The Case for Data Assurance

Martin Arlitt, Keith I. Farkas
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto
HPL-2005-38
February 24, 2005*

E-mail: firstname.lastname@hp.com

Information Technology Computing Environments, monitoring data, data quality

Monitoring data, while used historically primarily for performance analysis and debugging, is increasingly playing a more central role in the control and management of IT environments. Yet, monitoring data contains errors, errors which can and do affect the decisions made by the automatic control and management systems. To support the increasing scale and complexity of IT environments, it is imperative that data quality problems be recognized before automated systems act on the data. We believe this need can be met with a data assurance layer, which sits between the data collectors and consumers (human or software), and which is responsible for informing the data consumers about its quality. In this paper, we lay the groundwork for this layer by examining the imperfections that exist in monitoring data, and then identifying some promising directions for further investigation.

* Internal Accession Date Only Approved for External Publication

The Case for Data Assurance

Martin Arlitt and Keith I. Farkas
Hewlett-Packard Laboratories
firstname.lastname@hp.com

Abstract

Monitoring data, while used historically primarily for performance analysis and debugging, is increasingly playing a more central role in the control and management of IT environments. Yet, monitoring data contains errors, errors which can and do affect the decisions made by the automatic control and management systems. To support the increasing scale and complexity of IT environments, it is imperative that data quality problems be recognized before automated systems act on the data. We believe this need can be met with a data assurance layer, which sits between the data collectors and consumers (human or software), and which is responsible for informing the data consumers about its quality. In this paper, we lay the groundwork for this layer by examining the imperfections that exist in monitoring data, and then identifying some promising directions for further investigation.

1 Introduction

Emerging utility computing models [1,2,3] and the growing scale of Information Technology (IT) compute environments are changing the role of monitoring data. Historically, monitoring data has been used predominantly for analyzing performance, and for identifying and fixing performance and functional problems. However, increasingly, monitoring data is playing a more direct role in the control and management of computing environments. Indeed, emerging automatic control and management systems [4,5,6] use monitoring data as a basis for the decisions they make.

While, as in other domains, poor quality data can lead to bad decisions, the dynamics of current compute infrastructure magnify the consequences. The growing scale, adaptivity, and complexity of the infrastructure is making it increasingly difficult for humans to ensure that data is being collected properly, let alone sanity check the collected data values. Moreover, control systems are using this data to make decisions more quickly than could a human, thus challenging the ability of humans to keep up. Finally, the growing complexity of the infrastructure reduces the ability of humans to reason about the cause-and-effect relationships between components in the environment, thus potentially leading to cascading failures following an initial bad decision. Clearly, to prevent the magnification of any data quality problems, these problems must be recognized early.

To address this problem, we believe there is a need for a software layer that sits between the data collectors and the users of the monitoring data. This *data assurance* layer will be responsible for evaluating the quality of the data, correcting simple errors, and providing the data consumers with quantitative measures of the data quality. In short, this layer will automate many of the data preparation tasks humans now do, tasks guided by human intuition. Consequently, it will help enable more automatic and adaptive computing environments.

In this paper, we present a necessary first step to creating this data assurance layer, namely an examination of the imperfections that exist in monitoring data in IT environments. While our focus is system-level and application-level resource utilization metrics (e.g., CPU utilization), our conclusions are not necessarily limited to this domain. Our examination draws from work in epistemology, the branch of philosophy that studies the nature of knowledge, its foundation, scope, and validity. To our knowledge, we are the first to present a systematic examination of the problems that occur in the monitoring data for IT systems, and the first to recognize that data assurance is required for the control and management of IT computing environments.

We also present a discussion of promising directions for further investigation, and summarize some of the open issues, whose resolution requires contributions from multiple communities.

