behavior better than existing specifications and vice versa.

The detection rate for type 9 vehicles (plots not shown) 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 weight sensor problems in the scale or calibration problems on specific dates. Type 9 is by far the most common, so the expert thought the scale and software may be calibrated to best recognize this type.

Figure 5.2 depicts to a correlation between low speed (speed < 40 mph) and over-length (length > 39 feet)—the plots for these attributes have a similar shape. This helps us to better understand how the length estimation works. The length is estimated from the time that passes between axles, assuming high-way speed. Therefore, if the speed is very low, the length will be over estimated.

Looking at the anomalies for axles in Figure 5.2, it appears there was a change in the WIM system starting with subset number 54 (observations starting November 1999). The number of axles is very noisy in earlier observations and very clean in

Predicate
40≤speed≤88
17≤length≤39
3≤axles≤3
12≤weight≤49

Table 5.5: Percentile predicates the expert chose for vehicle type 6

Predicate
39≤speed≤85
42≤length≤79
5≤axles≤5
16≤weight≤94

Table 5.6: Percentile predicates the expert chose for vehicle type 9

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. The expert was surprised to see this behavior. The expert was also surprised to learn that a large number of vehicles with one axle exist in the data (during the early data collection period)[1]; all trucks should have at least two axles, and the filtering algorithm should have detected such anomalies.

Both Figure 5.1 and Figure 5.2 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 feet. The expert was unaware of the large number of exceptionally long and slow 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 (Table 5.7) cannot be compared between Rectmix and Percentile because the attributes are not all the same. In addition, these two techniques describe different behavior, as Section 5.1.10 that follows discusses. However,

[1]For the purpose of inferring predicates that characterize the normal behavior of the WIM system, it could be better to ignore the early data-collection period. However, the expert only learned about the system's poor behavior during that period by looking at the predicates that CUES inferred and the anomalies that CUES detected.

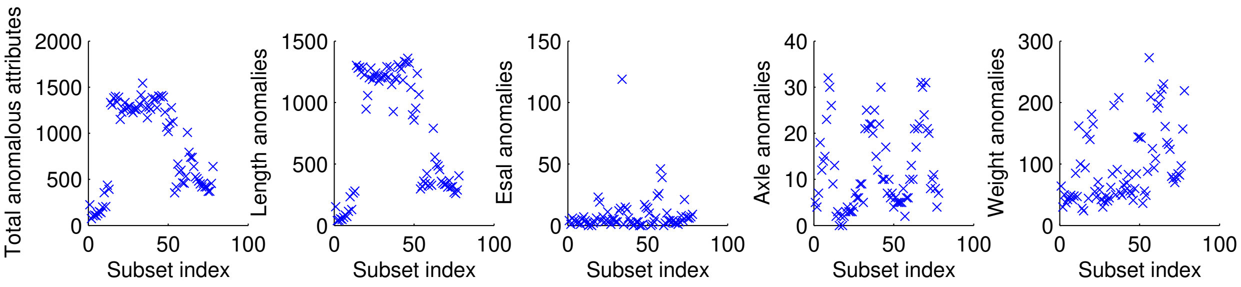

Figure 5.1: Counts of anomalies detected using Rectmix predicates for vehicle type 6

	Vehicle type	Average detection rate (%) Total	Length	ESAL	Speed	Axles	Weight
Rectmix	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 5.7: Average detection rate

it is interesting to compare the detection rates for the identical attributes (length, axle, weight). Understanding differences contributes to a better understanding of the models.

The axles anomaly detection rate is very different between Rectmix and Percentile because the predicates the expert chose differ. However, predicates resulting from both techniques may be useful (because the techniques characterize different behavior).

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

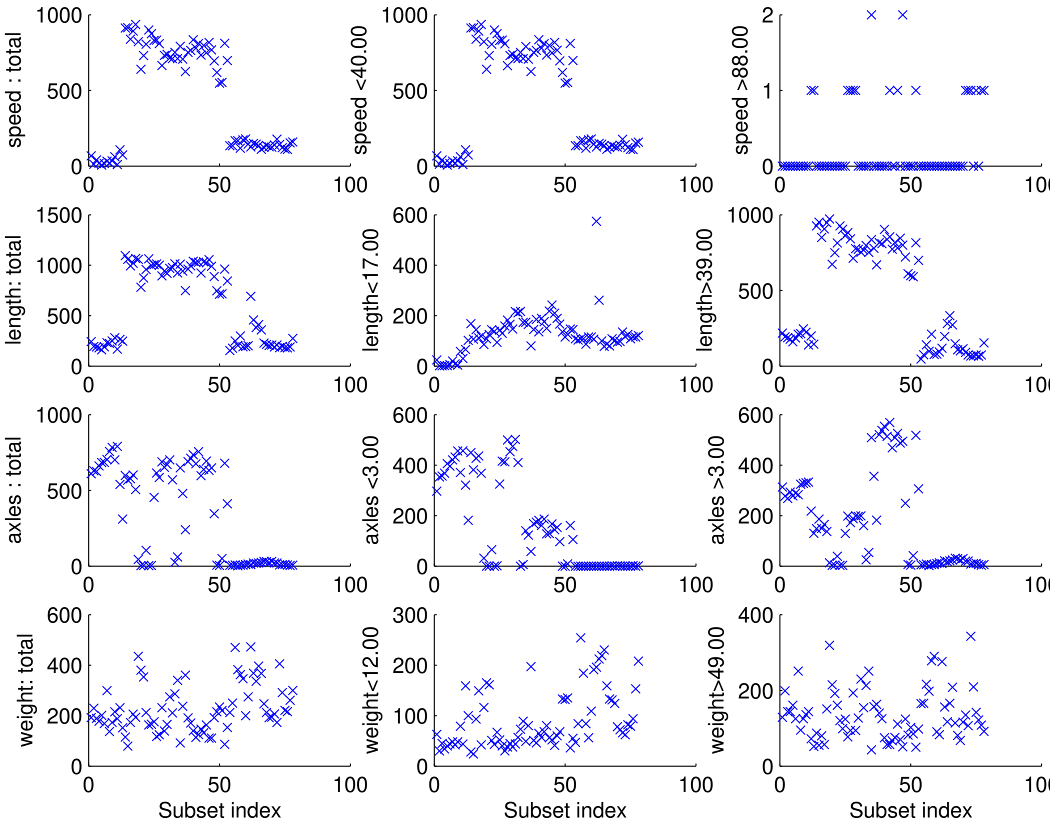

Figure 5.2: Counts of anomalies detected using Percentile predicates for vehicle type 6

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 also considers trucks with more axles.

5.1.6 Misclassification rate

The overall misclassification is $\frac{FP+FN}{Nor+Ab} = \frac{Ab+FP-TP}{Nor+Ab}$, as Section 4.3.2 defines.

Determining the above measures is subjective even though documentation for the WIM system exists. This is because, on the one hand, the documentation is sometimes incomplete and imprecise, and on the other hand, it sometimes describes

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

Table 5.8: Average overall misclassification rate

behavior that neither Rectmix nor Percentile can express.

To determine Ab, FP, and TP, the 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 5.1.7 discusses these differences). There is often a trade-off between true and false positives. Table 5.8 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 data is the cleanest, so the techniques do best on them.

The template mechanism aids in achieving low misclassification rates because it enables the user to choose only predicates that describe behavior the user cares about and thinks is normal (the alternative is to use all the predicates a technique infers).

5.1.7 Inferred model vs. documented model

We use the WIM system documentation of vehicle types [Chalkline et al., 2001] and attribute bounds [Minnesota Department of Transportation, 2000] 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; e.g., for vehicle type 9, length $\leq$ 75 feet, for vehicles type 4 or 6, length $\leq$ 40 feet. CUES' tool-kit techniques infer predicates about lower bounds as well (e.g., Tables 5.1–5.6). The expert found the lower bounds useful. For example, very slow vehicles correlate with vehicles that are over-length.

In general, the WIM software classification and filtering 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 often fails to match the documentation. This led the 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 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. The expert chose predicates that are different from the documentation when they described vehicles the expert 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 over-estimation. The models reflect these different emphases.

5.1.8 Summary of expert insights

The major insights the expert gained from the analysis detailed above are as follows:

- The data behavior strongly suggests that there was a system wide change in the Mn/ROAD WIM system starting November 1999.
- The system (hardware and software) seems to be calibrated for the most common type of trucks and possibly also for passenger cars. This, in turn, seems to adversely affect the accuracy of vehicle identification and classification of other types.
- The interaction of the classification, estimation, and filtering algorithms seems to occasionally have undesirable effects.

5.1.9 Confirmation from providers

The data providers confirmed the expert insights and cause analysis. They were unaware of the behavior that surprised the 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.

This provides an additional confirmation about the usefulness of CUES. Moreover, it demonstrates the benefits of having automated support for making expectations precise and for detecting anomalies.

To set up the model, the expert invested less than ten hours. The anomaly detection was fully automated and quick (minutes). In comparison, it had taken the data providers several months to notice the same problems, possibly because they had not done much data analysis during the early data collection period. Analyzing the anomalies requires expert time and this time depends on the number of anomalies and their nature. However, CUES directs the attention of the expert to problems, so expert time is invested efficiently.

5.1.10 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 vehicle 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 predicates often show a correlation among attributes. For example, the first two predicates in Table 3.5 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 the user did not give the speed attribute to Rectmix (this was part of the attribute selection process).

5.1.11 Results summary

The experimental results show that: (1) The model is useful for anomaly detection. It enables detection of 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 and the system calibration. The data providers confirmed the expert insights.

Moreover, the case study results corroborate the benefits of interacting with the template mechanism to make expectations precise and of analyzing the resulting anomalies. CUES: (1) detected hardware and software problems from observed data only. It detected, for example, problems that were caused by mis-calibration, software modifications, or state changes, (2) promptly detected these problems. It had taken the providers months to notice the same problems independently, and (3) increased the understanding of existing documentation. For example, the exact cut-off point between normal and anomalous was not clear from the data though it was clear (for upper bounds) from the documentation, suggesting the documentation bounds may be too strict.

Therefore, the case study provides empirical evidence in support our hypothesis (Section 5.1.1).

5.2 The Stock quotes case study

The stock quotes case study shows that it is possible to infer useful predicates for a single data feed of numeric attributes. The facets of usefulness we concentrate on are detecting semantic anomalies that would be obvious to a human and having no more than a reasonable number of false alarms. We consider a number reasonable if it is small enough to allow a human to hand-check the alarms.

The stock quotes case study also demonstrates how detected anomalies may help us to suggest missing data feed documentation.

Section 5.2.1 describes the case study methodology. Section 5.2.2 presents the stock quotes data feeds. Section 5.2.3 summarizes the case study results and Section 5.2.4 discusses these results.

5.2.1 Methodology

We want to improve incomplete specifications. The case study obtains expectations over a training set and checks the precise expectations to detect anomalies over a disjoint validation set. The only documentation available for the stock quote data feeds that the case study uses specifies a minimum delay of 20 minutes in the values these data feeds provide. This documentation is not helpful in determining a proper behavior model for these data feeds. Therefore, the precise expectations are composed solely of inferred predicates. This provides an indication of the strength of inferred predicates for semantic anomaly detection.

The case study uses two of the tool kit techniques to infer predicates: the program invariant detection engine of Daikon, augmented to handle noise, and Mean—estimating a confidence interval for the mean of the underlying distribution. These techniques were introduced in Section 3.3.

We apply each experiment to each of three freely available, distinct, web-based, stock quote services, S_0–S_2 [S1, 2000, S2, 2000, S3, 2000]. Each S_i has multiple comparable numeric attributes, listed in Table 5.9.

The data feed is then the result of querying each service for a specific ticker symbol. We mark each data feed S_i, Q_0, where S_i, $i \in \{0, 1, 2\}$ is queried for a stock ticker symbol Q_0. The same query is issued to all services. The specific stock ticker symbol is an arbitrary choice (with respect to predicate effectiveness) for data with similar change characteristics (beta and volume). To demonstrate this, we repeat all the experiments for two additional ticker symbols: Q_1 and Q_2.

Selecting only the comparable attributes might, in general, require manual intervention, thus limiting the automation of our approach. We test the importance of attribute selection by running two different variants for each experiment. In one, *comparable-attrs*, the data includes only the comparable numeric attributes. In the other, *all-attrs*, the data includes all numeric attributes.

To summarize, we have two types of experiments, corresponding to the technique used: Daikon and Mean. Each type of experiment has two variants: comparable-attrs and all-attrs. Each type of experiment is run over data $\{S_i, Q_j\} \mid i \in \{0, 1, 2\}$, $j \in \{0, 1, 2\}$. The data is divided into disjoint training and validation sets. In each experiment, the technique is used to infer predicates over the training set (step 1). The inferred predicates form the expectation and are checked on the validation set to detect anomalies (step 2).

Step 1: inferring predicates

For validation and analysis of the experimental results of step 1, a human judges whether the inferred predicates are intrinsic or incidental. *Intrinsic predicates* are invariants: they should always hold, due to the semantics of the attributes. *Incidental predicates* happen to hold over the training data but either should not hold in general or may change over time. Predicates inferred by Mean are likely to be incidental because they include specific attribute values, and these change over time. Daikon infers both intrinsic and incidental predicates. For validation purposes, we compare the intrinsic predicates Daikon infers to what we believe it should infer. In addition, for each of Mean and Daikon, we compare all of the inferred predicates across experiments.

Step 2: checking inferred predicates

After inferring predicates from the training set these predicates are checked on the validation set: each predicate is evaluated for all observations of the validation set. We then calculate the true positives rate, the false positives rate, and the misclassification rate. Section 4.3.2 defines all these measures.

Checking intrinsic predicates results in true positives only, by definition of intrinsic. To check if there are false negatives due to incomplete intrinsic predicates, we first determine all the intrinsic predicates we expect, given the attributes and the constraints on the kinds of relations Daikon infers. We then check these predicates over all the validation sets.

Note that if an experiment is reported as having detected 100% anomalies (a TP rate of 1), it does not mean all possible anomalies are detected, just the anomalies reflected by the predicates in the model. The detection capabilities are limited by two major factors: the inductive bias of a technique and incomplete information. *Inductive bias*, in Machine Learning terminology, is the a priori assumptions regarding the identity of the target concept. Such assumptions are necessary for generalizing beyond the observed data [Mitchell, 1997]. In the experiments, we only count in abnormal anomalies that are relevant to the inductive bias of the technique used.

The other factor limiting detection capabilities is incomplete information. When checking incidental predicates it is hard, or even not possible, to decide what is a false positive or a false negative, because no oracle is available to tell us what a specific value should have been. But we are interested in the kind of problems that can be recognized and understood by a human (which we call *identifiable*), not in determining what the actual value was nor in detecting precision problems. When in doubt, we consider the report to be false. Our reported numbers are, therefore, a worst case for the specific data and technique.

Augmenting step 2 with a voting heuristic

To detect anomalies, CUES checks the precise expectations (inferred predicates) with and without a voting heuristic. The voting heuristic cross-checks data with another data feed. The goal of the voting heuristic is to decrease the number of false positives and to detect some false negatives.

Without the voting heuristic, the predicates obtained from the training set in step 1 are checked over the validation set.

With the voting heuristic, additional available data (a nearly redundant data feed) is used. The voting heuristic evaluates the predicates inferred from the training set of the tested data feed over the validation set of the tested data feed and over the corresponding observations in a redundant data feed. If a predicate does not hold in both cases, it assumes the data has indeed changed (i.e., "normal" change) and no anomaly is reported. This is designed to reduce the number of false positives. A false negative is indicated if a predicate does not hold only over the redundant data feed.

To validate the experimental results when the voting heuristic is used, we hand-check whether the voting heuristic has removed true positives or added false positives. We cannot always determine this, because the true value is often unknown. The fault tolerance community often uses some sort of voting, assuming independence of voters, to decide what a result should be when the truth is unknown [Villemeur, 1992] . Our voting heuristic follows this approach and can be viewed as multi-version comparison. However, it will work only for independent data feeds. To have more confidence in the results of the voting heuristic, the heuristic can use a larger number of data feeds for voting, assuming it is unlikely that the majority of the data feeds will have the same subset of correlated attributes.

S	Numeric attributes
0	*cur*, change, *last*, *52l*, *52h*, beta
1	*cur*, change, *52l*, *52h*, *open*, *dlow*, *dhigh*, vol
2	*cur*, change, *last*, *52l*, *52h*, beta, *open*, *dlow*, *dhigh*, vol

Table 5.9: Numeric attributes. Comparable attributes are in italic.

5.2.2 Data

We use real-world data in our experiments. We chose stock quote data feeds because these are semi-structured, no oracle is available for the value of stocks at any arbitrary moment in time, and stock quote data feeds include a number of numeric attributes, some of which are comparable.

We downloaded HTML pages that are the result of querying on-line stock quote services S_0–S_2 for a ticker symbol that is one of Q_0–Q_2 (CSCO, SUNW, TXN). Stock quotes are provided by each of these services with minimum delays of 20 minutes. This data was collected Mon–Fri, every ten minutes between 10am and 4pm, for about six weeks. Because we are interested in semantic anomalies, we ignored pages from all data feeds if any had communication problems at a specific time. We also ignored syntax/form problems by manually adjusting our parsing scripts whenever the format of the HTML page changed. Each observation results from parsing one HTML page.

Table 5.9 lists the numeric attributes of the data feeds. Each data feed has a subset of the following attributes: current value (cur), last value (last), change in value (change), highest and lowest values in 52 weeks (52h and 52l), highest and lowest daily values (dhigh and dlow), value when daily trade began (open), stock's anticipated fluctuations relative to the market fluctuations (beta), and stock volume (vol).

We use disjoint validation and training sets with equal sizes for each of the techniques. Each validation set has observations from a period of one week (about 170 observations). Each training set has data from two and a half weeks (425 observations). We use a common approach for dealing with time-changing data: a moving window [Hulten et al., 2001b]. We set the window size to three and a half weeks, where the last week in the window is the validation set and the rest is the training set. At the end of each week we replace the observations of the oldest week by the observations of the current week: so for data of six weeks we have three pairs of training and validation sets for each data feed. This is a simple way to update the expectation (step 3): re-infer the predicates over recent data.

Determining the appropriate size of a training set is a difficult task. Statistical approaches exist for simple cases. Unfortunately, they often make assumptions that do not hold for our data. In addition, they can only be applied when exact information about the statistical techniques used is available. As more theoretical results become available, CUES should incorporate them. However, because CUES uses off-the-shelf techniques, full implementation details are not always available[2]. We empirically find a good training set size for Daikon, treating it as a black box, and use the same size for Mean.

Daikon needs data with enough instances of distinct values to justify a predicate. However, the more data (larger period of time) the more likely it is to contain an anomaly, thus falsifying a valid predicate. The period of time over which the training data should be collected depends on the change characteristics of the data. We chose stocks that have large beta, implying frequent changes. We empirically found the time constant for one stock and it applied to the other stocks as well. For our data, data from half a week is sufficient for Daikon to infer a single set of predicates. Because we need to augment Daikon with noise handling capabilities, inference of several initial sets of predicates is required for creating the final set. The cost of this augmentation is more data. Training sets of two and a half weeks (425 observations) seem to suffice.

