Specification Of Data Properties To Identify Anomalies In Scientific Sensor Data

Irbis Gallegos
University of Texas at El Paso, irbisle@utep.edu

Follow this and additional works at: https://digitalcommons.utep.edu/open_etd
Part of the Computer Sciences Commons, and the Environmental Sciences Commons

Recommended Citation
https://digitalcommons.utep.edu/open_etd/2285

This is brought to you for free and open access by DigitalCommons@UTEP. It has been accepted for inclusion in Open Access Theses & Dissertations by an authorized administrator of DigitalCommons@UTEP. For more information, please contact lweber@utep.edu.
SPECIFICATION OF DATA PROPERTIES TO IDENTIFY ANOMALIES IN
SCIENTIFIC SENSOR DATA

IRBIS J. GALLEGOS
Department of Computer Science

APPROVED:

Ann Q. Gates, Ph.D., Chair

Steve Roach, Ph.D.

Vladik Kreinovich, Ph.D.

Craig Tweedie, Ph.D.

Benjamin C. Flores, Ph.D.
Dean of the Graduate School
Copyright ©

by

Irbis Gallegos

2011
Dedication

A mis papás Ramón y Rosario,

a mi hermana Diana, y a mi sobrina Anaíd.

Gracias por su apoyo,

sin ustedes nada de esto seria possible.
SPECIFICATION OF DATA PROPERTIES TO DETECT ANOMALIES IN SCIENTIFIC DATA

by

IRBIS GALLEGOS, B.S

DISSEPTION

Presented to the Faculty of the Graduate School of The University of Texas at El Paso in Partial Fulfillment of the Requirements for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science
THE UNIVERSITY OF TEXAS AT EL PASO
December 2011
Acknowledgements

I would like to thank my committee members: D. Ann Q. Gates, Dr. Steven Roach; Dr. Vladik Kreinovich, and Dr. Craig E. Tweedie for their guidance and support to accomplish this work. In particular, I would like to thank Dr. Gates for believing in me and my work, for opening so many doors for me, and for showing me the way for becoming both a successful professional and a great individual. Thank you Dr. Gates for all of your hard work and for serving as an inspiration in my professional and personal life. I also thank Dr. Roach for his mentorship and honest comments he always has given me when I needed them. Dr. Roach’s mentorship helped me to leverage different options when I had to make important professional decisions. I am also thankful to Dr. Kreinovich for helping me build the communication bridge I needed to collaborate with scientists, and most importantly for being critical of my work that helped me identify the loopholes. A special thanks also goes to Dr. Craig Tweedie for giving me the opportunity to collaborate with him and his group and for showing me that there are so many things that I can do with my research that go beyond a cubicle. Thank you Dr. Tweedie for giving me the opportunity of visiting the Arctic and for believing in my work even though sometimes it seemed like it was too much out of the box.

I also thank Dr. Salamah Salamah for his honest support and mentoring through the years. I am also grateful to the professors of the Computer Science Department at the University of Texas at El Paso for giving me the scholarly foundations and most importantly for believing in my work. Thank you Dr. Martine Ceberio, Dr. Yoonsik Cheon; Dr. Luc Longpré, Dr. Patricia Teller, Dr. Francois Modave, and Dr. Nigel Ward for sharing your knowledge that prepared me for the qualifying exams. Thank you Dr. Paulo Pinheiro da Silva and Dr. Rodrigo Romero for your help in my research and for your invaluable input. A special thanks also goes to Dr. David Novick for encouraging me to pursue a Ph.D. and for always being supportive.

I would also like to thank those people who played a big role in helping me accomplish my work. I thank Santonu Goswani, Aline Jaimes, and the UTEP Systems Ecology Lab for helping me and for embracing me as part of their team—you all really made a difference. I would also like to thank Dr. John Gamon and Dr. Deana Pennington for their mentorship and support to address the different
scientific challenges. Of course, I also thank my biggest support group through the years; I thank all of the members of the PACES Lab and Cyber-ShARE Lab at which I spent lots of quality time. It was all of your support and jokes that got me through the PhD; thank you all.

I am also grateful to the administrative staff who through the years not only helped me with all the administrative issues, but also gave me valuable guidance and personal support. Special thanks go to Mary Contreras, Beatriz Tarango, and Sue Ann Walker. I also thank Dr. Chastity Bradford and Yvonne Lopez at Graduate School for their support through the toughest times; your support really made a difference. A very special thanks also goes to Elsa Q. Villa for being one of the very first persons who believed in me back when I was staring college. Thank you, Elsa.