The remainder of the paper is organized as follows. Section 2 identifies relevant areas of related work. Section 3 discusses uses of monitoring data in computing environments. Section 4 summarizes an existing taxonomy [9] of imperfection in data, classifies the type of monitoring data that we focus on in this paper, and describes a mapping of problems we have encountered with monitoring data onto the taxonomy. Section 5 elaborates on the implications of imperfections in monitoring data, and how we intend to apply data assurance in IT computing environments. Section 6 concludes the paper with a summary of our work.

2 Related Work

Data quality is a recognized problem in a number of domains, one of which is military intelligence. Keegan notes in [7] that data acceptance is an extremely important step in the processing of military intelligence data. Data is not trusted until assurances can be obtained about its correctness and consistency. Incorrect or inconsistent data can have profound consequences.

Data quality is also the subject of several conferences including the International Conference on Information Quality [8], which is focused on the management issues related to information quality, and covers such topics as information quality concepts, management, best practices, and case studies. While such work is related to our own, we are not aware of any that specifically address the issue of data quality in the control and management of IT resources.

An example from one of these communities that clearly shows the importance of data quality is a sensor-to-shooter (STS) network [10] for use in battlefield situations to automatically detect, target, and engage enemy targets. In a fully automated STS network, humans are removed from the loop with the aim of reducing the army's exposure and improving response times. Yet, removing humans removes human intuition about whether an actual target was detected, and whether the correct actions are being taken to engage it, for example, that the weapons are pointed in the correct direction.

In the context of the control and management of IT environments, the consequences of incorrect decisions could be as severe as in an STS network. Indeed, computer systems control many aspects of today's physical infrastructure. Incorrect decisions could have significant personal and business ramifications. We thus believe that data assurance has an important role to play in IT environments.

Finally, the nature of knowledge, its foundations, scope, and validity is being studied by epistemologists. This area of study provides a basis for understanding the type of errors that can occur, and how these errors give rise to uncertainty; our discussion in Section 4 is based on some work done by Smets [9] in this area.

3 Uses of Monitoring Data

Data assurance is an increasingly important problem for the control and operation of computing environments owing to data playing an increasing central role in the decisions made by control and management systems. In this section, we describe several such systems, highlighting how monitoring data is used, and some of the consequences of bad data.

In enterprise IT environments, one important management task is the allocation of compute resources to applications such that the time-varying resource needs of the often long-running enterprise applications are met. This task is performed by a capacity manager [4][5]. Capacity management algorithms base the allocation decisions on monitoring data that reflects the past resource needs of the applications and the capabilities of the target computer systems. Typical metrics include peak CPU utilization, peak memory utilization, and network bandwidth. Scheduling decisions are made by first evaluating for each measurement interval how much resource capacity is required by each application, and then selecting application groupings for which the aggregate resource needs can be met at each interval by a given target computer system. If, however, the metric values are incorrect, a capacity manager may either over-subscribe the resources of a given system, thus impacting the performance of the applications, or under-subscribe the resources, thus increasing the cost of running the IT environment. In either case, this incorrect allocation decision could lead to further incorrect allocations, thus complicating correction efforts.

A second important management task is that of diagnosing performance problems once they occur. Cohen et al. [11], for example, use probabilistic models to identify relationships between relevant system metrics and unmet performance targets. These models are then used to locate the probable cause of a problem. Thus, for example, if an application is not meeting its performance target, this approach can be used to determine whether the CPU utilization, disk read operations, swapping, or combination thereof is the likely cause. Moreover, the approach can be used to identify normal operating ranges for the metrics, and thus identify patterns that likely forecast performance problems. Given the statistical nature of this approach, poor quality data may impact the analyses, although, the impact cannot be analyzed a priori. Poor analysis may lead to an increase in false positives, which are costly and time consuming to resolve.