5.2.3 Results

We find that the inferred predicates (step 1) are useful in detecting anomalies in the tested data feed. Furthermore, anomalous behavior of a data feed, detected by checking the inferred predicates on unseen data (step 2), suggests plausible missing documentation or specifications of the data feed.

Step 1: inferring predicates

The predicates in comparable-attrs are a proper subset of the predicates in all-attrs for both Daikon and Mean, because the attributes in comparable-attrs are a proper subset of the attributes in all-attrs. All predicates over two attributes (Daikon) in comparable-attrs are intrinsic, because comparable-attrs includes only attributes that are meaningful to compare.

[2]Full details are available for Daikon because it is described in technical papers and distributed in source form.

S	Predicates
0	$cur \leq 52h$, $cur \geq 52l$
1,2	$cur \leq dhigh$, $cur \geq dlow$, $dhigh \leq 52h$, $dlow \geq 52l$, $dhigh \geq open$, $dlow \leq open$

Table 5.10: Intrinsic predicates inferred by Daikon

Daikon: intrinsic predicates are similar both within a ticker symbol, Q_i, and between different symbols. Table 5.10 shows these predicates.

Daikon's predicates helped us to identify intrinsic predicates. We had some relations in mind, yet after examining the inferred predicates, we realized additional relations should hold. For example, in advance of these experiments, we did not think of predicates related to the attribute open.

In the majority of the experiments (26 out of 27 for each of comparable-attrs, all-attrs), all intrinsic predicates were inferred. Often, an inferred predicate does not include equality (e.g., $<$ rather than $\leq$), because the training examples do not include equality. In one experiment (S_2,Q_2) two predicates were missing ($dhigh \leq 52h$, $dhigh \geq open$). This is due to the training data containing anomalies related to the involved attributes in at least four of the subsets used in the noise handling augmentation.

Incidental predicates over one attribute differ slightly within a single ticker symbol, indicating normal changes in data, and significantly between different symbols, as expected for different stocks. In one experiment (for each of comparable-attrs and all-attrs; S_2,Q_0) Daikon learned an anomalous value ($52l = 8$). This is balanced by intrinsic predicates that detect these observations as anomalous. In all-attrs, incidental predicates over two non-comparable attributes exist, due to the different units used for these attributes. Examples include: $cur < vol$, $dlow > change$.

Mean: all Mean predicates are incidental. These predicates always involve specific values of an attribute, for example: $42.31 \leq cur \leq 63.65$. Although such values may be inherent to an attribute, resulting in an intrinsic predicate, this was not the case for our data. We examine the mean and variance that mean estimates. These vary significantly between different attributes. For specific data feed and attribute, the values are rather similar. For specific ticker symbol and attribute across different data feeds, the values are usually similar. Cases that are not similar may suggest a difference in underlying specifications, as discussed in Section 5.2.4 below.

		Q_0			Q_1			Q_2		
		S_0	S_1	S_2	S_0	S_1	S_2	S_0	S_1	S_2
	Ab									
D	atr	0	34	52	4	1	44	8	0	0
	obs	0	14	45	4	1	44	3	0	0
M	atr	0	20	32	4	0	0	8	0	0
	obs	0	14	16	4	0	0	3	0	0
	Nor									
D	atr	2040	3026	3519	2008	3017	3477	2032	3060	3570
	obs	510	496	465	499	502	459	502	510	510
M	atr	2040	3040	3538	2008	3018	3521	2032	3060	3570
	obs	510	496	494	499	503	503	502	510	510

Table 5.11: Total number of abnormal (Ab) and normal (Nor) data in the validation sets, found by either technique. Counted by attributes (atr) and by distinct observations (obs), in the comparable-attrs variant for Daikon (D) and for Mean (M)

Step 2: checking inferred predicates

The anomaly detector checks the predicates inferred in step 1 over the validation set. An anomaly is detected if a predicate does not hold over the validation set. The results are summarized in Table 5.11 and in Figure 5.3.

A single observation can trigger multiple warnings, because it contains multiple attributes. An observation is anomalous if it contains at least one anomalous attribute. The numbers we report in Table 5.11 are, for each comparable-attrs variant of an experiment (Daikon or Mean applied to all S_i,Q_j combinations), the total abnormal data and the total normal data, as reported by the two techniques, counted by attributes and by distinct observations.

The number of observations is the same for both comparable-attrs and all-attrs variants. The number of attributes is larger in all-attrs: 33% more for S_0, 25% more for S_1, and 30% more for S_2. The majority of identifiable anomalies involve predicates solely over comparable attributes. Only in one case is there an identifiable anomaly over a non-comparable attribute (*beta*), in three different observations (for Q_2, S_0). Some anomalies involve multiple attributes of the same observation (Table 5.11, Q_0, S_1 and S_2; Q_2, S_0: the number of abnormal attributes is larger than the number of abnormal observations). It seems that often an anomalous attribute indicates the raw data used by the data feed was somehow corrupted and therefore other attributes are anomalous as well. In addition, some attributes are calculated based on others (e.g., $52l$ is based on *dlow*) and an anomaly might propagate.

Figure 5.3 shows that Daikon and Mean detect most of the anomalies in the tested data feeds (most of the points are at the lower right corner), with TP/Ab near 1 and FP/Nor less than 0.3. The overall misclassification is always under 0.3. With the voting heuristic it is always under 0.15 and usually under 0.02, and with the voting heuristic after manually removing attributes that seem to be computed differently by different data feeds (h-; Section 5.2.4) it is always under 0.02. Although the worst cases we encountered have a lot of data points, most of these data points were misclassified due to only one or two predicates, so it was easy to deal with manually. In addition, the voting heuristic is very effective in removing false positives, provided attributes in different data feeds are independent.

5.2.4 Discussion

We compare the all-attrs and comparable-attrs variants and find that they produce similar results. We compare the two techniques we use and find it is helpful to use both. We demonstrate how detected anomalies can help us find feasible implicit specificationsor documentation. We discuss the voting heuristic, which we find to be very effective in reducing the number of false positives, but not the number of false negatives.

Whenever we indicate S_i,Q_j the experiment can be found in Figure 5.3 by using the indices i, j to locate the appropriate marker symbol.

Comparable-attrs vs. all-attrs

The same anomalies are detected in both comparable-attrs and all-attrs variants. Using all numeric attributes sometimes produces more false alarms than using comparable attributes only. This is to be expected, because the total number of attributes is larger. Still, the results are good, as summarized in Section 5.2.3. This is due to comparable-attrs predicates being a proper subset of all-attrs predicates and to the additional all-attrs predicates rarely triggering anomalies. For Daikon, non-comparability of attributes helps to explain the latter: the units are calibrated differently. Comparability of attributes is not meaningful for Mean, because it uses only a single attribute per predicate. These results are encouraging, as they indicate attribute selection is not a major issue for Daikon and for Mean. Due to the similarity of comparable-attrs and all-attrs, the following analysis is relevant to both.

Daikon vs. Mean

We found Mean useful for inferring predicates over single attributes and Daikon useful for inferring predicates over multiple attributes. Daikon predicates for single attributes contain exact values, which may result in a large false positives rate (Figure 5.3). However, this is easily solved by ignoring these few predicates. There are some anomalies that Mean does not and cannot be expected to detect. These are anomalies that stem from a relation that should hold between attributes, where the values for each attribute are reasonable (not very different from other values of the attribute). In our experiments these anomalies are discovered by the Daikon intrinsic predicates: $open \leq dhigh$, $open \geq dlow$ (data feeds: Q_0,S_2; Q_1,S_1 and S_2), and incidental predicates for $52h$ and $52l$ that accompany the intrinsic predicates: $dhigh \leq 52h$ and $dlow \geq 52l$ (Q_0,S_1). However, Mean is able to identify anomalous attributes that Daikon cannot. In the case of S_0,Q_2 (0,2 in Figure 5.3), a Daikon incidental predicate involving $52l$ detects 4 anomalous observations. The same observations are detected by Mean predicates involving *beta* and *last*. The Mean predicate involving $52l$ does not detect anomalies. Only by using both techniques do we notice that these observations include multiple anomalous attributes.

In addition, the combination of Daikon and Mean predicates makes it easier to decide what anomalies are true. Whereas Daikon intrinsic predicates are easy to understand, Mean can explain some anomalies detected by checking the Daikon incidental predicates, which otherwise seem accidental. For example, for S_0,Q_1 Daikon $52h = 129.31$ does not hold in 4 observations and for S_0,Q_2 Daikon $52l = 35.00$ does not hold in 4 observations. This might be due to normal changes. But Mean predicates involving $52h$ or $52l$ do not hold as well. Manual inspection of the data shows that these values are inconsistent: other values before, between, and after these observations have not changed. In general, when both Daikon and Mean predicates do not hold over an observation, our confidence regarding the validity of the warning increases.

Exposing implicit specifications

Even though the data feeds we test provide a similar service, and may even use the same raw data, the anomalies detected in the experiments help us expose some differences in the behavior of the data feeds, beyond the stated 20 minute delay. Such differences are a manifestation of incomplete specifications and entail feasible implicit specifications. Daikon and Mean predicates involving *dhigh*, 52*h*, *dlow*, 52*l*, detect anomalies in S_1,Q_0 and in S_2,Q_0, that help us expose such implicit specifications. S_2 immediately updates the 52*h* or 52*l* value whenever it is exceeded by the *dhigh* or *dlow* value; S_1 does not. The behavior of S_2 causes inconsistent values of 52*l* or 52*h* whenever the *dlow* or *dhigh* values are anomalous, which is detected by Mean predicates and by Daikon incidental predicates. The behavior of S_1 causes $dhigh \leq 52h$ or $dlow \geq 52l$ to break whenever *dlow* or *dhigh* are anomalous.

Voting heuristic

The voting heuristic is usually very effective in eliminating false positives for both Daikon and Mean. Moreover, it sometimes exposes additional implicit specifications. The heuristic is not effective in eliminating false negatives. However, the number of false negatives is small (according to our estimation of the number of false negatives that Section 5.2.1 describes) and all the cases in which anomalous attributes are missed include multiple anomalous attributes for a single observation. At least one of these attributes is always detected.

The voting heuristic sometimes either adds false positives (Figure 5.3, Mean, all-attrs, S_1,Q_0 and S_2,Q_0), as a result of problems at the redundant data feed, or does not reduce the number of false positives significantly (S_0,Q_0), as a result of the data feeds being dependent. Using more data feeds for voting should alleviate these problems, as we hope only a minority of data feeds are dependent with respect to the same attributes. Dependent data feeds can also cause the voting heuristic to eliminate true positives, as consistently happens for S_1,Q_0 and S_2,Q_0, where both data feeds have the same anomalous values for *dhigh*, *dlow*, hinting the data feeds may be using the same raw data. However, the detection is still effective because the processing of this raw data (e.g., 52*l*, 52*h*) seems to be independent (as exposed in the description of the implicit specifications), and the voting heuristic works well with independent voters.

If the voting heuristic is ineffective in reducing false positives even when a larger number of data feeds is used in voting, true differences among the data feeds may exist (implicit specifications). We found attributes that seemed to be calculated and updated differently in the different data feeds by looking at these cases (S_0,Q_0 for attributes 52*l* and *beta*; Mean, S_1,Q_0 and S_2,Q_0 for *vol*). For example, by looking at 52*l* in S_0 and in S_2 we discover additional implicit specifications of S_2: we already know 52*l*, 52*h* change immediately with *dlow*, *dhigh*. In addition, we realize that whenever the values of *dhigh* or *dlow* do not exceed those of 52*h* or 52*l*, 52*h* and

$52l$ are updated infrequently: less frequently than once in a few weeks. S_0 seems to update $52h$, $52l$ about every week. For S_1 and S_2 we notice differences in the calculation of vol, which seem to exist only for Q_0 (the values of vol are consistently and significantly different). This is an example of implicit specifications that apply only to a specific stock. Demanding that such specifications be made explicit is not reasonable.

The graphs involving $h-$ in Figure 5.3 display the results of applying the voting heuristic after manually eliminating attributes that seem to be computed differently by different data feeds. This is part of checking our former assumption regarding the heuristic exposing implicit specifications. We check whether a large number of false alarms involves attributes which we have identified as being computed differently. We see that this is indeed the case. Manually eliminating such attributes always improves the results (reduces false positives). However, to verify that this is indeed due to different computation we would need additional information about the data feeds.

5.2.5 Results summary

The case study results demonstrate it is possible to infer useful predicates over a single data feed of numeric attributes and that this can be done, to a large extent, automatically (in the context of stock market tickers). In particular, Daikon, which was originally designed for detecting program invariants from program executions, and which we augmented to handle noise, is effective in inferring useful predicates that include multiple attributes. Estimating a confidence interval for the mean is effective for inferring useful predicates that include a single attribute.

The inferred predicates were effective in discovering semantic anomalies in a data feed. Moreover, we were able to deduce implicit specifications of the data feed. The number of false positives and false negatives was reasonable for a human to handle. The voting heuristic described in Section 5.2.1 worked well in reducing the number of false positives. Nonetheless, more data feeds are needed for voting (we assume values for each attribute are independent among the majority of such data feeds). Attribute selection was not a major issue for the data and techniques we used. Moreover, except for the selection of a training set size and an additional parameter (a constant for Mean), predicate inference and anomaly detection were fully automated.

5.3 The stock daily low case study

The stock daily low case study demonstrates the ability of CUES to tune predicates: CUES successfully adjusted the model of proper behavior to changing correlations among multiple feeds. It did so by re-inferring predicates corresponding to "update" templates. Further, this case study demonstrates that CUES can handle a time lag in the data, when this lag is either fixed or small compared to the sampling interval.

Section 5.3.1 states the hypothesis the case study supports. Section 5.3.2 describes the methodology of the case study. The case study was successful in helping the user turn his imprecise and inaccurate expectations into precise expectations. Section 5.3.3 introduces the user's expectations and Section 5.3.4 lists the precise expectations. CUES utilized two of the tool-kit techniques—MUSCLES and Daikon—to form the precise expectations. Using both techniques was beneficial. Section 5.3.5 contrasts MUSCLES and Daikon. The precise user expectations were useful for sematic anomaly detection in the data feeds. Section 5.3.6 details the resulting detection rate and Section 5.3.7 details the resulting misclassification rate. Section 5.3.8 summarizes the case study results.

5.3.1 Hypothesis

The case study explores the hypothesis that CUES can elicit precise user expectations over multiple data feeds with numeric valued attributes, where the data feeds:

1. are synchronized, possible loosely—the time lags are either fixed or small compared to the sampling interval—and
2. have expected behavior that may change over time.

The case study supports the hypothesis by showing that precise expectations are useful in detecting semantic anomalies in the data feeds. We measure usefulness by the detection and misclassification rates. Moreover, the inferred predicates and the resulting anomalies pointed out differences in the behavior of the feeds and enabled the user to select a feed best fit for the user's needs.

5.3.2 Methodology

We chose data feeds from three different providers ([S1, 2000, S2, 2000, S3, 2000]). These data feeds are freely available online. These are the same data feeds of the stock quotes case study (Section 5.2), collected over the same time period. However, in that case study, CUES ran over each of the data feeds separately and examined multiple attributes (e.g., current value, 52 weeks high and low values). In the daily low case study, CUES runs over the three feeds together and examines only the daily low attribute of each feed.

We chose the daily low value of the same stock from three feeds because this data: (1) is synchronized, with a possible (small) time lag and (2) has correlations that may change over time. The specific stock ticker symbol is an arbitrary choice with respect to predicate effectiveness. Therefore, we run the case study over three different ticker symbols (CSCO (Q_0), TXN (Q_1), SUNW (Q_2)).

The user in this case study is a casual user. To select tool-kit techniques appropriate for the problem, the user started with the recommendations of the template mechanism (see Section 3.4.4 for details about the recommendation process). The template mechanism recommended using Daikon, Percentile, and MUSCLES. The user then, through interactions with the template mechanism (in the setup stage of CUES), chose to have CUES check the predicates of Daikon and MUSCLES but not Percentile. Percentile gives probable values for the daily low values of each feed. These are expected to change in ways that are very hard (or impossible) to model well. Daikon gives a correlation that should always hold between the current value of the daily low and the preceding value. MUSCLES finds a linear correlation, and also sometimes a time lag. MUSCLES and Daikon were effective in capturing the user's imprecise expectations (details follow in Section 5.3.3).

During the setup stage of CUES, the user selects predicates, and consequently techniques, by classifying the predicates inferred the techniques CUES recommends. For example, the user classified the MUSCLES predicates as "update", some Daikon predicates as "accept" and some as "reject", and all the Percentile predicates as "reject". CUES semi-automatically tunes the predicates that the user classifies as "update" (Section 3.4.2 provides details about the role of the template mechanism in the tuning stage). It does so by automatically re-inferring predicates that match "update" templates over the data subsets of the moving window. It utilizes a sanity-check heuristic to decide whether to prompt the user for guidance: if the detection rate is too high (more than 30% of the data is flagged as anomalies) then it asks the user to re-examine the model. Section 4.4 provides details about the sanity-check heuristic and about the role of the moving window of observations in the tuning stage. CUES checks predicates that match"accept" templates over all the data subsets. CUES ignores predicates that match "reject" templates.

Because the data is from multiple feeds, CUES pre-processes the data. Section 4.2.1 discusses CUES data pre-processing. First, it creates one "united" feed from the different feeds. Each observation in the united feed includes the values of the attributes of each of the feeds at time t. In the daily low case study, each observation includes the daily low value at time t of each of the three feeds: $\text{DailyLow}_1(t)$, $\text{DailyLow}_2(t)$, $\text{DailyLow}_3(t)$. Second, when a time lag in the data exists, CUES creates "mega" observations for techniques that infer stateless predicates (all the techniques except MUSCLES). A mega observation includes a spread-out time window of observations as a single observation. For example, if the window is of size one, then each mega observation includes the current daily low value (value at time t) for each of the feeds and the previous daily low value (value at time $t-1$) for each of the feeds: $\text{DailyLow}_1(t), \text{DailyLow}_2(t), \text{DailyLow}_3(t), \text{DailyLow}_1(t-1), \text{DailyLow}_2(t-1), \text{DailyLow}_3(t-1)$.

For all the MUSCLES parameters, except window size and the feed to predict, CUES uses the default that MUSCLES supplies. CUES recommends asking MUSCLES to predict the feed with the most lag. For the daily low data feeds, this results in MUSCLES inferring a correlation among all the feeds. Asking MUSCLES to predict one of the other two feeds results in a correlation between these feeds only, ignoring the feed with the most lag.