A third management task concerns tuning the performance of an application. Xue et al [6] are investigating using feedback control algorithms to dynamically and automatically adjust the resources allocated to a resource container (e.g., virtual machine) so as to deliver the required performance with minimal resource allocation levels. The control algorithms use monitoring data to assess whether the desired performance is being met, and if not, to assess the impact of changes in resource allocations. While short term non-repeating data errors may not impact the performance of the system, longer term or repeated errors could affect the stability of the control system or its ability to meet its design objectives. These effects could affect the behavior of other inter-dependent control systems, if they exist.

4 Problem and Measurement Taxonomies

Smets discusses the importance of cooperation between the systems and uncertainty communities, in order to arrive at more accurate solutions [9]. Smets suggests the following methodology when confronted with imperfect data: first, realize the form of imperfection; then, select the most appropriate model for addressing the imperfections. In this paper we focus on the first of these steps, and attempt to better understand the forms of imperfection that exist in system utilization data. We also discuss some simple techniques that can be used to identify some of these imperfections, as well as possible methods for addressing several of these problems. We defer a thorough examination of modeling the imperfections to future work.

The remainder of this section is organized as follows. Section 4.1 presents a visualization of the Smets taxonomy of imperfect information [9], and applies it to the IT context. Section 4.2 outlines properties of system utilization data. Section 4.3 provides a mapping between imperfections we have encountered in system utilization data and Smets taxonomy. Section 4.4 summarizes the contributions of this section.

4.1 Smets Taxonomy

Smets provides a categorization of imperfect data along three axes: *imprecision*, *inconsistency*, and *uncertainty* [9]. Imprecision and inconsistency are properties of the data; we focus on these aspects in this section. Uncertainty concerns the state of knowledge of the users of the data; that is, how informed the users are with respect to the quality of the data. We consider this third aspect in Section 5.

Figure 1 shows the complete categorization of imperfect data as provided by Smets [9].[1] Below we reiterate the definitions provided in [9] and provide some IT specific examples.

Imprecision: a piece of data is imprecise if it is not exact. Smets identified two general classes of imprecise measurements: those without an error component, and those with an error component.

Imprecise measurements without an error component are further classified into either vagueness or

1. *Please note that the graphical representation of Smets' taxonomy is our interpretation of his textual description.*

missing. The most specific classes are defined below:

- *ambiguous:* something that has multiple meanings; e.g., 'The system is up' could mean the system is powered or could run a job.
- *approximate:* close to reality and well defined, and decidable if the information is correct or not; e.g., 'The number of servers is in the 40's is true if the actual number is 42.
- *unclear:* something that is not well defined and not decidable; e.g., 'The number of servers is close to 40' when the actual number is 42.
- *incomplete:* data is missing, but is not expected; e.g., the number of virtual machines running on a system is not defined for a system that is turned off
- *deficient:* data is missing, and is expected; e.g., a virtual environment is running on a system but the number of virtual machines is not defined.

Smets defines seven types of imprecise measurements with an error component. They are:

- *erroneous:* the data is simply wrong; e.g., 'the number of CPUs is 12' when the actual number is 23.
- *inaccurate:* the data is wrong, but the error is small; e.g., 'the number of CPUs is 12' when the actual number is 13.
- *invalid:* the data is wrong, and could lead to invalid conclusions; e.g., 'the number of CPUs is 12' when the actual number is 8, which could lead to the conclusion that 12 applications can be run on the system
- *distorted:* inaccurate and invalid data; e.g., 'the number of CPUs is 12' when the actual number is 6, which could lead to paying twice the required license fee for a software package.
- *biased:* all data has been subjected to a systematic error; e.g., the number of CPUs is recorded as 'number - 1'.
- *nonsensical:* the recorded value is so extreme that the user can easily identify the problem; e.g., 'the number of CPUs in a server is 1,000,000'.
- *meaningless:* the value has no meaning, and no importance; e.g., 'the OS reports an ID for a non-existent second processor'.

Inconsistency: Inconsistency can occur when several pieces of information are combined and aspects of imperfection appear. With inconsistency, there is always some type of error. The specific types of inconsistency are described below.