To find an appropriate size for the window of observations when there is a time lag in the data, CUES utilizes one of the many existing information criteria (BIC—Bayesian Information Criterion [Wei, 1990]). Section 4.2.2 discusses selecting a technique window size. Often, as in this case study, after reviewing the resulting predicates, the user chooses a smaller size for the window. CUES recommended a window of size 9. However, the MUSCLES predicates with window size 9 had values close to zero (or summing to 0) for all the coefficients in the time window except the coefficients corresponding to the current and previous values. This reminded the user of the stated 20 minutes delay in the values of the feeds (the data was collected so that each time tick differs by 10 minutes from the previous time tick). Therefore, the user chose to have a time window of size 1.

To select the size of the training data—the size of the moving window of observations—the user starts with the default CUES recommends. Section 4.2.2 discusses selecting a moving window size. CUES recommends using 100 observations for MUSCLES and 30 for Daikon. The user was happy with the MUSCLES default and chose to give Daikon the same number of observations for training. CUES augments Daikon with a voting heuristic to handle noise in the data (Section 3.3.4): it runs Daikon on three subsets of 33 observations each. It only outputs predicates that Daikon infers over the majority of the subsets.

5.3.3 User expectations

The user expectations for the behavior of the data feeds evolved through interactions with CUES. The user's initial expectation was that the values of the daily low for the same ticker symbol at time t should be equal. The user provided CUES with a predicate that exactly described this expectation: $\text{DailyLow}_1(t) == \text{DailyLow}_2(t) == \text{DailyLow}_3(t)$. However, after seeing both the resulting anomalies and the predicates that CUES inferred using MUSCLES and Daikon, the user realized that this expectation was not accurate. The detection rate was very high, mainly because of the user-supplied predicate. The user realized that there may be a time lag between the feeds. The feeds specify a minimal delay of 20 minutes in the values they provide. This delay may not be equal in all feeds and may change within the same feed. In addition, very small differences between the daily low values provided by different feeds are fine. This is both because the user samples the feeds and because different feeds may have different policies for updating the daily low values. The MUSCLES predicates were effective in capturing this expectation.

Another expectation was that daily low values should only stay the same or go down. The Daikon predicates, restricted to apply only to a specific day, were effective in capturing this expectation

5.3.4 Precise Expectations

Table 5.12 lists the MUSCLES predicates for one of the ticker symbols (Q_0). The predicates for the other two ticker symbols are similar.

A MUSCLES predicate is a linear regression equation. The template corresponding to the predicates of Table 5.12 is:

$$\begin{aligned}\text{DailyLow}_3(t) &= \text{Coef1} * \text{DailyLow}_1(t-1) \\ &+ \text{Coef2} * \text{DailyLow}_2(t-1) + \text{Coef3} * \text{DailyLow}_3(t-1) \\ &+ \text{Coef4} * \text{DailyLow}_1(t) + \text{Coef5} * \text{DailyLow}_2(t)\end{aligned}$$

CUES utilizes MUSCLES to re-infer fresh coefficients for each data subset because the underlying template is an "update" template. This results in tuning the dominant

Subset	Coef1	Coef2	Coef3	Coef4	Coef5
1	0.0	0.0	0.2	0.0	0.8
2	0.0	0.0	1	0.0	0.0
3	-0.7	1.3	0.0	0.1	0.3
4	0.0	-0.8	0.8	0.0	1
5	-0.7	0.9	-0.2	0.4	0.6
6	0.0	1	0.0	0.0	0.0
7	-0.4	0.4	0.1	1.8	-0.9

Table 5.12: MUSCLES predicates for Q_0. Re-inferred for each data subset

correlation among the feeds. It is often not possible to determine what behavior is the "correct" one. A changing correlation is possible because: (1) there may be a small time lag between the feeds and this time lag may vary slightly, (2) there may be small differences in the daily low values that result from the fact that the user is sampling these values, and (3) different providers may have different policies for refreshing the daily low values.

The user asked MUSCLES to predict the value at time t of Feed3. Therefore, the coefficient corresponding to Feed3 at time t is always zero. A low value of a coefficient often means low correlation between the corresponding attribute and the predicted attribute. Similarly, a high value of a coefficient often means a high correlation (or anti-correlation, when negative).

The predicates constructed from the coefficients of Table 5.12 indicate the dominating behavior of each subset. This behavior may be changing. For example, the predicate for subset 1 is: $\text{Feed3}(t) = 0.2 * \text{Feed3}(t-1) + 0.8 * \text{Feed2}(t)$. This predicate describes the behavior of the value of the daily low of Feed3 at time t as being mostly correlated with the value of the daily low of Feed2 at time t, and also correlated with its own previous value (the value of Feed3 at time $t-1$). The predicate for subset 3 is: $\text{Feed3}(t) = -0.7 * \text{Feed1}(t-1) + 1.3 * \text{Feed2}(t-1) + 0.1 * \text{Feed1}(t) + 0.3 * \text{Feed2}(t)$. This predicate describes the behavior of the value of the daily low of Feed3 at time t as being correlated with the values of Feed1 and Feed2, in both the previous observation (most) and the current observation.

Occasionally, the re-inferred predicate may describe behavior that is not helpful in detecting anomalies or is anomalous. In Table 5.12, this happens for subset 2. The corresponding predicate indicates that the value of the daily low of Feed3 at time t is identical to the previous value of the same feed. Though this is usually true, it is not helpful for anomaly detection. This is the result of MUSCLES inferring the

Predicate	Restriction
$\text{DailyLow}_1(t) \leq \text{DailyLow}_1(t-1)$	$\text{Date}_1(t) == \text{Date}_1(t-1)$
$\text{DailyLow}_2(t) \leq \text{DailyLow}_2(t-1)$	$\text{Date}_2(t) == \text{Date}_2(t-1)$
$\text{DailyLow}_3(t) \leq \text{DailyLow}_3(t-1)$	$\text{Date}_3(t) == \text{Date}_3(t-1)$

Table 5.13: Daikon predicates for all ticker symbols, restricted to hold only on a single day

predicate over a subset in which there are anomalies: the values of Feed3 often differ from the values the other two feeds. As a result, the dominant correlation is of the feed with its own previous value. Checking this predicate worked reasonably well over this subset. In general, CUES has a heuristic intended to overcomes cases in which such predicates perform poorly for anomaly detection. It checks the resulting detection rate. If it is too high, it asks the user to intervene by re-examining the predicates and their classification.

Table 5.13 lists the Daikon predicates that the user selected. The user classified these predicates as "accept" for all three ticker symbols. Therefore, CUES checks them over all the data subset.

The user enhanced the Daikon predicates with one of CUES' pre-defined restrictions. This restriction states that a predicate should only hold for attribute values of the same day. Adding this restriction only requires minimal domain knowledge (Section 3.4.3 provides more details about CUES' pre-defined restrictions).

5.3.5 MUSCLES vs. Daikon

MUSCLES and Daikon tend to complement each other. Figure 5.4 provides an example of the detection resulting from checking the MUSCLES or the Daikon predicates over one of the subsets of one of the ticker symbols (subset 1 of Q_0). This Figure demonstrates pictorially how the two techniques are often complementary: MUSCLES often finds anomalies manifesting as one of the feeds behaving differently from the other two feeds. Daikon often finds anomalies manifesting as a daily low value that is higher than a previous value in the same day. Occasionally, the two techniques find the same anomalies.

Daikon:

Muscles usually looks for anomalies that are the result of a feed behaving very differently from the other one or two feeds (depending on the coefficients of the current model). Often, these anomalies result in a big change in values in one of the feeds. Therefore, MUSCLES usually flags big changes as anomalies, except when the changes occur in the other feeds MUSCLES looks at as well. This is usually the desired behavior because values are allowed to go up between days. However, correlated failures are sometimes missed. Daikon does not find anomalies where one feed behaves very differently from the others. However, if the feed corrects such anomalous behavior by a daily low value that is higher than previous values on the same day then Daikon warns about this change (but not about the preceding anomalous values).

Daikon notices very smally fluctuations in the values of a single feed. MUSCLES does not. This is because the comparison of actual values to predicates differs between the techniques. MUSCLES allows a difference that is proportional to the standard deviation of the predicted feed. This is because MUSCLES looks for statistical linear correlations. Daikon compares values of "accept" predicates (invariants) exactly.

5.3.6 Detection rate

Table 5.14 details the detection rates resulting from checking the user's precise expectations (the predicates the user selected). Section 4.3.2 defines detection rate. The detection rate is the number of observations flagged as anomalous out of the total number of observations (100 observations in a subset times the number of subsets checked). The detection rate is listed for all three stock ticker symbols, detailed for each data subset and averaged over all the subsets of a single ticker symbol.

Figure 5.4 depicts the anomalies detected over one of the data subsets for one of the ticker symbols.

Table 5.15 lists the number of true anomalies for all three stock ticker symbols, detailed for each data subset and averaged over all the subsets of a single ticker symbol. We determined true anomalies using the rules that we set together with the user for calculating the misclassification rates (See Section 5.3.7).

Because MUSCLES and Daikon are usually complementary, the detection rate when using both techniques is better than when using only one of the techniques. Section 5.3.5 discusses the differences between MUSCLES and Daikon with respect to anomaly detection. The combined detection rate is the result of flagging as an anomaly an observation that is flagged as anomalous either by checking the MUSCLES predicates or by checking the Daikon predicates. The combined detection rate (the "both" entry of Table 5.14) is closer to the true anomaly rate (Table 5.15) than the detection rate of each of the techniques alone (the "MUSCLES" and "Daikon" entries

Ticker	Technique	Detection rate per subset (%)							Average (%)
Q_0	MUSCLES	14	8	6	2	1	10	2	6.1
	Daikon	3	4	7	3	3	2	1	3.3
	Both	15	9	9	5	4	11	2	7.9
Q_1	MUSCLES	0	0	13	0	9	3	–	4.2
	Daikon	4	0	9	3	8	5	–	4.8
	Both	4	0	17	3	14	5	–	7.2
Q_2	MUSCLES	5	4	8	6	9	1	–	5.5
	Daikon	0	4	13	2	1	0	–	3.3
	Both	5	7	19	6	9	1	–	7.8

Table 5.14: Detection rates: detailed for the data subsets and averaged over all subsets (out of 100 observations in each subset)

Ticker	True anomalies per subset (%)							Average (%)
Q_0	12	6	9	5	4	13	2	7.3
Q_1	4	0	15	3	18	7	–	7.8
Q_2	2	7	18	7	9	10	–	8.8

Table 5.15: Number of true anomalies: detailed for the data subsets and averaged over all subsets (out of 100 observations in each subset)

of Table 5.14).

5.3.7 Misclassification rate

We determine, together with our user, true positives (TP), false positives (FP), true negatives (TN), false negatives (FN), and misclassification rates. Section 4.3.2 defines these measures. A true positive is:

1. A daily low value that is higher than the previous value of the same day. The Daikon predicates (Table 5.13) define this case.

2. A daily low value that is visibly different from the values of the other feeds. Small differences are fine because they may be due to either intentional delays in refreshing values by the providers or to sampling the feeds.

Ticker	Technique	Misclassification rate (%) per subset							Average misclassification (%)
Q_0	MUSCLES	4	4	3	3	3	5	0	3.1
	Daikon	9	2	2	2	1	11	1	4
	Both	3	3	0	0	0	4	0	1.4
Q_1	MUSCLES	4	0	6	3	9	4	–	4.3
	Daikon	0	0	6	0	10	2	–	3
	Both	0	0	2	0	4	2	–	1.3
Q_2	MUSCLES	5	5	12	1	0	9	–	5.3
	Daikon	2	3	5	5	8	10	–	5.5
	Both	5	2	1	1	0	9	–	3

Table 5.16: Misclassification rate: $\frac{FP+FN}{AllData}$. Detailed for the data subsets (out of 100 observations in each subset) and averaged over all subsets.

Table 5.16 details the misclassification rates per data subset and the average misclassification rates over all subsets of a stock ticker symbol. For each ticker symbol, it lists the misclassification rates resulting from checking the MUSCLES predicates only, the Daikon predicates only, and the predicates of both techniques.

Figure 5.5 provides more details about the misclassification rates: it depicts the TP rate vs the FP rate. A lower FP rate is better (0 is best, 1 worst). A higher TP rate is better (1 is best, 0 worst). Each symbol in the plot corresponds to one of the data subsets. The top row shows the results of checking the MUSCLES predicates for each stock ticker symbol. Similarly, the middle and bottom rows show the results of checking the Daikon predicates and the predicates of both techniques, respectively.

The FP rate (y-axis in Figure 5.5) is very good: it is always less than 4% for MUSCLES, and perfect (0%) for Daikon (by definition of "accept" predicates). Checking both the MUSCLES and Daikon predicates does not affect the FP rate.

MUSCLES false positives are usually due to large changes in the daily low values between days. Muscles does well, especially considering that it does not know about days. We marked the anomalies that muscles flagged when value changed greatly at the beginning of the next day as false positives. We were interested to see whether this would result in poor performance of MUSCLES (it did not). However, this behavior could be considered anomalous with respect to the change in value. A different user may have chosen to mark the same cases as TP.

The TP rate of each technique alone is mediocre. However, because the two techniques are often complementary (as Section 5.3.5 discusses) their combined TP rate is much higher. The bottom row of Figure 5.5 shows that the TP rate over most of the subsets (different markers) moved closer to 1 (x-axis in Figure 5.5) compared to the higher rows (MUSCLES and Daikon alone). Combining MUSCLES and Daikon in the detection also compensates for the FN rate of each technique alone (usually very low but occasionally as high as 10%). Usually, anomalies that are overlooked by one technique are detected by the other so the combined FN rate is very good (usually 0, and always less than 4% except for one subset for which the FN rate is 9%). The result is a joint misclassification rate that is much better than the misclassification rate of each technique alone. Table 5.16 shows that the average misclassification rate drops for all stock ticker symbols when using both MUSCLES and Daikon.

MUSCLES false negatives are usually due to minor fluctuations in the values of a single feed. These are detected by the Daikon predicates. Only one of the feeds (Feed2) exhibits this behavior, once for Q_1 and three times for Q_2. These fluctuations last for several hours. An example is daily low values that alternate between 41.62 and 41.63. Though such small differences in values are probably insignificant for most casual usages, this behavior may indicate some instability in refreshing the daily high values. If such a problem re-occurs (as happens in Feed2), the user may prefer to rely on a data feed that does not exhibit this behavior.

A small number of the MUSCLES false negatives are due to MUSCLES overlooking a value that is very different from the other values. This is an example of the inherent trade-off between FP and FN. MUSCLES uses a measure that is proportional to the standard deviation to compare actual values to predicted values. In general, it is not possible to to find a value (e.g., of the number of standard deviations to allow) that will always perform well. A smaller value will result in a more sensitive detection and therefore lower FN rate, but may be too sensitive, resulting in a higher FP rate. Similarly, a higher number will result in a higher FN rate but also a lower FP rate.

Daikon false negatives are due to differences in the values of the feeds: the daily low values that one of the feeds gives are very different from the values the other feeds give. An example is the first few observations in Figure 5.4.

5.3.8 Results summary

The case study results provide empirical evidence in support of the case study hypothesis: the user expectations, made precise through CUES, were useful in detecting semantic anomalies in multiple feeds providing stock daily low values. Though the detection and misclassification rates of either MUSCLES or Daikon alone were mediocre, checking the predicates of both techniques greatly improved these rates: the detection rate increased and was close to the actual anomaly rate (see Section 5.3.6 for details), and the misclassification rate decreased by close to 50% and was less than 1.5% for two of the stock ticker symbols and 3% for the third (Section 5.3.7). This was because the two techniques were complementary (Section 5.3.5).

Moreover, the resulting anomaly detection helped the user in deciding which of the feeds to rely on. For the time period of the stock daily low case study, it seemed that Feed1 performed best: Feed3 had more anomalies than the other two feeds (Table 5.15), and Feed2 occasionally suffered from an instability in the daily low values (Section 5.3.7).

CUES' tuning stage successfully adapted the model of proper behavior to altering correlations among the feeds. It did so by re-inferring predicates corresponding to "update" templates. CUES successfully inferred predicates over multiple time feeds that were loosely synchronized: there was a time lag between the feeds and this lag was small compared to the sampling interval. Further, CUES inferred stateful predicates—predicates over a window of observations: MUSCLES has a built-in ability to handle a window of observations. CUES augments other tool-kit techniques that infer stateless predicates with the ability to handle a window of observations. It does so by pre-processing.

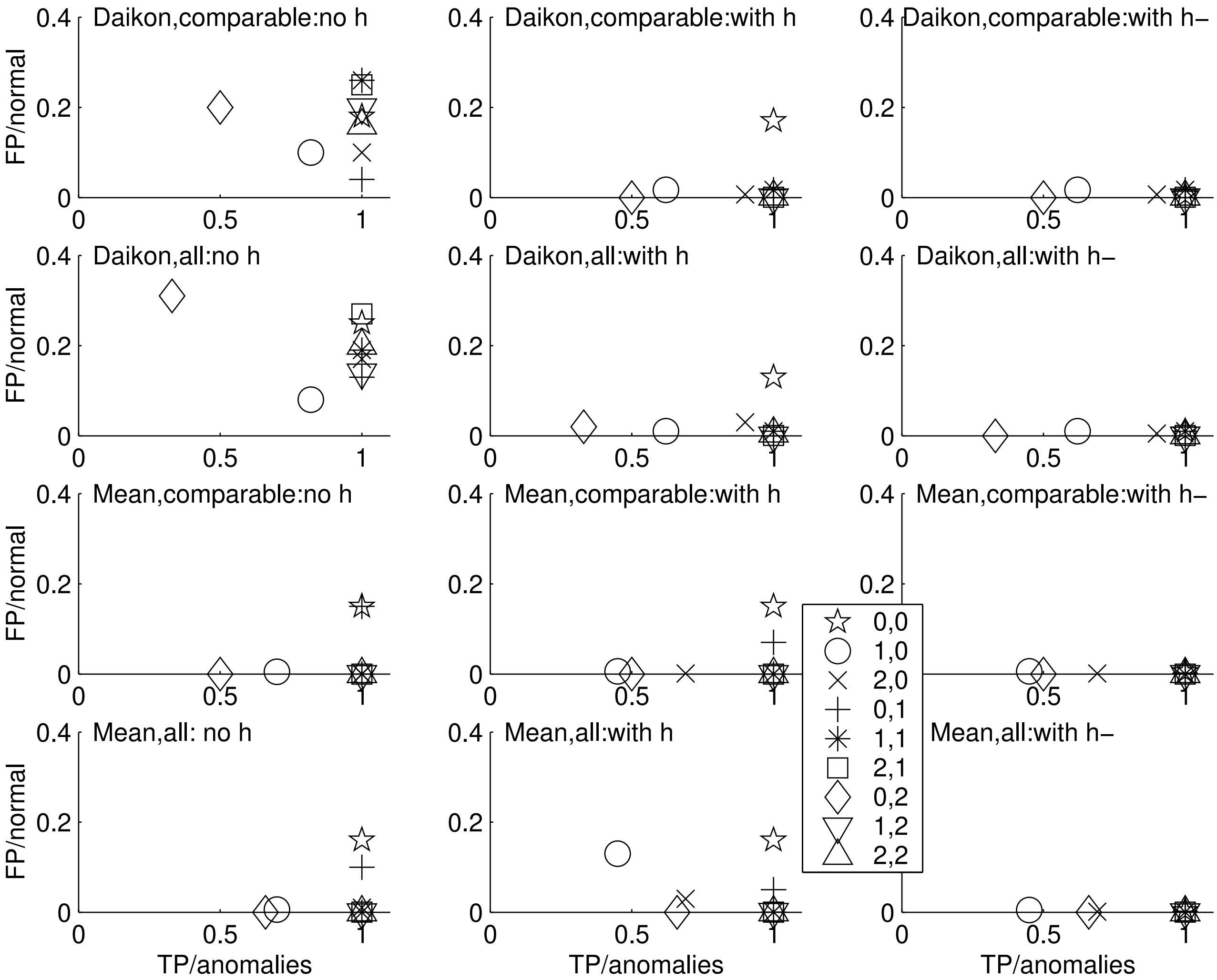

Figure 5.3: TP (True Positives) ratio vs. FP (False Positives) ratio. Best possible result: $(TP/anomalies, FP/normal) = (1, 0)$. *Each data feed is indicated by a single marker denoted in the legend by i,j, corresponding to* S_i,Q_j. *Rows 1–2 depict results for Daikon, rows 3–4 for Mean. For each of Daikon/Mean, first row is the comparable-attrs variant, second row is all-attrs. First column is results without heuristic (h); second with h; third with h, after manually removing attributes that seem to be computed differently by the different data sources (h-).*

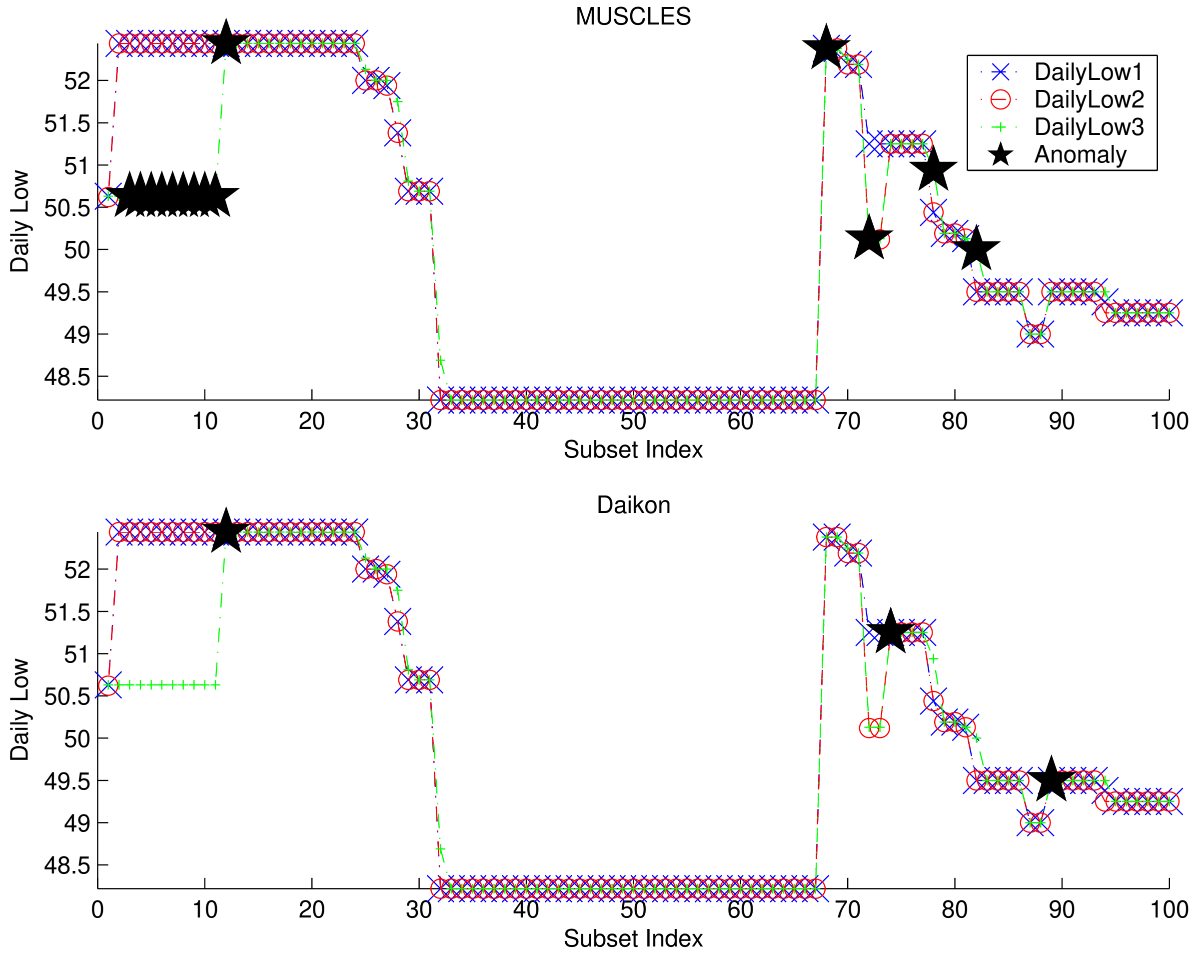

Figure 5.4: An example of detecting anomalies using the MUSCLES predicates (top) and the Daikon predicates (bottom)

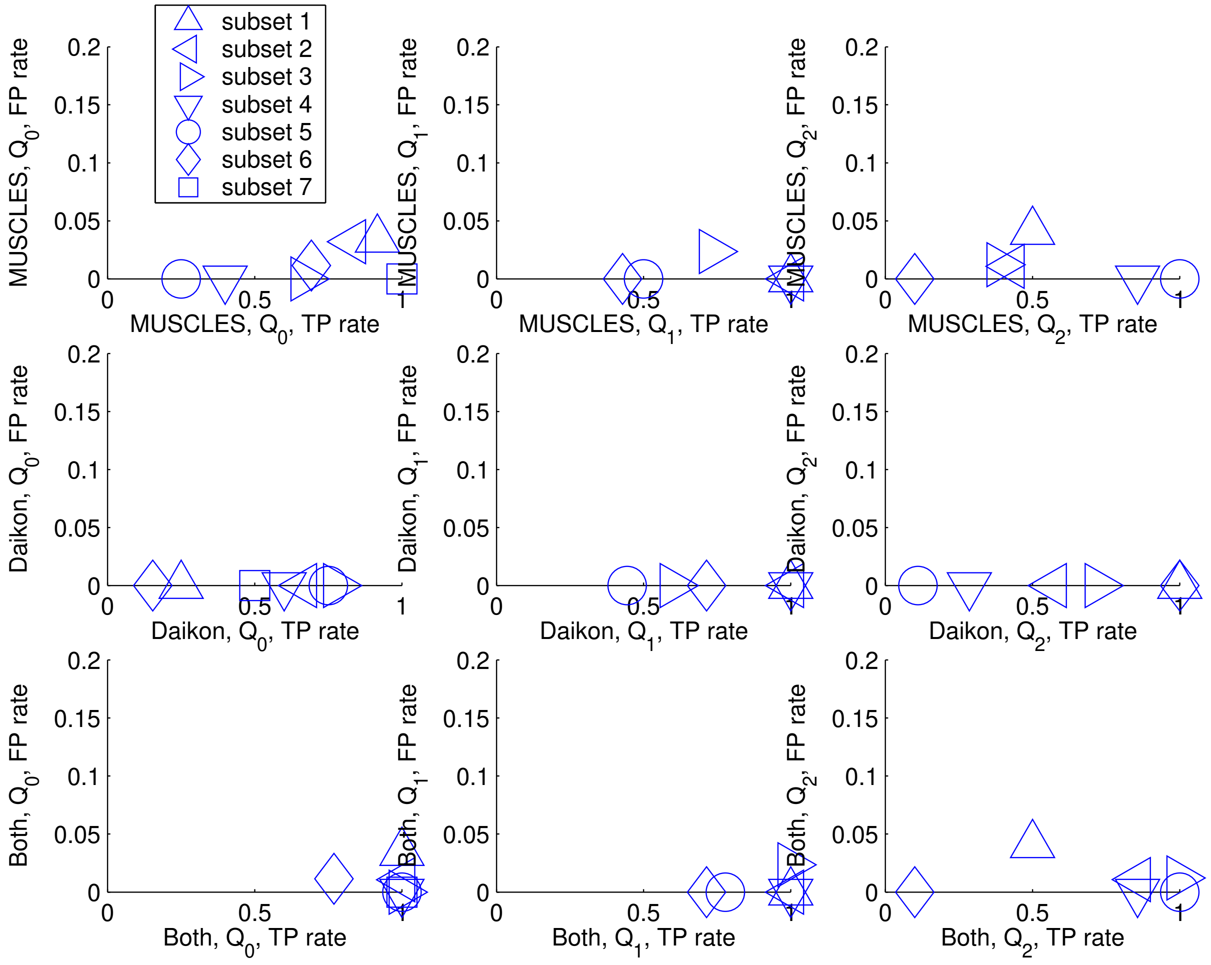

Figure 5.5: TP rate vs. FP rate for MUSCLES, Daikon, and when using both techniques. Each ticker symbol corresponds to one of the data subsets. The FP rate is always good (less than 4% for MUSCLES and always 0 for Daikon, by definition). The TP rate of MUSCLES and Daikon separately is mediocre (detecting more than 50% of the anomalies for most subsets). However, checking the predicates of both techniques and flagging as an anomaly observations flagged by either one of the techniques greatly improves the TP rate for most subsets.

Chapter 6

Validation

We provide evidence in support of the claims of this research. The thesis claims (Section 1.1) are: "this research contributes to the engineering of information systems that incorporate data feeds by providing CUES. CUES is a *practical* method for helping users state precisely their own expectations about the behavior of data feeds. CUES is a *practical* means of exploiting the users' descriptions of expectations for *semi-automated* semantic anomaly detection".

To establish these claims we show that:

- CUES is practical. By practical we mean that CUES is technically feasible and that it shows promise for being useful to users. This research demonstrates the feasibility of the technical foundations of CUES. Our results indicate that it would be worth while to invest effort in designing and implementing both an effective user interface and a convenient means of incorporating CUES-based checking during system composition (Section 7.2.1 provides details about some of the components of possible user interfaces).

  Three case studies and a validation study, all with real-world data feeds, provide evidence in support of the technical feasibility of CUES and of the usefulness of CUES to potential users. Section 6.1 summarizes evidence from the case studies (Chapter 5) demonstrating the feasibility of each of the three stages of CUES and supporting the usefulness of CUES. Section 6.2 presents the validation study. The validation study demonstrates the end-to-end technical completeness and feasibility of CUES. It also demonstrates the practicality of CUES to potential users. Two reviewers who had not previously utilized CUES were successful in making expectations precise for real-world weather-conditions data feeds. These precise expectations, repeatedly updated by the CUES tuning stage, were useful in checking the data feeds. The reviewers evaluated the usefulness of CUES to a potential user by assessing the effort the user would need to invest and the benefit the user would gain. Both reviewers concluded that non-expert users can reasonably be expected to invest the necessary effort and that the benefits are substantial and immediate.

- CUES is cost-effective. We measure cost qualitatively both by computation load and by load on the user. We measure effectiveness and usefulness both qualitatively, by the insights the user gains, and quantitatively, by the detection and misclassification rates. Section 6.3 summarizes the cost and the effectiveness of CUES. The cost of utilizing CUES is reasonable and the potential benefits substantial.

- CUES can contribute to increasing the dependability of everyday information systems. Section 6.3 also discusses how this research contributes to this long-term goal.

6.1 Case studies

Three case studies with real-world data demonstrate the feasibility of the three stages of CUES and provide evidence supporting the usefulness of CUES.

The truck WIM case study (Section 5.1) demonstrates the feasibility of CUES' setup stage, utilizing CUES' novel template mechanism. The truck WIM case study provides evidence supporting the effectiveness of the template mechanism for eliciting precise user expectations. It demonstrates that the resulting precise expectations are a "good enough" engineering approximation to missing specifications, for the purpose of semantic anomaly detection. In that case study a domain expert interacted with CUES to make her expectations precise. The case study results show that:

- The resulting model is useful for anomaly detection. It enables detection of actual anomalies that the expert cares about: classification problems and unlikely vehicles. In addition, the misclassification rate is reasonable for a human to handle (usually less than 3%).

- 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 and the system calibration. The data providers confirmed the expert insights.

- CUES detected hardware and software problems from observed data only. It detected, for example, problems that were caused by mis-calibration, software modifications, or state changes.
- CUES promptly detected these problems. It had taken the providers months to notice the same problems independently.
- CUES helped to increase the understanding of existing documentation. For example, the exact cut-off point between normal and anomalous was not clear from the data though it was clear (for upper bounds) from the documentation, suggesting the documentation bounds may be too strict.

The stock quotes case study (Section 5.2) demonstrates the feasibility of CUES' checking stage. The stock quotes case study results show that

- It is possible to infer useful predicates for a single data feed of numeric attributes.
- The predicate inference can be done, to a large extent, automatically. Checking is fully automated.
- The predicates are effective in discovering semantic anomalies in the data feed. The misclassification rate is reasonable for a human to handle (usually less than 2% with a voting heuristic).
- The anomalies these predicates detect help us to deduce implicit specifications of the data feed—we were able to suggest missing data feed documentation.

The stock daily low case study (Section 5.3) demonstrates the feasibility of CUES' tuning stage. CUES successfully adjusted the model of proper behavior to changing correlations among multiple feeds. It did so by re-inferring predicates corresponding to "update" templates. The case study results show that

- The resulting model is useful in detecting semantic anomalies in multiple feeds providing stock daily low values. The detection rate is close to the actual anomaly rate. The misclassification rate is reasonable for a human to handle (usually less than 3%).
- The resulting anomaly detection helps the user in deciding which of the feeds to rely on.
- CUES' tuning stage successfully adapts the model to altering correlations among the feeds.

- CUES successfully infers predicates over multiple time feeds that are loosely synchronized: there is a time lag between the feeds and this lag was small compared to the sampling interval.

- CUES infers stateful predicates—predicates over a window of observations. MUSCLES has a built-in ability to handle a window of observations. CUES augments other tool-kit techniques that infer stateless predicates with the ability to handle a window of observations. It does so by pre-processing.

6.2 Validation study

A validation study provides evidence in support of both the end-to-end technical feasibility (completeness) of CUES and the practicality of CUES to potential users. Section 6.2.1 introduces the setup of the validation study. Sections 6.2.2 and 6.2.3 present the reviews of reviewer A and reviewer B, respectively. These reviews and Section 6.2.4 that discusses the sanity-check heuristic of the tuning stage demostrate the technical completeness of CUES, as well as its practicality to potential users. Section 6.2.5 analyzes and summarizes the results of the validation study.

6.2.1 Setup

The study involved new data feeds and two reviewers who had not previously used CUES. The reviewers are colleagues—software engineering researchers. CUES did not change in preparation for nor during the validation study. We summarize the goals of the validation study, describe the data feeds it uses, introduce the usage scenario that we provided to the reviewers, and summarize the methodology of the validation study.

The validation study follows recommendations of usability engineering principles for designing and implementing a product [Nielsen, 1992]. CUES is a prototype of the detection part of a bigger system (Chapter 7). Therefore, only a subset of the usability engineering recommendations for the pre-design and design stages is relevant to the current state of CUES. Usability engineering principles in the pre-design stage aims to understand the target user population and user tasks. We define these in the motivation of our research. The target user population is non-expert (non-domain-expert and non-analysis-expert) users. The target user tasks are incorporating data feeds in everyday information systems. In addition, we define a usage scenario for the review.

Goal

The validation study demonstrates that CUES is technically feasible end-to-end and technically complete by showing that the reviewers can set up a model of proper behavior through interaction with CUES. It verifies that this model is effective for semantic anomaly detection in the data feeds, as measured by the detection and misclassification rates. It also demonstrates the technical feasibility of CUES' tuning stage.

The validation study also provides evidence in support of the usability and benefit of CUES to a potential user who is neither a domain nor analysis expert. This evidence is based on the assessment of the reviewers regarding the practicality of CUES.

Data

The data feeds in the validation study are two weather data feeds that provide local climatological data [NCDC, 2003]. One reports hourly sensor data from Pittsburgh International Airport (PIT). The other reports hourly sensor data from Allegheny County Airport (AGC). The distance between these two locations is less than 20 miles. Each data feed has the following attributes: dry-bulb temperature (`PITdry` for PIT, `AGCdry` for the AGC), wet-bulb temperature (`PITwet` for PIT, `AGCwet` for the AGC), and dew-point temperature (`PITdew` for PIT, `AGCdew` for the AGC). The dry-bulb is a thermometer that is dry. The wet-bulb is a regular thermometer with a wet (distilled water) muslin wick covering it, and brisk air flowing across the wick. This results in evaporative cooling. The dew-point temperature provides a measure of humidity in the air. It can be calculated from a table that uses the dry-bulb and wet-bulb temperatures.

The validation study uses an unedited version of these data feeds—a version that did not go through the complete quality assurance process of the National Climatic Data Center. It uses one year's worth of data (for the year 2003, about 8600 observations)

The data feeds are loosely synchronized. Each provides hourly observations but the time during the hour differs between the two feeds.

Usage scenario

We describe a hypothetical usage of the data feeds that Section 6.2.1 describes. A person utilizes hourly weather feeds for automated house-hold environmental control. The house is located between the PIT and AGC airports. The person expects the data feeds to be roughly similar.

Methodology

Usability engineering design-stage activities include participatory design. In a participatory design the designers have access to representative users. Participatory design recognizes that expecting users to come up with design ideas from scratch is unreasonable. However, users are very good at reacting to concrete designs and indicating what they do not like or will not work in practice. The reviewers in the validation study examine the CUES prototype with the potential users in mind. They indicate what they do not like or may not work for the task the scenario defines.

Usability engineering principles recommend prototyping during the design phase and doing some form of user testing. There are various ways to do user testing, among them observing users work on a set of representative standard tasks. In the validation study, the reviewers and the designer of CUES work together to provide qualitative information about the effectiveness of CUES for potential users in the validation scenario.

Before giving the data to CUES we unite the feeds according to the hour of the observation, ignoring the difference in minutes. This synchronization would be done in practice by the user, but here we prepared the data feeds and usage scenario for the reviewers. We also indicated the data measurement scale and expected correlation to get CUES' initial technique recommendation. Again, this would be done by the user, but here, because this is a review, we provided the data feeds, not the user. The techniques CUES recommended were Percentile, MUSCLES, and Daikon.

At the beginning of each review session the reviewer gets a reviewer's guide for the validation study (Appendix A). The guide introduces briefly CUES, the data, the usage scenario, the review goals, and the recommended tool-kit techniques.

CUES has a graphical interface for displaying anomalies to the user. This interface plots the data and highlights the anomalous observations. However, CUES does not have a graphical user interface for changing parameters. It is currently a rough prototype. Therefore, for the purpose of this study, the designer of CUES acts as an interface interacting with the user. The designer only performs actions that are reasonable for a user interface during this interaction.