- *conflicting:* disagreement among the data; e.g., 'a system has 1 CPU but lists jobs as running on two'.
- *incoherent:* incoherent conclusions are reached as the result of conflicting data; e.g., (using the previous example) the peak CPU utilization is 200%.
- *inconsistent:* not compatible with other information (best used when time is involved); e.g., 'the system was rebooted at 3:00pm' 'it is now 3:15pm and the system has been up for 2 hours'.
- *confused:* a milder form of incoherence that can be recovered from small modifications of the

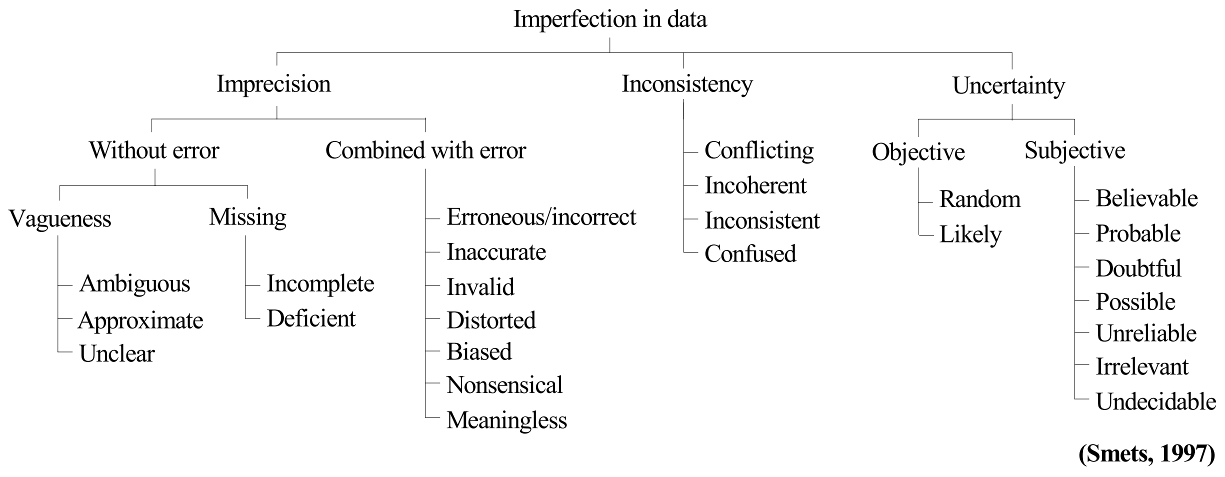

Figure 1. Smets' Taxonomy of Imperfections in Data

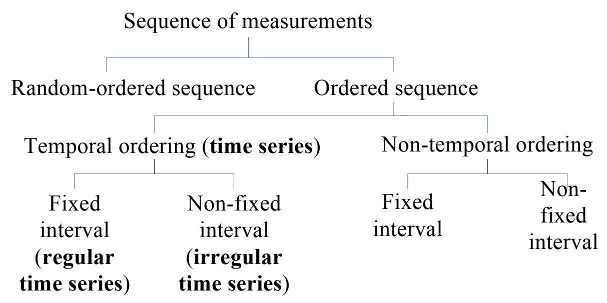

Figure 2. Types of Measurement Sequences

data; e.g., ‘the system was rebooted at 3:00pm’, ‘it is now 3:15pm and it has been up 16 minutes’.

4.2 Properties of Monitoring Data

One observation from 4.1 is that some of Smets' distinctions between specific classes are subjective. This provides more flexibility in applying the taxonomy to different spaces, but also requires agreement within a community about the definitions of imperfections for a specific space (e.g., computing systems). For this reason, we examine the imperfections that occur in a *time series* in the context of computer systems. We defer discussion of other measurement sequences for future work.

A time series is one of several measurement sequences, which may be categorized by how the measurements are ordered. Figure 2 shows one possible classification. A time series is a special type of ordered sequence, one in which the measurements are ordered using time (usually the time at which the measurements were taken). In a time series, the time stamps are typically monotonically increasing, and reflects a (partial) history of the environment under observation. If the interval between pairs of successive measurements in a time series is constant, we say the time series is *regular*; otherwise, it is *irregular*.[2]

More formally, a time series comprises one or more *data rows* (T_i, M_{1i}, M_{2i}, … , M_{ni}), where T_i is the time stamp for time interval i, and M_{1i}, M_{2i}, …, M_{ni} are the corresponding values of the desired metrics. Each M_{ji} value is called a *data point.*

2. *We are interested in comments regarding the existence of a more formal taxonomy describing sequences of measurements.*

4.3 Mapping Between Taxonomy and Data

In this section we attempt to map imperfections we have encountered in system utilization data to Smets’ Taxonomy. We provide an example for each specific type of imprecision and inconsistency. For selected types, we provide additional information on techniques that could be used to identify, or possibly address the imperfection, depending on how the data is to be used.

Ambiguous Data

In some computer systems, multiple agents are used to collect measurements of system utilization. Occasionally, these agents may use similar metric names (e.g., CPU utilization), but with different meanings. Without knowing which data collector was used (or even which version), it may be impossible to know how to properly interpret the value of the metric.

Approximate Data

Some data collectors may round metric values to a specified number of decimal places. This may result in a slight deviation from the true value.

Unclear Data

In an attempt to reduce the performance overhead of collecting data, collectors often aggregate data over longer intervals (e.g., five minutes). If a user requires maximum values at one minute intervals, they may only be able to ascertain that the maximum is at least as great as the maximum over the five minute data (which may only provide average values).

Incomplete Data

Data collectors provide values on a finite set of metrics. In some cases, this set may not include all of the metrics of interest to the user. However, occasionally it is possible to derive the desired metric from the collected data.

Deficient Data

A common problem we have observed with time series data (for system utilization) is missing data rows. Identifying missing rows is straightforward,

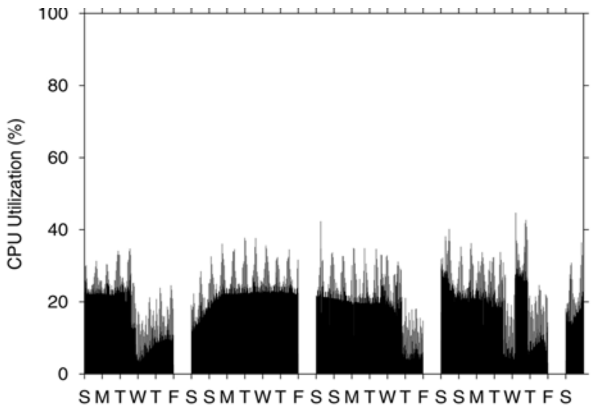

Figure 3. Example of Deficient Data.

assuming that the start time and the interval length of the data set are known. Identifying the cause of the missing data is more problematic, as there are multiple potential causes. For example, the system may have been powered off, or the data collector may have been turned off. Figure 3 indicates that other causes are possible. In this data set, there were no data rows on Fridays. Although we were unable to determine the cause of this problem, we suspect it was unrelated to the data collection.

There are also methods for 'repairing' deficient data in a time series. The timestamp for each data row can be calculated exactly, while the values for each metric could be approximated using any number of algorithms, such as linear interpolation. Obviously though, the usefulness of repairing this type of problem depends on how the data is to be used.

Erroneous Data

Within a data row, several types of incorrect data may occur. First, the timestamp (or interval length) for time *i* may deviate from the expected time. In a manner similar to the previous example, this deviation is easy to detect. Second, there may be errors with the metric values. As we will describe in subsequent examples, some of these errors can be detected with relative ease, while others are more difficult and perhaps even impossible to detect.