Each reviewer spent up to an hour setting expectations for the behavior of the weather data feeds through interactions with CUES over a subset of the data. Each reviewer also indicated manually observations the reviewer thought were anomalous in the entire data, to enable the calculation of the misclassification rate. Each reviewer provided feedback about the practicality of CUES to potential users and about what an effective user interface should include. CUES checked the precise expectations to detect anomalies in the remaining unseen observations of the data feeds and calculated the detection rate. For predicates the reviewers marked as "update", CUES' tuning stage re-inferred the predicates over each data subset and checked that model on the next data subset. Though it was unnecessary for the tuning stage to execute its sanity-check heuristic (Section 4.4.2) for the models the reviewers selected, we experimented with alternative models that did require tuning to execute its heuristic, to ensure CUES' end-to-end technical feasibility.

6.2.2 Reviewer A

We summarize the interaction of reviewer A with CUES, the resulting model and its performance for anomaly detection in the data feeds, and the reviewer's assessment of the practicality of CUES for potential users. Reviewer A is a software engineering researcher with expertise in microeconomics, utility theory, and decision theory.

Interaction with CUES

Reviewer A immediately rejected all the Percentile predicates. The rejection was based on the description of the kind of predicates Percentile can infer and on the reviewer's basic knowledge about the characteristics of weather data (semi-random, so values that are lower or higher than most of the data values are likely to still be normal).

Reviewer A interacted with CUES to find parameter values for MUSCLES and gained a better understanding about what he believed is normal or anomalous simultaneously. The reviewer examined the resulting anomalies guided by the question "Would I want to be warned about this kind of behavior?". If the answer was no, he changed the parameter value and re-examined the resulting detection. If the answer was yes for most of the flagged observations, he selected that parameter value.

Reviewer A did not change the default CUES' moving window size (100 observations for MUSCLES) or the default feed to predict (the first attribute of the first feed—PITdry for this data). He investigated selecting attributes, changing the sensitivity of comparing actual to estimated values (the number of standard deviations to allow), and changing the MUSCLES window size.

For each combination of parameter values the reviewer ran CUES on only a few data subsets (usually less than five) and then investigated a different combination. This helped the reviewer to understand the effect of the different selections and the way the technique works.

Reviewer A started with all the attributes (six attributes), window size zero, predicting `PITdry`, and a sensitivity of five standard deviations. After a few iterations he changed the sensitivity to two and decided the resulting model was too sensitive. Seeing the anomalies resulting from sensitivity two helped him to better understand his expectations. It became clear to him that he would not want to be warned about most of the resulting anomalies.

After examining the detection over a few subsets when using all attributes the reviewer decided that he only cared about the dry-bulb temperature. He then selected two out of the six attributes: `PITdry` and `AGCdry` and interacted with CUES again. He examined two window sizes (one and zero) and four sensitivity values (the recommended values: two, five, and ten, and a value he manually selected—eight).

When using only two attributes the reviewer noticed interesting behavior he thought was anomalous. He then checked again the performance of the model when using all attributes. This interesting behavior was not detected, so the reviewer decided that indeed two attribute are better than all attributes. The resulting model is easier to understand and anomalies become evident.

When examining the results of a model with window size zero the reviewer felt that there were too many false positives. This helped him to understand that a larger window size (one) provides some continuity. Further, this continuity matches his expectations for the behavior of the data. The value at PIT should be close to the value at AGC and should probably be not very different from the previous value. The reviewer chose a less sensitive model (ten) because he felt this eliminated many false positives.

Reviewer A accepted the Daikon predicates after two iterations with CUES (in which he changed the default subset size from ten observations to twenty) because he though they indicate physical truth.

Model and its performance

Reviewer A set up a model of proper behavior for the PIT dry-bulb temperature. He did so by interacting with CUES using the first 500 observations of the data feeds. CUES then automatically checked the resulting model over the remaining unseen observations (8100 observations). We report the model performance over all the observations, to enable a better comparison of the models the two reviewers selected.

Reviewer A choose to have CUES update the MUSCLES predicate resulting from running MUSCLES on the values of two attributes–`PITdry` and `AGCdry`, predicting `PITdry`, using a window size of one, and allowing a difference of ten standard deviations between the predicated and actual values. The resulting "update" template for the MUSCLES predicate is $\texttt{PITdry}(t) = \#\texttt{PITdry}(t-1) + \#\texttt{AGCdry}(t) + \#\texttt{AGCdry}(t-1)$, where # indicates a numeric value.

Reviewer A thought that the anomalies in the data that are relevant for the purpose the scenario describes are observations of the PIT dry-bulb temperature that are very different from the AGC dry-bulb temperature or that are very different from the previous observation. Figure 6.1 shows an example of such anomalies in one of the data subsets. It shows the `PITdry` and `AGCdry` temperatures for observations in that subset. The x-axis is the sequential observation index in the window of observations. The y-axis is the temperature in Fahrenheit. A black star marks the anomalous observations. The detection rate of the MUSCLES predicate, averaged over all the data subsets (8600 observations) is 0.1 percent. The second column of Table 6.1 indicates that the total number of true anomalies, as judged by the reviewer, is 15. Therefore, the anomaly rate of the data is 0.2 percent. The difference between the detection rate and the anomaly rate is due mainly to anomalies in one of the subsets (the first subset in Table 6.1, depicted in Figure 6.1). That subset includes 10 consecutive anomalous observations. Though the MUSCLES predicate only flags two of these as anomalous, the reviewer felt this is fine because the important thing is to draw the user's attention to this behavior.

This is an example of selecting a model that results in detection that is within the "comfort zone" of the user with respect to the inherent trade-off between false positives and false negatives. The reviewer felt that it is sufficient if CUES flags some anomalies in a series of consecutive anomalies. This results in fewer false positives yet still draws the user's attention to interesting behavior.

Table 6.1 summarizes the misclassification rate of the MUSCLES predicate. It only shows data subsets in which there are either true anomalies or anomalies falsely

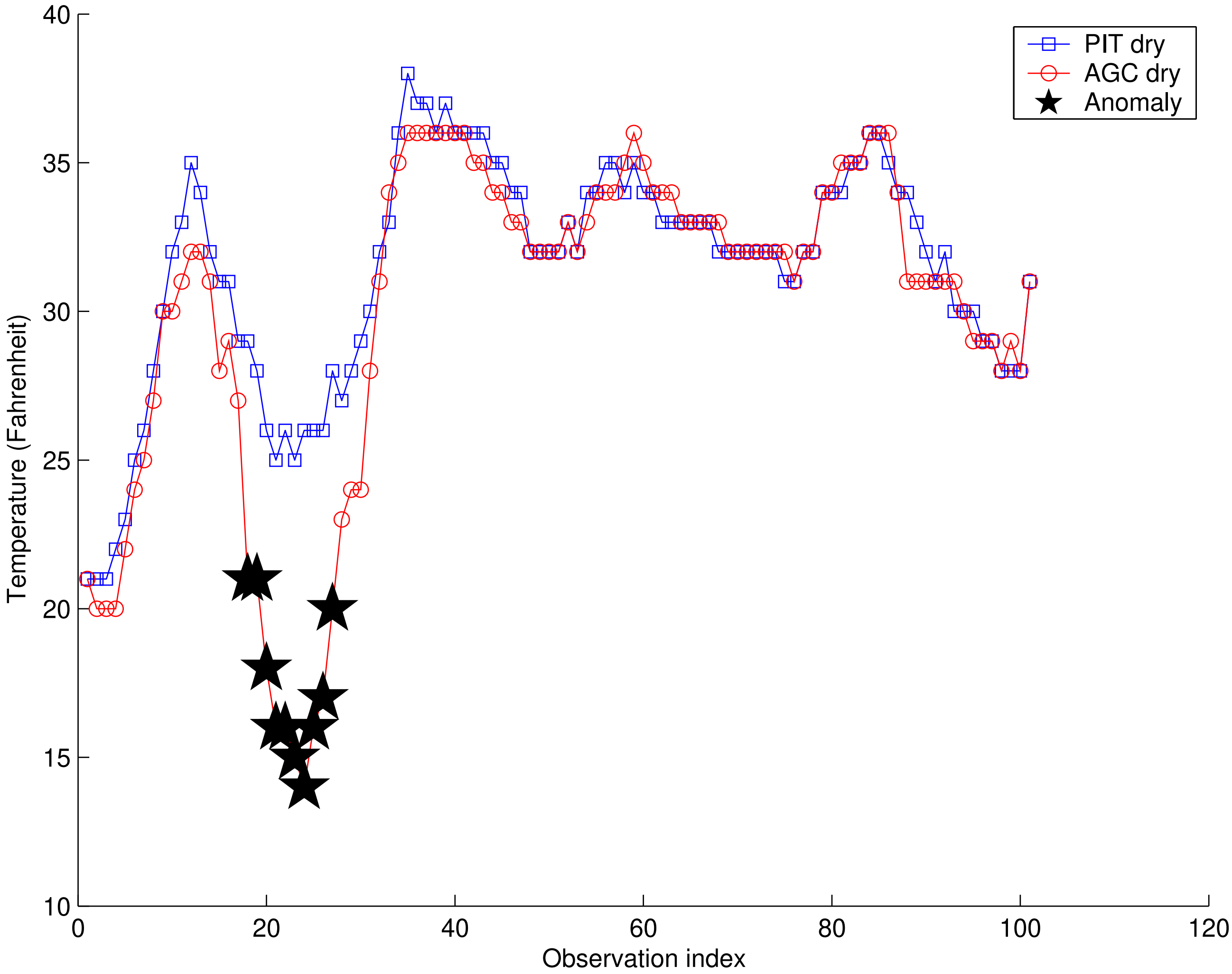

Figure 6.1: Anomalies in the dry-bulb-temperature attribute of the AGC data feed

reported by CUES. The misclassification rate is usually reasonable. The first subset shown in Table 6.1 has a high misclassification rate, but that is acceptable because CUES draws the user's attention to the region of anomalous behavior (though it flags only 2 out of the 10 consecutive observations as anomalous). The average misclassification rate (averaged over 86 subsets of 100 observations each) is 0.2 percent.

The reviewer also choose to accept the Daikon predicates that involve the relation between the dry-bulb temperature, the wet-bulb temperature, and the dew-point temperature, for each of the two locations. These predicates are

- $\texttt{PITdry}(t) \geq \texttt{PITwet}(t)$
- $\texttt{PITwet}(t) \geq \texttt{PITdew}(t)$

Index of observations	Anomalies	TP	FN	FP	Misclassification (%)
600–700	10	2	8	0	8
1800–1900	1	1	0	0	0
2900–3000	0	0	0	1	1
3900–4000	2	2	0	0	0
4200–4300	0	0	0	1	1
4300–4400	0	0	0	1	1
5100–5200	0	0	0	1	1
5800–5900	0	0	0	1	1
7100–7200	1	1	0	0	0
7200–7300	1	0	1	0	1
8000–8100	0	0	0	1	1

Table 6.1: Performance of the MUSCLES predicate for anomaly detection. The Misclassification rate $\frac{FP+FN}{SubsetSize}$ summarizes this performance. Only subset indices in which there are anomalies (either true or falsely reported by CUES) are shown (the misclassification rate for the other subsets is 0)

- $\texttt{AGCdry}(t) \geq \texttt{AGCwet}(t)$
- $\texttt{AGCwet}(t) \geq \texttt{AGCdew}(t)$

Though these predicates did not flag any anomalies, the reviewer thought they are effective as a sanity check over the data because they describe behavior that should always be true due to underlying physical properties. The detection and misclassification rates of the Daikon predicates were, therefore, 0. All observations obey these invariants. The data proprietors may already be checking and cleaning the data by applying similar invariants.

Assessment of practicality

Following is reviewer's A assessment of the practicality of CUES to potential users, summarizing the reviewer's statements and suggestions.

A one time set up effort is reasonable and worth while to a potential user of CUES. A user who understands the data is likely to be willing to do the set up and should easily be able to determine when a model flags relevant anomalies.

A potential user needs to have intermediate to advanced understanding of the data for the purpose this user is utilizing the data. The introduction to the concepts of what a technique does is achieved by running the technique and seeing the resulting anomalies. This results in a quick understanding of what a technique does so the user does not need prior knowledge.

It would be especially useful to have a high level explanations of the techniques along with examples of anomalies. CUES could present these examples to the user before starting the interaction for the user's data. This would give the user basic intuition about the effect of different parameter values. Having default parameter values is good. It would be helpful to have hints about about what the different parameter values mean. For example, a window of size one provides continuity checking. A sensitivity value of two is highly sensitive whereas a sensitivity value of ten is less sensitive.

There are many possible combinations of parameter values. It is hard to remember what combinations the user already investigated. A user interface could alleviate this problem. A wizard that remembers what the user did and how it worked (what the user thought of resulting anomalies) could help. However, going through the set up may be worth while even without such a wizard because of the big and immediate benefit. The user gets to immediately discover interesting data behavior (anomalies).

A user interface that would enable the user to work in a way similar to learning a new integrated development environment could be effective. To start the user would follow a wizard. Therefore, the user could set up an initial model in a few minutes. As the user gets a better understanding of CUES and the various predicate inference techniques the user could interact with CUES directly.

The initial technique recommendation and parameter setting is important because it affects the performance of the model for anomaly detection. Providing more hints about the domain could help in providing a better recommendation. For example, CUES could let the user choose from a pre-defined list of likely data characteristics and likely anomaly characteristics. This way CUES could recommend techniques and parameters that are likely to be a good match for the data and the user expectations.

This may be achieved by having specialized instances that come from a careful examination of the domain. A user who is a domain expert could do the specialization once an then all the users in that domain could benefit from the specialization.

6.2.3 Reviewer B

We summarize the interaction of reviewer B with CUES, the resulting model and its performance for anomaly detection in the data feeds, and the reviewer's assessment of the practicality of CUES for potential users. Reviewer B is a software engineering researcher with extensive experience in managing large software projects. Reviewer B has expertise in designing and delivering tools to enhance the productivity of programmers.

Interaction with CUES

Reviewer B interacted with CUES to run Percentile on a few data subsets, with the default parameter values. After examining the first subset in which anomalies were flagged the reviewer gained a better understanding of the kind of predicates that Percentile infers. He rejected all the Percentile predicates because he felt that the Percentile predicates do not make sense for the weather data.

Reviewer B interacted with CUES to select parameters for MUSCLES. He did not change the CUES default for the size of the moving window (100 observations) or the feed to predict (`PITdry`). He investigated attribute selection and different sensitivities. The reviewer started with the default values, using all six attributes. However, he felt that this resulted in too much information and he could not make judgments about anomalies. He felt that though the six attributes seemed highly correlated they measured different things. Therefore, he could not utilize all these attributes to indicate what is normal and what is anomalous. He then realized that he cared only about selecting a feed to use for the dry-bulb temperature. He decided that the most relevant information for this purpose would be in the dry-bulb temperatures at the two locations. Therefore, he selected the `PITdry` and `AGCdry` attributes.

Reviewer B was much happier with the results of running MUSCLES over the two dry-bulb attributes than with the results of running MUSCLES over all six attributes. He felt that he could understand better what was going on. When the reviewer started the interaction with CUES he felt that he could not distinguish between normal and anomalous observations. However, after seeing the observations that were normal and anomalous according to the MUSCLES predicates over the two dry-bulb attributes he gained a better understanding of the data and could make judgments.

He started with the CUES default parameters values for MUSCLES with the two dry-bulb attributes. He always asked CUES to update the MUSCLES predicates, selected a window of size one, and experimented with several sensitivities. He decided that a sensitivity of two was too sensitive. A sensitivity of either five or ten could be useful—he could imagine different users with different sensitivities. He decided that anomalies of the kind CUES flagged in two of the subsets (observations number

400-500 and 600–700) are the kind of anomalies he was interested in. He then looked for a sensitivity value that would detect these anomalies and would not produce too many false positives. A sensitivity of six flagged behavior that was not interesting. A sensitivity of seven performed well so the reviewer selected that value.

Reviewer B wanted to inspect not only data that CUES flagged (anomalous data) but also data that CUES did not flag (normal data). This helped him to understand how the data usually behaves.

Reviewer B indicated that it would be interesting to predict both feeds. However, after interacting with CUES to set parameter values for predicting `PITdry` he felt that this predicate performed well and wanted to keep the model simple. He felt that he would only want to make changes if he thought the detection was not effective.

Reviewer B interacted with CUES to select parameters for Daikon. After observing one run of Daikon with default parameter values the reviewer decided to reject the Daikon predicates. He felt that predicates that compared the different attributes (dry-bulb temperature, wet-bulb temperature, and dew-point temperature) were not relevant to his expectations.

Model and its performance

Reviewer B set up a model of proper behavior for the PIT dry-bulb temperature. He did so by interacting with CUES using the first 700 observations of the data feeds. CUES then automatically checked the resulting model over the remaining unseen observations (7900 observations). We report the model performance over all the observations, to enable a better comparison of the models the two reviewers selected.

Reviewer B choose a model similar to the one reviewer A choose. The sensitivity the two reviewers chose for comparing actual to estimated values differed. Reviewer A choose a sensitivity of ten while reviewer B chose a sensitivity of seven. The attributes they selected and the value of the other MUSCLES parameters were identical.

Reviewer B rejected both the Percentile predicates and the Daikon predicates.

The detection rate of the MUSCLES predicate, averaged over all the data subsets (8600 observations) is 0.6 percent. The detection rate is a higher than the rate resulting from the model reviewer A chose because reviewer B chose a more sensitive model. Reviewer B was concerned about difference of even a few degrees between the feeds, whereas reviewer A was concerned only about larger differences. Therefore, the anomaly rate, as judged by reviewer B was also higher than the anomaly rate as judged by reviewer A. Table 6.2 indicates a total of 55 anomalies, so the anomaly rate is 0.6 percent. The more sensitive model results in more false positives, as Table 6.2 summarizes. However, it also catches more anomalies that seem to result from true failures. For example, subset 7200–7300 contains one obvious anomaly in `AGCdry`. The value of one observation is very different from other observations around it, including the observation at the same time in the PIT location. Figure 6.2 depicts this. It shows the `PITdry` and `AGCdry` temperatures for observations in that subset. The x-axis is the sequential observation index in the window of observations. The y-axis is the temperature in Fahrenheit. A black star marks the anomalous observation. The average misclassification rate of the more sensitive predicate is 0.5 percent (the misclassification rate of the less sensitive model was 0.2 percent).