Inaccurate Data

We have observed this type of imperfection quite often, typically in the timestamps associated with a data row. In particular, the timestamp for a data row is often +/−1 second off the expected value. In this situation, it may be acceptable to the user to simply change the observed timestamp for the data row to the expected value.

Invalid Data

One of the more significant problems that can arise is one that leads to invalid conclusions regarding the specific scenario in which the data is being used. For example, in a capacity planning exercise, outliers in the metric values could suggest a significantly higher demand exists for resources than is actually the case. This could result in an overly conservative capacity plan, that leads to unnecessary purchases of equipment.

Distorted Data

This is a less significant case of the previous type of data imperfection. Use of distorted data may result, for example, in capacity plans that are slightly higher or slightly lower than the correct plan. Being able to identify distorted data may help users of the data to provide bounds on the accuracy of any conclusions drawn from the data; this could be extremely helpful in certain situations (e.g., customers who must compare the bids of several competing hardware suppliers).

Biased Data

This type of imperfection is quite common when using time series data from multiple computer systems. In particular, the clocks on each system may be skewed from the true (universal) time by some constant amount. This type of error may be difficult to detect, depending on how the data collector records the information. The best solution is to promote the systematic synchronization of all system clocks (e.g., using a protocol like `ntp`), to minimize the significance of such imperfections.

Nonsensical Data

One of the more common imperfections that we have detected in system utilization data are metric values that are "obviously" wrong. In some circumstances the nonsensical values are immediately obvious, as the reported values fall outside of a well defined range for the metric. In other cases, the nonsensical values only become obvious once they have been identified through other means. We explain this in more depth below.

Table 1 provides a list of metrics that we have seen in practice report nonsensical values. The first two metrics, CPU and memory utilization, were defined for the specific collector used to return a value between 0 and 100 percent. For well defined metrics such as these, it is straightforward to identify problematic values.

Table 1 Empirical Examples of Nonsensical Data

	Metric Name	Maximum Observed Value
1	*CPU utilization*	*>100%*
2	*memory utilization*	*>100%*
3	*CPU run queue length*	*327.67*
4	*disk queue length*	*327.67*
5	*memory queue length*	*327.67*
6	*page request rate*	*3276.7*
7	*pageout rate*	*3276.7*
8	*packet rate*	*6524830.0*

The values of the next five metrics in Table 1 (rows 3 to 7) were all observed in one data set. These values all appear erroneous, because (if we ignore the decimal place) they are all equivalent to 2^{15}-1, the maximum value of a 16 bit signed integer, the data type used for the metrics.

The final metric in Table 1 is the average packet rate per second. It is even less clear that the value shown for this metric is in fact erroneous. However, after identifying the network resources in the specific machine, we verified that it would be impossible for the machine to generate this rate of packets.

All of the nonsensical values shown in Table 1 could be identified by verifying the values fall within an acceptable range. The primary challenge is in identifying the range of values to use, as not all metrics have well defined ranges, and some actually vary from machine to machine, and collector to collector.

Meaningless Data

Imperfections that have no impact on a study that uses the data could be considered meaningless. For example, if a study only considers CPU utilization, the imperfections in the other seven metrics described in Table 1 could be considered meaningless, for that particular study.

Conflicting Data

Many data collectors of system utilization provide both summary statistics (e.g., of the system as a whole) as well as more detailed statistics (e.g., of individual system components). For example, a data collector may report the total number of CPU seconds used across all processors in the system, during a particular interval, as well as the number of CPU seconds used on each individual processor in the system. An example of conflicting data is when the sum of the values for each individual CPU does not match the reported value for all CPUs.

Incoherent Data

This type of imperfection is again dependent on the use of the data. If the data described in the previous example were used in an analysis, it could result in incoherent conclusions being made.

Inconsistent Data

When using time series (often the case with system utilization data), it is important to use timestamps in universal time (UTC). Timestamps left in local time can be affected by offset changes (i.e., due to daylight savings time) which may result in the time series appearing to jump backwards in time, and contain multiple data rows for the same interval of time.