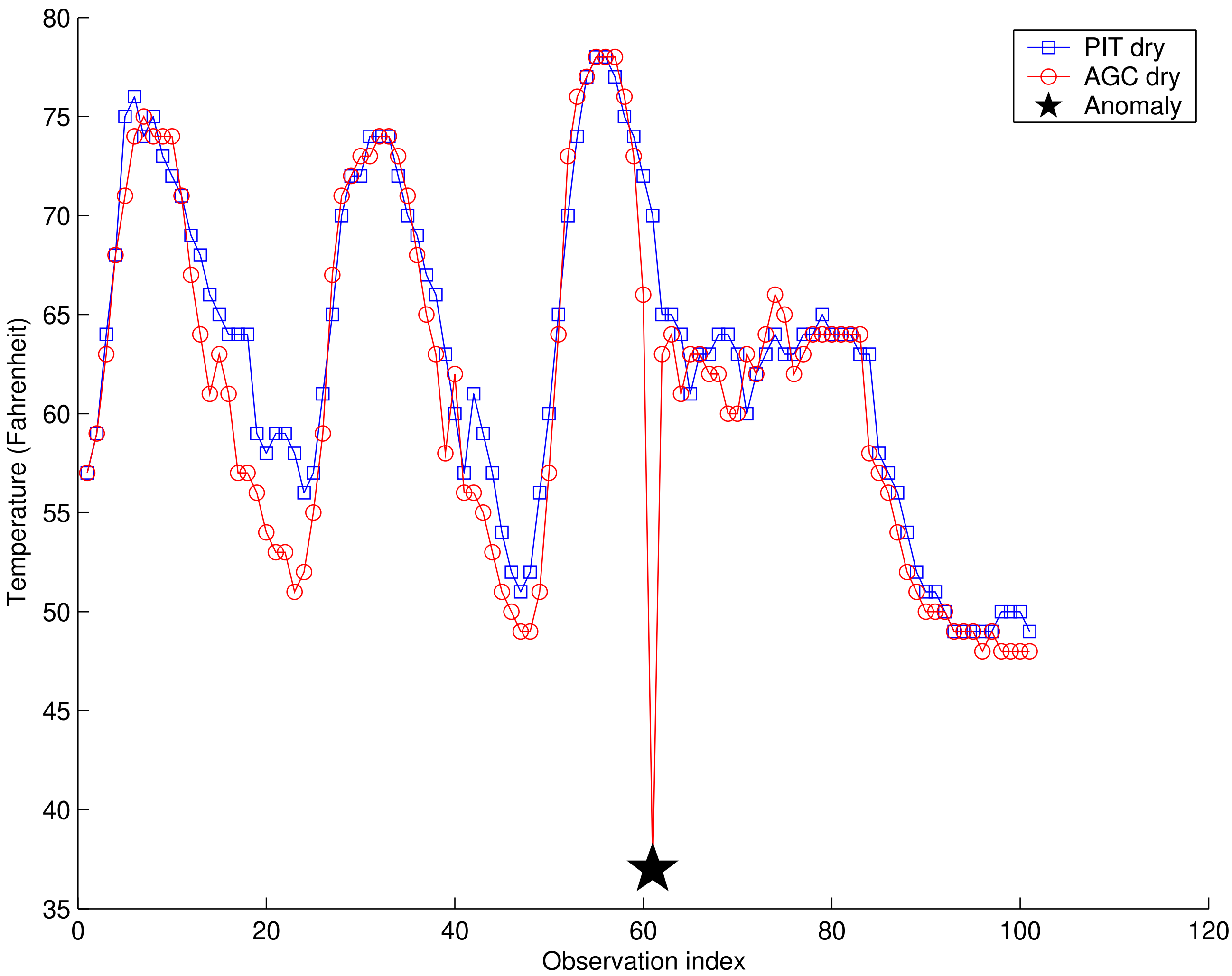

Figure 6.2: An anomaly in the dry-bulb-temperature attribute of the AGC data feed

Assessment of practicality

Following is reviewer's B assessment of the practicality of CUES to potential users, summarizing the reviewer's statements and suggestions.

A potential user of CUES could benefit from utilizing CUES. The user does not need to have prior understanding of the predicate-inference techniques.

CUES could benefit not only everyday users but also expert analysts. CUES could play an important role in the initial indication of what a specialist should inspect. CUES analysis is very quick and requires little effort to set up. It could provide a "quick look" at the data. Instead of starting with very complex techniques that require a lot of expert time the analyst would start with CUES. This would help to spend the most scarce resource—expert attention—more effectively.

Parameter setting should be visual to remove the need for technical terminology. A user interface could let the user select parameter values by showing in multiple windows anomaly detection resulting from various value selections.

The initial technique recommendation is important. If a user starts with one technique the user may insist on only using that technique, even if it does not make sense for the data. One option is to engage all the recommended techniques in parallel and immediately, so the user will not learn only a single technique. Another option is to do a better job at recommending the initial set. For example, CUES could describe the techniques by their effects on the data and provide examples, preferably from a similar domain. CUES could include a short questions and answers phase to achieve this or by explaining the advantages and weaknesses of each technique to the user.

It may be useful to support setting parameters by showing CUES what effect the user wants to achieve. The user will flag anomalies and ask CUES to find the value that would detect these anomalies most precisely.

6.2.4 Tuning alternative models

The models each reviewer selected did not require the tuning stage to execute its sanity-check heuristic (Section 4.4.2). This often suggests that the models are well-fit for the data. However, to examine the technical feasibility of CUES' end-to-end, we experiment with other possible models that do require the tuning stage to execute its sanity-check heuristic.

The first model is inferred by MUSCLES over all attributes (six attributes). The MUSCLES window size is zero (i.e., MUSCLES takes into account only current observation values), the MUSCLES sensitivity is five standard deviations, and the attribute to predict is `PITdry`. CUES (as during the reviews) re-infers the model over each data subset and checks the current model on the next data subset. The resulting detection is usually good: it flags most observations that the reviewers thought were anomalous, though there are a few false positives. For one data subset (observations 7300-7400) the model performs poorly and the resulting detection rate is too high (65%). The sanity-check heuristic of the tuning stage helps. It uses the previous model instead, resulting in a detection rate of 0%. Indeed, manual inspection of the data indicates that the `PITdry` values in this subset seem normal.

The second model is inferred by MUSCLES over the wet-bulb temperature attributes (`PITwet` and `AGCwet`). The MUSCLES window size is zero, the MUSCLES sensitivity is ten standard deviations, and the attribute to predict is `PITwet`. Again, CUES re-infers the model over each data subset and checks the current model on the next data subset. The resulting models indicate that the PIT and AGC attributes should have the same or nearly the same value. The resulting detection is good until data subset 2900–3000, for which the detection rate is too high. The sanity-check heuristic of the tuning stage tries to lower the high detection rate by using previous models. However, because all models are basically the same—an equality between the two attributes— using previous models does not help. The tuning stage asks the user to intervene. Indeed, a model of an equality between just these two attributes is obviously an over-simplification. The user may want to add attributes and change parameters. In general, the inability of the sanity-check heuristic to automatically improve the model performance by using a previous model may indicate that the model is not strong enough. The user may need to change parameters, add more attributes (if the user selected only a subset of the attributes), or even change techniques.

6.2.5 Analysis and Summary

We compare the model selection and parameter selection of the two reviewers, and their effect on anomaly detection in the data feeds. We then summarize the results of the validation study, concentrating on the assessment of the reviewers.

Comparison of reviewer choices

The models reviewer A and reviewer B chose were very similar. Both chose to concentrate on the values of the dry-bulb temperature at the two locations. This is probably because both followed the same usage scenario (Section A.5). In addition, it seemed difficult to understand predicates that contained more than a few (around four) attributes. Because both reviewers had limited knowledge about the weather domain—knowledge of casual users—they chose to concentrate on the attribute they understood best—the dry-bulb temperature.

Both reviewers rejected quickly the Percentile predicates. Indeed, such predicates are not relevant to weather temperature. This is because it may be normal for the temperature to be lower than or higher than the long-term average for a specific date.

Reviewer A accepted the Daikon predicates. Reviewer B rejected the same predicates. However, because these predicates detect no anomalies over the entire data (a year's worth of hourly observations) this does not affect the performance of the model for anomaly detection.

Because the models the reviewers chose are similar, these models detect the same kind of anomalies. The model that reviewer B chose (model B) is more sensitive than the model that reviewer A chose (model A). Therefore, model B flags more observations as anomalous. However, only in one case does this result in model B detecting a true anomaly that model A misses (the anomaly that Figure 6.2 shows). The other cases reflect a different sensitivity of the reviewers. Reviewer B wanted to be warned about smaller differences between the PIT and AGC observations than reviewer A. He considered less of these warnings to be false positives compared to reviewer A. In addition, he was less sensitive to false positives (the false positive rate of model B is nearly four times the false positive rate of model A, as the FP columns in Tables 6.2 and 6.1 indicate).

Summary of the validation study review

Each reviewer was able to set up a model of proper behavior for the weather data. They were not familiar with CUES or the data before the review (though both reviewers had a general understanding of weather conditions). Initially, both felt unable to make any judgments about the data. Within less than an hour they could make judgments about whether an observation was anomalous or not. They also gained a good understanding about CUES' predicate inference—enough to be able to utilize CUES effectively to detect the kind of behavior they thought was anomalous.

The models both reviewers selected (i.e., their precise expectations) were effective in detecting semantic anomalies in the dry-bulb temperature attributes of the weather data feeds, as measured by the detection and misclassification rates.

The tuning stage successfully re-inferred the model predicates over each data subset and the checking stage checked the resulting model over the next data subset. The models the reviewers selected did not require the tuning stage to execute its sanity-check heuristic. Experimenting with alternative models indicated this heuristic generally works well. When a model usually captures the data behavior reasonably well, the heuristic automatically ignores an occasional misguided model (e.g., a model inferred over a non-representative or highly anomalous data subset). When a model usually captures the data behavior poorly, the heuristic fails to correct this automatically and the tuning stage asks the user to intervene.

We did not make changes to CUES in preparation for or during the reviews. Both reviewers could utilize CUES to make their expectations precise, CUES' checking of the resulting precise expectations was effective in detecting semantic anomalies in the data feeds, and CUES' tuning worked well. This provides evidence in support of the technical completeness of CUES. However, CUES is currently only a prototype. The reviewers suggested areas for future work (see summary in Chapter 7). Most of these suggestions are related to an effective user interface. Both reviewers felt that the initial technique recommendation and default parameter values are important. Their suggestions for future work include ways to improve this recommendation either by adding more domain knowledge or through the design of a user interface.

Both reviewers felt that CUES provides significant benefit to a non-expert user. Setting up an initial model is quick. Moreover, it only requires understanding the application domain. It does not require experience with the inference techniques that CUES utilizes or with data mining in general. There is immediate reward to a user of CUES, in the form of the detected anomalies. Reviewer B even felt that CUES could benefit an expert analyst by indicating the most promising places for detailed analysis.

6.3 Cost-effectiveness analysis

Our experience with real-world data and with two users in the case studies enables us to provide rough estimates for the cost and the benefit a potential user of CUES may incur. The reviewers in the validation study corroborate and refine our estimates.

Our analysis is two fold. Section 6.3.1 concentrates on the cost and benefit of CUES. CUES is a technical solution to an important part of the problem of increasing the dependability of information systems that incorporate data feeds. Section 6.3.2 discusses how this solution fits within the bigger problem and the benefits it provides.

6.3.1 Cost and benefit for technical solution

We detail the major cost factors of the application of CUES and its major benefits. We then narrate on how the cost and benefit play together.

Cost

Applying CUES involves a cost to the user and a small computational cost. Utilizing data feeds requires effort but is part of the system development and therefore is independent of CUES. With respect to CUES, a user incurs the effort of setting up a model, handling flags regarding anomalies, and re-tuning the model.

Setup Setting up a model of proper behavior for data feeds is semi-automated. It relies heavily on the user. The main task of the user during CUES' setup stage is to make judgments about expected behavior. Technically, this translates into selecting parameter values and selecting predicates. In our experience, this takes no more than a few hours (usually about an hour) of the user's time. CUES provides assistance to the user by recommending techniques that are likely to be a good fit for the data and problem and by providing default values for parameters. The reviewers in the validation study suggested ways to improve this initial recommendation. However, these require more domain knowledge (see discussion in Section 7.2.1).

Checking Checking for anomalies is fully automated. The task of the user during CUES' checking stage is to handle the flags that CUES reports. Handling flags of true anomalies may require effort but is the result of problems in the data, not of overhead caused by CUES. In addition, noticing true problems in the data is the goal of utilizing CUES and one of the major benefits of utilizing CUES. However, CUES sometimes flags normal observations of anomalous. In our experience, the detection rate is usually close to the actual anomaly rate and the misclassification rate is low (usually less than 3%).

Tuning Tuning is semi-automated. The main task of the user during CUES' tuning stage is to adjust the model when the tuning stage asks the user to intervene. In our experience, this does not happen often. In addition, CUES couples user-intervention during tuning (if necessary) with inspecting anomalies. Therefore, it is done when the user is already paying attention. User intervention in tuning requires mainly re-examining the predicates in the model. It may also require selecting new values for parameters and possibly selecting different techniques.

Performance There is a small performance cost when running CUES. This is because CUES adds a step between getting the data and using the data. We do not have execution timing data. However, the performance cost is essentially similar to the cost of dynamic specification checking. When the tuning stage executes the sanity-check heuristic the performance incurs an additional penalty of checking multiple models. However, checking takes no more than seconds. Predicate inference takes no more than minutes.

Benefit

A user of CUES gets the benefit of automated detection of anomalies. Our case studies and validation study indicate that it is often possible to set up a model that results in effective detection, where the detection rate is close to the actual anomaly rate and the misclassification rate is low (around 97% of the anomaly call-outs are appropriate). The benefits of noticing a problem are user-dependent, but they are probably significant.

In addition to noticing anomalies, the user gains a better understanding of the data by making judgments about expected behavior. In our experience, this has resulted in clarifying the user's expectations, being able to complement existing documentation, and being able to correct existing documentation.

Balance

The main cost of using CUES is user effort for setting up a model of proper behavior for the data feeds. However, setup is only done once. In addition, the setup effort is reasonable because CUES provides assistance to the user—it provides default parameter values and a list of predicates for the user to choose from. The user gets an immediate payoff of better understanding the data and the user's own expectations. After a model exists, the user gets the benefit of noticing problems. In our experience, data that CUES flags is usually indicative of true problems.

6.3.2 Cost and benefit for original problem

Our long term goal is to increase the dependability of everyday information systems that incorporate data feeds. Detecting problems is the first step in many dependability enhancement techniques. Detecting anomalies in the data feeds that are part of an information system could contribute to an overall higher dependability of the system. With detection in place, analyses and decisions based on the operation of such information systems will be able to use good quality data or at least account appropriately for any deficiencies in the quality of data they are using. We contrast the cost and benefit of the existing support for detecting problems in data feeds (currently there is no support) with the cost and benefit of applying CUES. CUES greatly improves the ability to detect anomalies in data feeds. This is an important step towards increasing the dependability of everyday information systems.

Current support

The situation today is that there is no support for noticing problems in data feeds. The user of the data feeds has to manually inspect the data. This is unreasonable because the body of the data may be large and because it is hard for humans to concentrate on such repetitive tasks. The cost of the current solution is much higher than its benefits.

CUES

CUES changes this balance. CUES provides a means to detect problems in data feeds. It provides assistance for noticing problems. As Section 6.3 discusses, the benefits resulting from applying CUES are usually well-worth the effort. CUES changes the current situation of incorporating data feeds in any kind of automated processing to being tractable. With CUES, automated processing of data feeds can alert the user about values that seem unreasonable thus helping in preventing misguided decision making. Of course, the designer of an everyday information system (or its user) would need to invest more effort to achieve end-to-end increase in dependability (Section 7.2.2 discusses possible ways to increase the overall system dependability).

Index of observations	Anomalies	TP	FN	FP	Misclassification (%)
400–500	4	1	3	0	3
600–700	10	3	7	0	7
900–1000	4	0	4	0	4
1100-1200	0	0	0	1	1
1800–1900	1	1	0	0	0
2800–2900	2	1	1	1	2
2900–3000	1	1	0	2	2
3200–3300	0	0	0	1	1
3800–3900	2	2	0	0	0
3900–4000	7	7	0	0	0
4100–4200	0	0	0	1	1
4200–4300	0	0	0	1	1
4300–4400	1	1	0	2	2
4400–4500	0	0	0	2	2
4500–4600	1	1	0	0	0
4700–4800	0	0	0	1	1
4800–4900	1	1	0	1	1
5000–5100	5	3	2	2	4
5100–5200	1	1	0	3	3
5200–5300	1	1	0	0	0
5500–5600	1	1	0	1	1
5600–5700	0	0	0	2	2
5700–5800	1	1	0	0	0
5800–5900	1	1	0	0	0
6300–6400	3	2	1	0	1
7100–7200	1	1	0	0	0
7200–7300	1	1	0	1	1
7800–7900	1	1	0	0	0
8000–8100	5	1	4	0	4

Table 6.2: Performance of the MUSCLES predicate that reviewer B selected for anomaly detection. The Misclassification rate $\frac{FP+FN}{SubsetSize}$ *summarizes this performance. Only subset indices in which there are anomalies (either true or falsely reported by CUES) are shown (the misclassification rate for the other subsets is 0)*

Chapter 7

Conclusions and future work

This research provides CUES—Checking User Expectations about Semantics. CUES is a method and a prototype implementation for making user expectations precise and for checking these precise expectations. This research demonstrates that CUES is a practical means for helping users state precisely their own expectations about the behavior of data feeds and for exploiting the users' descriptions of expectations for semantic anomaly detection. Section 7.1 details the contributions of this research. Section 7.2 discusses possible directions for enhancing CUES and for making further advancements towards the above-stated long term goal.

7.1 Conclusions

This research proposes a method for helping users make their own expectations about the behavior of data feeds precise and for checking these precise expectations to detect semantic anomalies in the data feeds. The research demonstrates the feasibility of the method through a prototype implementation. Three case studies and a validation study, all with real-world data feeds, provide evidence in support of the practicality and usefulness of these method and prototype implementation, CUES, to everyday users. Section 7.1.1 summarizes this evidence (Chapter 6 provides details). Section 7.1.2 summarizes the contributions of this research.

7.1.1 Evidence

In the case studies and in the validation study the users were able to set up a model of proper behavior for the data feeds. They did so by utilizing CUES to make their expectations for the behavior of the data feeds precise. CUES successfully tuned the models to account for changing data behavior. CUES checked the resulting models to effectively detect semantic anomalies in the data feeds, as measured by the detection and misclassification rates.

The studies involved real-world data from various domains: truck weigh-in-motion data, online stock market data, and online weather conditions data. They involved multiple users with varying levels of domain knowledge: a domain expert in the truck weigh-in-motion case study, an everyday (non-expert) user in the stock daily low case study, and two colleagues with only basic everyday-usage knowledge about climatology in the validation study.

The time and effort necessary for interaction with CUES were reasonable. A user of CUES only needed enough domain knowledge to support the user's own usage of the data feeds. Setting up an initial model was quick (usually roughly an hour)[1]. Moreover, there was immediate reward to a user of CUES, in the form of the anomalies that CUES detects. Only in one instance of data feeds we experimented with—river gauges data feeds—was CUES ineffective. Details follow in Section 7.2.1. These data feeds exhibited data characteristics that required eliciting more domain knowledge than CUES' light-weight approach supports.