Confused Data

Even when system clocks are synchronized, small deviations may still exist between them. When trying to compare data across machines, these small deviations may complicate the comparisons. It may be necessary, for example, to consider timestamps within *d* time units of one another to be equivalent, in order to work around this problem. Figure 4 provides an example of confused data. In this particular situation, we had data from two different systems, one running a RedHat Linux operating system, the other HP-UX. An error in the timezone

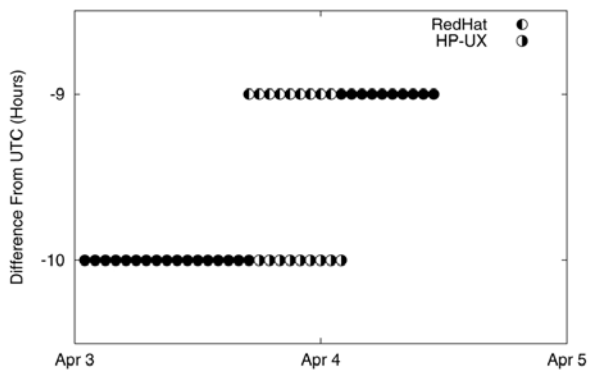

Figure 4. Example of Confused Data.

rules on the RedHat system caused it to adjust its offset nine hours too soon. During that nine hour period, errors could occur in any comparisons between the systems, as the universal times would differ by one hour.

4.4 Summary

In this section we have proposed a potential convergence of the uncertainty and systems communities, following the guidelines of Smets. We provided examples of imperfections we have encountered in system utilization data sets, and mapped them onto the taxonomy of imperfections proposed by Smets (see Figure 1). In the next section we discuss implications of our work.

5 Implications

The discussion in the preceding section points to the wide variety of problems that can occur in monitoring data and the subtle differences in how these problems give rise to data uncertainty.

One goal for a data assurance solution is for it to provide control and management systems with a quantitative measure of the quality of data. To derive this measure, the data assurance solution must associate values to the various factors from which uncertainty flows. How these values are derived is one of the issues we are currently addressing. We believe that there are three cases that must be taken into account. These cases relate to the uncertainty classification shown in Figure 1.

According to Smets, uncertainty flows from subjective and objective properties of the data. Subjective properties are linked to the observer's opinion about the true value of the data, while objective properties are linked to the information and the environment from which it was collected. Subjective properties flow, in turn, from humans, as software entities cannot form subjective opinions. Hence, one case reflects the need to capture human intuition in a programmatic way.

Objective properties may further be divided into "likely" properties, which result from some probably cause or known frequency distribution, and "random" properties, which result from something that is seemingly without purpose. The second case, which address likely properties, could be solved using algorithms to uncover trends in data. We do not presently know how to address random properties.

As we have already mentioned, our long-term vision involves enabling more automated and adaptive computing environments. We believe that data assurance will play a critical role. First, it will facilitate more repeatable and consistent analyses, by humans and automated control and management systems, thereby ensuring that a consistent set of conclusions are drawn from a single data set. Second, it improves the scalability of the 'acceptance' process, by enabling "experts" to examine larger data sets, and by sharing the combined intuitions of a "group of experts" with a much broader audience. Similarly, it benefits automated systems by freeing the developers from these systems from developing point data-assurance solutions. Third, identifying a more formal description of human intuition (applied to the computer systems space) will help overcome numerous trust barriers, a necessary step in activating automated systems.

We believe that there are numerous important research questions that must be solved before our long-term vision will be achieved. Some of these questions are:

- What mechanisms can be used to quantify uncertainty?
- What meta-data should be retained about the monitoring data to provide important contextual information?