7.1.2 Contributions

The research presented in this dissertation provides CUES. CUES is a practical means for helping users state precisely their own expectations about the behavior of data feeds. CUES thereby provides a practical means of exploiting the users' descriptions of expectations for semi-automated semantic anomaly detection.

The case studies and the validation study demonstrated the benefits a user of CUES gets in return for a modest investment. The user's effort of selecting parameter values and predicates results in automated detection of semantic anomalies in data feeds. The data feeds may have missing specifications. Further, the user often gains a better understanding of the user's own expectations about the data behavior, of the actual behavior of the data, of missing specifications, and of imprecise existing specifications.

The truck WIM case study provides an especially encouraging example. Though the user in that case study was a domain expert, she discovered data behavior she was previously unaware of. That case study resulted in a contribution to the application

[1] The setup always included assistance by the author of CUES.

domain. It demonstrated a means to support automated cleaning of data from event-based monitoring systems in civil engineering [Raz et al., 2004b].

The research showed, through CUES' technique tool kit, how to exploit and adapt existing techniques from machine learning and dynamic program analysis for the purpose of inferring anomaly detection predicates. CUES provides a domain-independent and flexible framework for utilizing such techniques. This framework supports customization by defining many hooks for the user.

The research developed a novel template mechanism to elicit user expectations in the form of anomaly detection predicates and to support updating these predicates.

The research helped to clarify the need for user-centered specifications. Traditional component-centered specifications are insufficient for the research setting of this dissertation. Providing complete and precise component-based specifications requires a large investment. For everyday usages, such investment is not cost-effective. It is unreasonable to expect data feed providers to cover all possible changes and usages. Moreover, providing perfect specifications may be infeasible regardless of the investment. CUES provides a tractable means for dealing with incomplete specifications, for the purpose of semantic anomaly detection. CUES achieves this through its context-sensitive and user-centered approach.

7.2 Future work

Automated anomaly detection is a first step in increasing the dependability of information systems that incorporate data feeds. Section 7.2.1 discusses possible extensions to CUES to make it more effective in the task of anomaly detection. Section 7.2.2 discusses additional work that could further support the assembly of information systems from failure-prone parts.

7.2.1 Extensions to CUES

CUES requires additional effort to take it beyond the prototype stage. CUES would need a good user interface. In addition, CUES could be more useful to its users by supporting inference of models that are more robust and stateful.

User interface

The results of the case studies and validation study suggest CUES' approach is promising. Therefore, it would be worthwhile to invest further effort in CUES, especially in designing an effective user interface and better understanding the usability issues of CUES. This would require a user study. Reviewer A in the validation study (Section 6.2) suggested useful features of a user interface. One feature is a component that remembers the choices the user makes during set up and their impact (the resulting anomalies). This feature is intended to overcome the problem of having many possible parameter combinations. Another feature is providing hints about the meaning of the different parameter values. These hints could be a short text description accompanied by examples of anomalies. In general, a user interface that enables learning how to use the tool may be effective. This interface could draw from design principles of integrated development environments. This interface will provide wizards for quick start up and for promoting better understanding of the method. It will also enable direct control for users who feel comfortable with the method.

Inferring stronger models

Often, there are different modes in the data. An example is the river gauges data described shortly. The normal data behavior may change according to context that is not part of the data (i.e., hidden context). In addition, some user expectations may be domain-knowledge intense. To effectively describe the proper data behavior the context should be part of the model. This would require eliciting more domain knowledge than this research explores. CUES currently cannot model expectations that are domain knowledge intense. Future work could extend CUES by adding mechanisms for eliciting more domain knowledge. One approach could be to investigate combining the statistical approaches to machine leaning that CUES currently relies on with episodic approaches that model expert knowledge. An example of machine learning research that explores a similar approach is co-training [Blum and Mitchell, 1998] and active learning [Cohn et al., 1996].

This research explores the effectiveness of a light-weight approach to modeling proper behavior. It is reasonable to expect that adding domain knowledge would improve the performance of the approach. Reviewer A in the validation study (Section 6.2) suggested a possible way of integrating more domain knowledge in CUES. CUES could have domain-specialized instances. For example, an instance for natural phenomena such as weather. A knowledgeable user could perform the specialization.

Future users in the same domain would all benefit from this effort. The specialization would include modeling expert knowledge. A possible implementation could create a pre-defined list of likely data characteristics and likely anomaly characteristics. Technique recommendation, including appropriate default parameter values, will take into account these characteristics. In general, if the user is interested in anomalies resulting from a physical phenomenon, it would be best to model the physical phenomenon directly. Signal processing techniques are often effective for this purpose. CUES could be effective as a sanity checker over such data and possibly as an aid in cases where the user initially does not know how to create a physical model. Naturally, it is unlikely to perform as well as a physical model.

CUES cannot handle data feeds that have a varying time lag. As data mining and time series techniques for handling varying time lags become available and easy to use, CUES could incorporate such techniques. Unfortunately, the current state of the art in time series or digital signal processing techniques are not a good match for CUES because they require much expertise to set up [Papadimitriou et al., 2003a].

CUES could be extended to include more predicate inference techniques. In particular, adding more techniques to infer stateful predicates could enhance the syntax of the predicates it can infer and therefore its applicability.

We describe an unsuccessful case study using river gauges data. CUES was unable to infer useful predicates over this data. We suspect this is because the data has characteristics that require eliciting more domain knowledge than the minimal knowledge CUES explores. The river gauges data has varying time lags and modes.

The river gauges case study The river gauges case study used correlated real-world data feeds of current stream flow conditions. The United States Geological Survey (USGS) has been operating a stream gauging network to collect information about U.S. water resources. There are over 7000 stream-gauging stations across the United States. Each stream-gauging station records stage, or water depth, at a fixed time interval, usually every 15, 30, or 60 minutes. Stage is usually measured using either a float and pulley device or a pressure transducer. USGS hydrographers develop a relationship to estimate discharge (measured in cubic feet per second) from stage data (measured in feet). This relationship is developed by making frequent direct discharge measurements at stream-gauging stations [USGS FAQ, 2003].

The stream-gauging data serves for multiple analyses [USGS streamgaging, 2003], such as current forecasting and operational decisions, long-term resource planning, infrastructure design, and flood hazard mitigation. Current stream flow conditions are also needed by agencies with responsibility for operating water resources systems such as dams, locks, diversions, and conjunctive ground-water and surface-water supply systems. Real-time stream-flow data also serves for decisions about recreational activities, such as whether to take a canoeing trip on a specific time and location.

Each data feed that we collected contained data from one of three gauges reporting, as an Internet service, stream-flow conditions along the same river. The data was collected every hour, twenty four hours a day, seven days a week. The case study used data collected during roughly ten months. Each observation had stream flow values reported at the same time by each of the three gauges.

The river gauges case study examined automated and semi-automated ways to update predicates and ways to deal with time correlated data. It used two tool-kit techniques (augmented Daikon and MUSCLES).

The river gauges case study helped in scoping the capability of CUES. The river gauges data has several characteristic that CUES is currently incapable of handling: the data is modal (its behavior depends on changing conditions that are not part of the data) and it has time-lagged correlations with varying lags. These characteristics have in common a need for more domain knowledge. This research concentrates on the benefits achievable by using limited domain knowledge: knowledge of secondary users (e.g., using river gauges data for recreational purposes rather than for flood prediction) about the application domain.

7.2.2 Enhancing dependability

Detecting semantic anomalies in data feeds is a first step towards increasing their dependability. Once detection is in place, mitigation and repair can follow. The implementation of a system that incorporates data feeds could include mechanisms to restore or preserve normal operation. Automated fault identification would help in providing self-healing abilities because it would support remedial and mitigation techniques that require knowledge about the source of the problem.

It may be possible to use the precise model of proper behavior to decide whether two data feeds are substitutable. This would support self-healing. It would allow a system to, possibly temporarily, replace a faulty data feed with a feed that correctly provides the same service. Deciding whether two data feeds are substitutable could be based on behavioral subtyping [Liskov and Wing, 1994] in a similar way to behavioral subtesting [McCamant and Ernst, 2003]. Behavioral subtyping may provide a way to formally compare two services and decide whether one can substitute the other or what the difference between them is. One can think of user expectations as a subtype specification. The expectations of all the users describe all the different subtypes specifications. The system specification is the supertype specification. Liskov and Wing identify two subtype categories: extensive subtypes and constrained subtypes. The first extends the supertype by providing additional functionality and possibly also state. The second constraints the supertype. In the first category, the subtype must conserve the supertype's predicates and constraints. In the second category, the supertype must anticipate all the possible subtypes and have appropriate non-determinism to allow for all of them. In our domain, the supertype specifications are incomplete or missing. We can only observe the behavior of a particular subtype. This behavior is the data feed as it is used by a particular user.

Economic mitigation (e.g., insurance) could complement technical mitigation (fault tolerance) [Raz and Shaw, 2001]. Anomaly detection may constitute a first step towards economic mitigation. Economic mitigation requires recognizing loss events and determining the loss. For software, loss events are software failures. CUES' anomaly detection approach, with an additional failure identification component, may serve for loss recognition in systems that incorporate data feeds.

Anomaly detection could be embedded in a tool that would let people compose everyday information systems from data feeds. This would, of course, require additional research.

CUES may be useful not only at the consumer's side, but also at the producer's side. For example, historical behavior within a context may be used to automatically establish sanity checks for entering data into a database.

CUES may be useful not only to non experts but also to analysis or domain experts. The truck WIM case study provides evidence supporting the potential usefulness of CUES to domain experts. Reviewer B in the validation study (Section 6.2) indicated that CUES could benefit an expert analyst by indicating the most promising places for detailed analysis. CUES could provide a quick look at the data and potentially enable the expert to invest the expert's time more effectively. This is useful because expert attention is a scarce resource.

Specification proxies—the set of predicates CUES infers—may be useful for many tasks that rely on the existence of specifications. For example, they may be useful in assessing independence of data feeds, or they may help to deduce likely specifications for validating, enhancing or evolving existing specifications.

Appendix A

Validation study reviewer's guide

A.1 Goal

- Ensure CUES is technically complete by
  - setting up a model of proper behavior to detect semantic anomalies in the data feeds
  - verifying the model is effective for semantic anomaly detection in the data feeds, as measured by the detection and misclassification rates.
- Provide input about the potential usability and a rough estimate for the order of magnitude of the time required by a potential non-expert user.

A.2 CUES in a nutshell

CUES—Checking User Expectations about Semantics—is a three-stage, user-centric approach for coping with incomplete specifications of data feeds. The stages of CUES are

1. Setup stage: CUES begins by helping users make their expectations about data feed behavior precise. These precise expectations may serve as proxies for missing specifications. CUES automatically generates predicates over a sample of the data. These predicates describe various aspects of the data behavior. CUES lets the user choose from these predicates the predicates that describe behavior that is relevant to the user's expectations. The resulting model is the set of predicates that the user selected.

2. Checking stage: CUES checks the resulting model to automatically detect semantic anomalies—data feed behavior that falsifies these expectations.

3. Tuning stage: CUES also tunes the model to account for changing data feed behavior or changing user expectations.

A.3 Emphasis: reviewing the user interaction with CUES

1. Setup stage: The user sets parameters and selects predicates. The user may also choose pre-processing actions. CUES automatically generates predicates by applying existing predicate inference techniques. The user specifies basic information about the data scale and nature of correlations the user expects. CUES then recommends an initial set of predicate-inference techniques to the user. It runs these techniques with different options for parameters and shows the user the predicates and the resulting anomalies. The user selects parameter values that best reflect the user's expectations. The user may further select techniques. The user may also change the default moving window size.

   The user's major task is to choose a model of proper behavior from a list of predicates CUES automatically generates. The user sorts predicates into three categories: (1) "accept": the user expects this predicate to hold over the data, (2) "update": the user expects this predicate to hold over the data but also expects some of its numeric values to change over time, and (3) "reject": the user does not expect this predicate to hold over the data (the predicate's vocabulary may be irrelevant to the user's expectations or the predicate may describe noise in the data).

2. Checking stage: the user examines the anomalies that CUES detects over the current moving window. The detection is the result of checking the predicates the user selected in the setup stage.

3. Tuning stage: if CUES requests the user to intervene the user re-examines the model by possibly changing classification or classifying newly inferred predicates. Otherwise, CUES automatically updates the model.

CUES automatically checks the model and flags observations as anomalous. CUES automatically updates the model by applying the sanity-check heuristic and only asks the user to intervene if the resulting model performs consistently poorly.

Manually examine the data to set rules for what a true anomaly is. This will enable us to compute the misclassification rate.

A.4 Data

Two weather data feeds. One reports hourly sensor data from Pittsburgh International Airport (PIT). The other reports hourly sensor data from Allegheny County Airport (AGC). The distance between these airports is less than 20 miles. Each feed has the following attributes: dry bulb temperature, wet bulb temperature, and dew point temperature.

The dry-bulb is a thermometer that is dry. The dry-bulb temperature is the temperature that you hear on the weather report or that you read on a thermostat in your house.

The wet-bulb is a regular thermometer with a wet (distilled water) muslin wick covering it, and brisk air flowing across the wick. This results in evaporative cooling. The less humidity in the air, the larger the cooling effect. The cooling effect is also affected by the temperature.

If you know the dry-bulb and wet-bulb temperatures, you can look up a measure for humidity, for example, the dew point temperature.

A.5 Scenario

A person utilizes hourly weather feeds for automatic house-hold environmental control. The house is located between PIT and AGC so the person inputs feeds from these two locations to the environmental control. The person expects the feeds to be roughly similar.

A.6 Recommended techniques and their parameters

The measurement scale of all the attributes is interval. The correlation among multiple attributes is complicated, but between subsets of attributes the correlation could possibly be approximated by a linear relation.

CUES recommends the following techniques: Percentile, Daikon, MUSCLES. For all techniques, the user may change the CUES recommendation for size of the moving window (the size of the training data).

A.6.1 Percentile

Percentile estimates an interval for the values of an attribute. 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 only assumes that the distribution values are somewhat centered and is insensitive to extreme values.

Percentile has two parameters: x and y. The technique tool kit runs Percentile with percentiles x=25 and x=10. The default is x=25. When comparing the results of evaluating predicates to actual values the tool kit allows ranges that are either y=25% or y=50% larger than the inter-percentile ranges. The default is y=50%.

User selects values for x and y.

A.6.2 Daikon

The Daikon technique dynamically discovers likely program invariants over program execution traces by checking whether pre-defined relations hold. Daikon, being intended for program analysis, is capable of generating a potentially rich syntax. Some of the most interesting predicates for CUES purposes are:

1. an equality between 2 attributes or between an attribute and a numeric value; appropriate for interval scale data
2. an inequality ($!=, \leq, <, >, \geq$) between 2 attributes or between an attribute and a numeric value; appropriate for interval scale data
3. a linear relation among 2 or 3 attributes; appropriate for interval scale data
4. a 'one of' relation between an attribute and a set of values; appropriate for nominal scale data

The tool kit maps Daikon's program points and variables to CUES observations and attributes, respectively. Daikon assumes the data is clean, but data feeds are noisy. Therefore, the tool kit applies voting over Daikon's output: the tool kit runs Daikon on multiple subsets of the data and selects only the invariants that appear in multiple subsets. As a result, augmented Daikon has one parameter: the number of subsets to take into account in the voting. The tool kit runs Daikon over three to five subsets. The default is three.

User selects values for the number of subsets to compare in the voting.

A.6.3 MUSCLES

The MUSCLES technique is a multi-variate linear regression technique for time sequences. MUSCLES estimates the current value of one of the sequences as a linear combination of past values of the estimated sequence and past and current values of the other sequences.

MUSCLES assumes the data sequences are co-evolving over time and this correlation is linear. It is appropriate for interval scale data.

MUSCLES has several parameters:

1. window size
2. sequence to predict (estimate)
3. number of observations to predict
4. number of observations for training
5. forgetting factor between 0 and 1

The important parameters for CUES purposes are the size of the window to use for past observations, the sequence to predict, and the number of observations to use for training, . The technique tool kit uses the MUSCLES defaults for the other parameters. CUES recommends a window size according to one of many existing information criteria. Section 4.2.2 provides details about selecting a size for the technique window. CUES recommends to the user estimating the data feed that has the most lag compared to the other data feeds (See the stocks daily low case study, Section 5.3.2, for an example). Finding a good number of observations to use for training is the same problem as finding a good size for CUES moving window of observations. Section 4.2.2 discusses selecting a moving window size—a problem that is relevant to all the tool-kit techniques (because CUES always infers predicates over a moving window of observations). When comparing the results of evaluating predicates to actual values the tool kit allows a difference of two to ten standard deviations (of the estimated data feed). The default is a difference of five standard deviations.

The user selects the size of the MUSCLES window and the feed to predict (it is fine to select multiple feeds to predict).

In the weather case study the BIC for dry temperature is 28, for wet temperature 29, and for dew point temperature is 3. Obviously, a model with 28 or 29 past observations is too complicated to understand. The fact that the BIC differs greatly between the dry and wet and the dew point temperatures suggests it is not appropriate for this data. The user expectations in the scenario are for correlations among attribute values observed at the same time. Therefore, the user decides between a window of size 0 and a window of size 1.

The CUES default for the training data is 100 observations.

CUES recommendation for the feed to predict is not meaningful for the weather data because there is no obvious time lag in the data. Therefore, the user asks CUES to predict each of the data feeds. The user may choose any of the resulting predicates.

Bibliography

[Agrawal et al., 2001] Agrawal, R., Bayardo Jr., R. J., Gruhl, D., and Papadimitriou, S. (2001). Vinci: A service-oriented architecture for rapid development of web applications. In *international conference on World Wide Web*, pages 355–365.

[Agrawal et al., 1993] Agrawal, R., Imielinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *ACM SIGMOD International Conference on Management of Data*, pages 207–216.

[Akamai, 2004] Akamai (2004). Akamai, on demand computing solutions and services. URL: `http://www.akamai.com/`. Accessed April 2004.

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

[American Association of State Highway and Transportation Officials, 1986] American Association of State Highway and Transportation Officials (1986). AASHTO guide for design of pavement structures.

[Ammons et al., 2002] Ammons, G., Bodik, R., and Larus, J. (2002). Mining specifications. In *ACM SIGPLAN-SIGACT Symposium on Principles Of Programming Languages*, pages 4–16.

[Andrle et al., 2002] Andrle, S., McCall, B., and Kroeger, D. (2002). Application of weigh-in-motion (WIM) technologies in overweight vehicle enforcement. In *International Conference on Weigh-in-Motion*.

[Auton, 2003] Auton (2003). Auton Lab. URL: `http://www.autonlab.org`. Accessed April 2003.

[Avizienis et al., 2001] Avizienis, A., Laprie, J., and Randell, B. (2001). Fundamental concepts of dependability. Technical report, UCLA CSD Report no. 010028, LAAS Report no. 01-145, Newcastle University Report no. CS-TR-739.

[Avnur and Hellerstein, 2000] Avnur, R. and Hellerstein, J. M. (2000). Eddies: Continuously adaptive query processing. In *ACM SIGMOD international conference on Management of data*, pages 261–272.

[Babu and Widom, 2001] Babu, S. and Widom, J. (2001). Continuous queries over data streams. *SIGMOD Record*, 30(3):109–120.

[Bauer and Dengler, 1999] Bauer, M. and Dengler, D. (1999). Trias: Trainable information assistants for cooperative problem solving. In *International Conference on Autonomous Agents*, pages 260–267.

[Berners-Lee et al., 2001] Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. Scientific American. URL: `http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html`. Accessed November 2001.

[Beshears et al., 1998] Beshears, D. L., Muhs, J. D., Scudiere, M. B., Marlino, L., Taylor, B. W., Pratt, A., and Koenderink, R. J. (1998). Advanced weigh-in-motion system for weighing vehicles at high speed. Technical Report C/ORNL-95-0364, Lockheed Martin Energy Research Corporation.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *COLT: Workshop on Computational Learning Theory*.

[Box et al., 1994] Box, G. E. P., Jenkins, G. M., and Reinsel, G. C. (1994). *Time Series Analysis: Forecasting and Control*. Prentice Hall, third edition.

[Breunig et al., 2000] Breunig, M., Kriegel, H.-P., Ng, R., , and Sander, J. (2000). Lof: Identifying density-based local outliers. In *ACM SIGMOD International Conference. on Management of Data*.

[Brotherton and Mackey, 2001] Brotherton, T. and Mackey, R. (2001). Anomaly detector fusion processing for advanced military aircraft. In *Aerospace Conference*, volume 6 of *IEEE Proceedings*, pages 3125–3137.

[Buchheit, 2002] Buchheit, R. (2002). *Vacuum: Automated Procedures for Assessing and Cleansing Civil Infrastructure Data*. PhD thesis, Carnegie Mellon University, Civil Engineering Dept.

[Buchheit et al., 2003] Buchheit, R., Garrett Jr., J., and McNeil, S. (2003). An automated procedure to assess civil infrastructure data quality: Method and validation. Submitted.

[Buchheit et al., 2002] Buchheit, R., Garrett Jr., J., McNeil, S., and Chalkline, M. (2002). Automated procedures for improving the accuracy of sensor-based monitoring data. In *Applications of Advanced Technologies in Transportation*.

[Chalkline et al., 2001] Chalkline, M. H., Dahlin, C., and Guan, R. (2001). *Documentation of Traffic Data Collection and Data Warehouse at Mn/ROAD*. Office of Materials and Road Research.

[Chen et al., 2000] Chen, J., DeWitt, D. J., Tian, F., and Wang, Y. (2000). Niagaracq: A scalable continuous query system for internet databases. In *ACM SIGMOD International Conference on Management of Data*, pages 379–390.

[Clayton et al., 2002] Clayton, A., Montufar, J., and Middleton, D. (2002). Using weigh-in-motion data in a modern truck traffic information system. In *International Conference on Weigh-in-Motion*.

[Cohn et al., 1996] Cohn, D., Ghahramani, Z., and Jordan, M. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145.

[Cover and Thomas, 1991] Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. John Wiley.

[Cypher, 1993] Cypher, A., editor (1993). *Watch What I Do: Programming by Demonstration*. MIT Press. URL: `http://www.acypher.com/wwid/`. Accessed February 2002.

[Daikon Invariants, 2004] Daikon Invariants (2004). Invariants daikon detects. URL: `http://pag.lcs.mit.edu/daikon/download/doc/daikon.html`. Accessed February 2004.

[Datanomic, 2001] Datanomic (2001). Datanomic. URL: `http://www.datanomic.com/technology.htm`. Accessed November 2001.

[Dickinson et al., 2001] Dickinson, W., Leon, D., and Podgurski, A. (2001). Finding failures by cluster analysis of execution profiles. In *International Conference on Software Engineering*, pages 339–348.

[Duda et al., 2000] Duda, R., Hart, P., and Stork, D. (2000). *Pattern Classification,*. John Wiley and Sons, 2nd edition.

[Engler et al., 2001] Engler, D., Chen, D. Y., Hallem, S., Chou, A., and Chelf, B. (2001). Bugs as deviant behavior: A general approach to inferring errors in systems code. In *ACM Symposium on Operating Systems Principles*.

[Ernst et al., 2001] Ernst, M. D., Cockrell, J., Griswold, W. G., and Notkin, D. (2001). Dynamically discovering likely program invariants to support program evolution. *IEEE Transactions on Software Engineering*, 27(2):1–25.

[Fenton and Pfleeger, 1997] Fenton, N. E. and Pfleeger, S. L. (1997). *Software Metrics*, chapter 2. PWS Publishing Company, 2nd edition.

[FirstLogic, 2001] FirstLogic (2001). eDataQuality. URL: `http://www.firstlogic.com/solutions/technologies/iq/edataquality.asp`. Accessed November 2001.

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

[GoZilla, 2001] GoZilla (2001). Go!Zilla download manager. URL: `http://www.gozilla.com`. Accessed April 2001.

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

[Hofmeyr and Forrest, 2000] Hofmeyr, S. and Forrest, S. (2000). Architecture for an artificial immune system. *Evolutionary Computation Journal*, 8(4):443–473.

[Hulten et al., 2001a] Hulten, G., Spencer, L., and Domingos, P. (2001a). Mining time-changing data streams. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 97–106.

[Hulten et al., 2001b] Hulten, G., Spencer, L., and Domingos, P. (2001b). Mining time-changing data streams. In *International Conference on Knowledge Discovery and Data Mining*.

[IBM autonomic computing, 2001] IBM autonomic computing (2001). IBM Autonomic Computing. URL: `http://www.research.ibm.com/autonomic/overview/`. Accessed November 2001.

[IBM eLiza, 2001] IBM eLiza (2001). IBM. eLiza: self-manging servers. URL: `http://www-1.ibm.com/servers/eserver/introducing/eliza/`. Accessed November 2001.

[IBM Web Services, 2001] IBM Web Services (2001). IBM. Web Services. URL: `http://www-4.ibm.com/software/solutions/webservices/overview.html`. Accessed November 2001.

[Information Criteria, 2004] Information Criteria (2004). Information criteria in a nutshell. URL: `http://www.agapow.net/science/maths/info_criteria.html`. Accessed February 2004.

[Johnson et al., 1998] Johnson, T., Kwok, I., and Ng, R. T. (1998). Fast computation of 2-dimensional depth contours. In *International Conference on Knowledge Discovery and Data Mining*, pages 224–228.

[Jolliffe, 1986] Jolliffe, I. T. (1986). *Principal component analysis*. Springer-Verlag, New-York.

[Kay, 1988] Kay, S. M. (1988). *Modern Spectral Estimation: Theory and applications*. Prentice-Hall.

[Knoblock et al., 1999] Knoblock, C., Lerman, K., Minton, S., and Muslea, I. (1999). Accurately and reliably extracting data from the web: A machine learning approach. In *Data Engineering Bulletin*.

[Knorr et al., 2000] Knorr, E. M., Ng, R. T., and Tucakov, V. (2000). Distance-based outliers: algorithms and applications. *The International Journal on Very Large Data Bases*, 8:237–253.

[Kolmogorov, 1965] Kolmogorov, A. N. (1965). Three approaches to the quantitative definition of information. *Problems of Information and Transmission*, 1.

[Kushmerick, 1999] Kushmerick, N. (1999). Regression testing for wrapper maintenance. In *National Conference on Artificial Intelligence*, pages 74–79.

[Lane and Brodley, 1998] Lane, T. and Brodley, C. E. (1998). Approaches to online learning and concept drift for user identification in computer security. In *Knowledge Discovery and Data Mining*, pages 259–263.

[Langley, 2000] Langley, P. (2000). The computational support of scientific discovery. *International Journal of Human-Computer Studies*, 53:393–410.

[Lee and Souny-Slitine, 1998] Lee, C. and Souny-Slitine, N. (1998). Final research findings on traffic-load forecasting using weigh-in-motion data. Technical Report 987-7, Texas University, Austin.

[Lee et al., 2000] Lee, Y. W., Bowen, P. L., Funk, J. D., Jarke, M., Madnick, S. E., and Wand, Y. (2000). Data quality in internet time, space, and communities (panel session). In *International Conference on Information Systems*.

[Levy et al., 1998] Levy, A., Knoblock, C., Minton, S., and Cohen, W. (1998). Trends and controversies: Information integration. *IEEE Intelligent Systems*, 13(5):12–24.

[Liskov and Wing, 1994] Liskov, B. and Wing, J. (1994). A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems*, 16(6):1811–1841.

[Lyu, 1996] Lyu, M. R. (1996). *Handbook of Software Reliability Engineering*. IEEE Computer Society Press and McGraw-Hill.

[Mackey, 2001] Mackey, R. (2001). Generalized cross-signal anomaly detection on aircraft hydraulic system. In *Aerospace Conference*, volume 2 of *IEEE Proceedings*, pages 657–668.

[Mackey et al., 2001] Mackey, R., James, M., Park, H., and Zak, M. (2001). BEAM: Technology for autonomous self-analysis. In *Aerospace Conference*, volume 6 of *IEEE Proceedings*, pages 2989–3001.

[Madden et al., 2002] Madden, S., Shah, M., Hellerstein, J. M., and Raman, V. (2002). Continuously adaptive continuous queries over streams. In *ACM SIGMOD international conference on Management of data*, pages 49–60.

[Magnum Opus, 2002] Magnum Opus (2002). Magnum opus, association rule discovery. URL: `http://www.rulequest.com/MagnumOpus-info.html`. Accessed January 2002.

[Marple, 1987] Marple, S. (1987). *Digital Spectral Analysis with Applications*. Prentice Hall.

[McCamant and Ernst, 2003] McCamant, S. and Ernst, M. (2003). Predicting problems caused by component upgrades. In *European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 287–296.

[Miller and Myers, 2001] Miller, R. C. and Myers, B. A. (2001). Outlier finding: Focusing user attention on possible errors. In *Annual ACM Symposium on User Interface Software and Technology*, pages 89–90.

[Minnesota Department of Transportation, 2000] Minnesota Department of Transportation (2000). *Minnesota Trucking Regulations*. Minnesota Department of Transportation. Technical Report.

[Minnesota Department of transportation, 2002] Minnesota Department of transportation (2002). Mn/DOT office of materials and road research. URL: `http://mnroad.dot.state.mn.us`. Accessed May 2002.

[Minnesota Road Research Section, 2003] Minnesota Road Research Section (2003). Mn/ROAD research projects. URL: `http://www.mrr.dot.state.mn.us/research/MnROAD_Project/projects90.asp`. Accessed Feb 2003.

[Mitchell, 1997] Mitchell, T. (1997). *Machine Learning*. McGraw Hill.

[Najafi and Blackadar, 1998] Najafi, F. T. and Blackadar, B. (1998). Analysis of weigh-in-motion truck traffic data. In *40th Annual Meeting of the Transportation Research Forum*.

[NCDC, 2003] NCDC (2003). Unedited local climatological data. URL: `http://www.ncdc.noaa.gov/servlets/ULCD`. Accessed January 2004.

[Nielsen, 1992] Nielsen, J. (1992). The usability engineering life cycle. *IEEE Computer*, pages 12–22.

[Paladyne, 2001] Paladyne (2001). Paladyne data discovery. URL: `http://www.paladyne.com/software/discovery.html`. Accessed November 2001.

[Papadimitriou et al., 2003a] Papadimitriou, S., Brockwell, A., and Faloutsos, C. (2003a). Awsom: Adaptive, hands-off stream mining. In *International Conference on Very Large Databases*.

[Papadimitriou et al., 2003b] Papadimitriou, ., andPhil Gibbons, H. K., and Faloutsos, C. (2003b). loci: Fast outlier detection using the local correlation integral. In *International Conference on Data Engineering*.

[Pelleg and Moore, 2000] Pelleg, D. and Moore, A. (2000). X-means: Extending k-means with efficient estimation of the number of clusters. In *International Conference on Machine Learning*, pages 727–734.

[Pelleg and Moore, 2001] Pelleg, D. and Moore, A. (2001). Mixtures of rectangles: Interpretable soft clustering. In *International Conference on Machine Learning*, pages 401–408.

[Raman and Hellerstein, 2001] Raman, V. and Hellerstein, J. M. (2001). Potters wheel: An interactive data cleaning system. In *International Conference on Very Large Data Bases*, pages 381–390.

[Ramaswamy et al., 2000] Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *ACM SIGMOD International Conference on Management of Data*.

[Raz et al., 2003] Raz, O., Buchheit, R., Shaw, M., Koopman, P., and Faloutsos, C. (2003). Eliciting user expectations for data behavior via invariant templates. Technical report, CMU-CS-03-105.

[Raz et al., 2004a] Raz, O., Buchheit, R., Shaw, M., Koopman, P., and Faloutsos, C. (2004a). Automated assistance for eliciting user expectations. In *International Conference on Software Engineering and Knowledge Engineering*. To appear.

[Raz et al., 2004b] Raz, O., Buchheit, R., Shaw, M., Koopman, P., and Faloutsos, C. (2004b). Detecting semantic anomalies in truck weigh-in-motion traffic data using data mining. *Journal of Computing in Civil Engineering*. To appear.

[Raz et al., 2002] Raz, O., Koopman, P., and Shaw, M. (2002). Semantic anomaly detection in online data sources. In *International Conference on Software Engineering*, pages 302–312.

[Raz and Shaw, 2000] Raz, O. and Shaw, M. (2000). An approach to preserving sufficient correctness in open resource coalitions. In *International Workshop on Software Specification and Design*, pages 159–170.

[Raz and Shaw, 2001] Raz, O. and Shaw, M. (2001). Software risk management and insurance. In *Position Paper for Workshop on Economics-Driven Software Engineering Research*.

[Runeson et al., 2001] Runeson, P., Ohlsson, M., and Wohlin, C. (2001). A classification scheme for studies on fault-prone components. In *Product focused software process improvement*, pages 341–355.

[S1, 2000] S1 (2000). Stock quotes service. URL: `http://finance.northernlight.com`. Accessed September–November 2000.

[S2, 2000] S2 (2000). Stock quotes service. URL: `http://qs2.cnnfn.com`. Accessed September–November 2000.

[S3, 2000] S3 (2000). Stock quotes service. URL: `http://quote.pathfinder.com`. Accessed September–November 2000.

[Schlimmer and Granger, 1986] Schlimmer, J. and Granger, R. (1986). Incremental learning from noisy data. *Machine Learning*, 1:317–354.

[Sequeira and Zaki, 2002] Sequeira, K. and Zaki, M. (2002). ADMIT: Anomaly-based data mining for intrusions. In *international conference on Knowledge discovery and data mining*, pages 386–395.

[Seshadri et al., 1996] Seshadri, P., Livny, M., and Ramakrishnan, R. (1996). The design and implementation of a sequence database system. In *International Conference on Very Large Data Bases*, pages 99–110.

[Shaw, 2000] Shaw, M. (2000). Sufficient correctness and homeostasis in open resource coalitions: How much can you trust your software system? In *International Software Architecture Workshop*.

[Shaw and Garlan, 1996] Shaw, M. and Garlan, D. (1996). *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall.

[Sousa and Garlan, 2002] Sousa, J. and Garlan, D. (2002). Aura: An architectural framework for user mobility in ubiquitous computing environments. In *IEEE/IFIP Conference on Software Architecture*, pages 29–43.

[Stephen Bay, 2003] Stephen Bay, M. S. (2003). Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *international conference on Knowledge discovery and data mining*, pages 29–38.

[Strong et al., 1997] Strong, D. M., Lee, Y. W., and Wang, R. Y. (1997). Data quality in context. *Communications of the ACM*, 5(40):103–110.

[Trillium, 2001] Trillium (2001). Trillium data quality. URL: `http://www.trilliumsoft.com`. Accessed November 2001.

[TSA, 2003] TSA (2003). Time Series Analysis Toolbox. URL: `http://www.dpmi.tu-graz.ac.at/~schloegl/matlab/tsa/`. Accessed April 2003.

[UDDI, 2000] UDDI (2000). UDDI. UDDI Technical White Paper. URL: `http://www.uddi.org/pubs/Iru\_UDDI\_Technical\_White\_Paper.pdf`. Accessed November 2001.

[UDDI, 2001] UDDI (2001). UDDI. UDDI 2.0, API Specification. URL: `http://www.uddi.org/pubs/ProgrammersAPI-V2.00-Open-20010608.pdf`. Accessed November 2001.

[USGS FAQ, 2003] USGS FAQ (2003). USGS frequently asked questions. URL: `http://dc.water.usgs.gov/faq/realtime.html`. Accessed October 2003.

[USGS streamgaging, 2003] USGS streamgaging (2003). A new evaluation of the usgs streamgaging network—a report to congress. URL: `http://water.usgs.gov/streamgaging/index.html`. Accessed October 2003.

[Villemeur, 1992] Villemeur, A. (1992). *Reliability, Availability, Maintainability, and Safety Assessment*. Jon Wiley & Sons.

[W3C. Semantic Web, 2001] W3C. Semantic Web (2001). World Wide Web Consortium (W3C). The Semantic Web, Activity. URL: `http://www.w3.org/2001/sw/`. Accessed November 2001.

[W3C SOAP, 2001] W3C SOAP (2001). World Wide Web Consortium (W3C). SOAP 1.1, Note. URL: `http://www.w3.org/TR/2000/NOTE-SOAP-20000508`. Accessed November 2001.

[W3C WSDL, 2001] W3C WSDL (2001). World Wide Web Consortium (W3C). WSDL 1.1, Note. URL: `http://www.w3.org/TR/2001/NOTE-wsdl-20010315`. Accessed November 2001.

[W3C XML, 2001] W3C XML (2001). World Wide Web Consortium (W3C). XML 1.0, Recommendation. URL: `http://www.w3.org/TR/2000/REC-xml-20001006`. Accessed November 2001.

[Wand and Wang, 1996] Wand, Y. and Wang, R. Y. (1996). Anchoring data quality dimensions in ontological foundations. *Communications of the ACM*, 39(11):86–95.

[Wei, 1990] Wei, W. (1990). *Time Series Analysis: Univariate and Multivariate Methods*. Addison Wesley.

[Widmer and Kubat, 1996] Widmer, G. and Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine learning*, 23:69–101.

[Wong et al., 2003] Wong, W.-K., Moore, A., Cooper, G., and Wagner, M. (2003). Bayesian network anomaly pattern detection for disease outbreaks. In *International Conference on Machine Learning*, pages 808–815.

[Yi et al., 2000] Yi, B.-K., Sidiropoulos, N. D., Johnson, T., Jagadish, H. V., Faloutsos, C., and Biliris, A. (2000). Online data mining for co-evolving time sequences. In *International Conference on Data Engineering*.

[Zhu et al., 2004] Zhu, C., Faloutsos, C., Kitagawa, H., and Papadimitriou, S. (2004). Obe: Outlier by example. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.