- Is Smets' taxonomy sufficient for formally describing the types of imperfections in monitoring data?
- How should users formally describe the significance of different types of imperfections?
- What techniques should be used to identify and/or "repair" each type of imperfection?

We cannot answer all of these (and related) research questions on our own. In particular, we need help from the systems community to identify methods of expressing our collective intuition about the problems with monitoring data, to develop a system for passing relevant information between data assurance tools and data consumers in a programmatic manner, and for generalizing our techniques across other computer systems. From the uncertainty community, we require assistance on understanding the key attributes to focus on (e.g., decidability), and how we can relate the imperfections we identify and the significance levels specified by users to the levels of uncertainty defined by Smets. We also believe that there are other communities that could contribute to this space, and welcome any assistance they offer.

In addition to the tasks we are attempting to do, we believe it is important to point out what we are not trying to do. For example, we are not advocating that all processes involving computer systems should be automated. Instead, we are simply trying to assist in overcoming the obstacles for automating simple and often mundane yet important tasks that need to be done. We anticipate that this would enable people to spend additional time on more important tasks. Also, we envision that data assurance will occur at a layer between data collection and data consumption, i.e., use of that data. This offers several potential benefits. First, it allows for easy integration, as none of the existing tools need to be modified. Second, even though some of the imperfections identified are due to problems in the implementation of the data collectors, not all are, so the data collectors cannot provide all of the required assurance functionality.

6 Conclusions

IT compute environments are becoming increasingly large in size and complex in nature. As a result, these environments are difficult to manage. Automation is seen as a desirable feature for reducing the total cost of ownership for such environments. However, automation introduces additional risk, as humans are removed from the control loop. Without some notion of data assurance, obvious errors may be propagated at a much faster rate, affecting a much larger part of the environment.

To mitigate the addition risk, we are developing techniques to automatically identify errors with the data, correct some of them, and assign a quality rating to a data set. This rating will permit control systems to, for example, adjust their aggressiveness or decide to take no action. We are also exploring how to report such errors to humans without overwhelming them with details.

This is a large and challenging problem space. We invite others to critique our work, and to assist us in identifying relevant work that can help develop this important area.

Acknowledgments

We are indebted to Dejan Milojicic for reviewing the paper. His comments significantly improved the contents and presentation.

References

[1] Wilkes, J., Mogul, J., Suermondt, J., "Utilification," in the proceedings of the *ACM European SIGOPS Workshop*, September 2004.

[2] "Utility Computing," IBM Systems Journal special issue 43(1), 2004.

[3] Foster, I. et al., "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration.", 2002.

[4] Rolia, J., Cherkasova, L., Arlitt, M., and Andrzejak, A., "A Capacity Management Service for Resource Pools", Hewlett-Packard Laboratories Technical Report, HPL-2005-01, January 2005.

[5] Asset Optimization Group, Capacity Planner^{TM}, http://www.aogtech.com/CapacityPlanner.htm

[6] Xue, L., Zhu, X., Singhal, S., and Arlitt, M., "Adaptive Entitlement Control of Resrouce Containers on Shared Servers", to appear in the proceedings of the *9th IFIP/IEEE International Symposium on Integrated Network Management,* 2005.

[7] Keegan, J., *Intelligence in War*, Pimlico, London, UK, 2004.

[8] International Conference on Information Quality, 1996-2005.

[9] Smets, P., "Imperfect Information: Imprecision - Uncertainty", Universite Libre de Bruxelles, 1999.

[10] Driscoll, Patrick J. and Pohl, Edward, "Modeling the Decision Quality in Sensor-to-Shooter (STS) Networks for Unattended Ground Sensor Clusters", in the proceedings of the *Seventh International Conference on Information Quality*, 2002.

[11] Cohen, I., Goldszmidt, M., Kelly, T., Symons, J., Chase, J., "Correlating instrumentation data to system states: A building block for automated diagnosis and control", in the proceedings of *Operating System Design and Implementation (OSDI)*, 2004.