I would also like to thank the University of Texas at El Paso (UTEP) and the many other institutions that supported me through the years through scholarships. I would like to thank the UTEP Dodson Dissertation fellowship, the Patricia and Jonathan Rogers scholarship, the UTEP Alliance for Graduate Education and the Professoriate (AGEP) fellowship, the National Science Foundation, the J. Lovett Endowed scholarship, and the Agilent/Hewlett Packard scholarship fund. Additionally I would like to thank the Sharing Resources to Advance Research and Education through Cyberinfrastructure (Cyber-ShARE) Center of Excellence, the Pan American Center for Earth and Environmental Studies (PACES), and the Computer Alliance of Hispanic-Serving Institutions (CAHSI) for funding and disseminating my research work.

Finally, but not least, I would like to thank my family and friends. I would like to thank my parents and my sister for their support and patience through the years and for allowing me to complete my PhD work through their personal and professional sacrifices. THANK YOU.
Abstract

Environmental scientists use advanced sensor technology such as meteorological towers, wireless sensor networks and robotic trams equipped with sensors to perform data collection at remote research sites. Because the amount of environmental sensor data acquired by such instruments is increasing, the ability to evaluate the accuracy of the data at collection time and to check that the instrumentation is operating correctly become critical in order to not lose valuable time and information. The goal of the research is to define a solution, based on software-engineering techniques, to support the scientist’s ability to specify data properties that can identify anomalies in scientific sensor data collected by instruments in remote locations.

The research effort included deriving a data property categorization from the findings of a literature survey of 15 projects that collected environmental data from sensors and a case study conducted in the Arctic. More than 500 published data properties were manually extracted and analyzed from the surveyed projects and the Arctic case study with scientists, who were collecting hyperspectral data using robotic tram systems. The data property categorization catalogs recurrent data patterns that have been used by scientists. The Specification and Pattern System (SPS) from the software-engineering community was used as a model to develop a system, called Data Specification and Pattern System (D-SPS), to define patterns and scopes of data properties based on the data property categorization. D-SPS provides the foundation for the Data Property Specification (DaProS) tool that can assist scientists in specification of sensor data properties.

A series of experiments were conducted in collaboration with experts working with Eddy covariance (EC) data from the Jornada Basin Experimental Range to determine if the approach for specifying data properties is effective for specifying data properties and identifying anomalies in sensor data. A complementary sensor data verification tool was developed to verify the expert-specified data properties over the EC data. The approach successfully identified and distinguished anomalies due to environmental variability events from anomalies due to equipment malfunctioning. In addition, this work also identified key factors that influence the effectiveness of the data anomaly detection process.
# Table of Contents

Acknowledgements.................................................................................................................. v

Abstract..................................................................................................................................... vii

Table of Contents..................................................................................................................... viii

List of Tables ............................................................................................................................. x

List of Figures ........................................................................................................................... xi

List of Illustrations .................................................................................................................. xii

Chapter 1: Introduction ............................................................................................................. 1
  1.1 Research Problem ............................................................................................................. 2
  1.2 Research Goal ................................................................................................................... 5
  1.3 Significance of the Research ........................................................................................... 5
  1.4 Broader Impacts ............................................................................................................... 6
  1.5 Organization of the Dissertation .................................................................................... 7

Chapter 2: Background ............................................................................................................ 8
  2.1 Overview ........................................................................................................................ 8
  2.2 Data Classification .......................................................................................................... 8
  2.3 Error Detection Approaches .......................................................................................... 10
  2.4 Property Specification Approaches ............................................................................... 16

Chapter 3: Data Property Categorization ............................................................................... 19
  3.1 Overview ........................................................................................................................ 19
  3.2 Data Property Categorization ......................................................................................... 19
  3.3 Data Property Categorization Findings .......................................................................... 27

Chapter 4: Data Specification and Pattern System ................................................................. 28
  4.1 Overview ........................................................................................................................ 28
  4.2 Data Property Specification and Pattern System (D-SPS) ............................................... 29
  4.3 Specification of Data Properties Using D-SPS ............................................................... 52

Chapter 5: Data Property Specification and Verification Tool Support .................................... 59
  5.1 Data Property Specification Tool Support Overview ....................................................... 59
  5.2 Data Property Specification (DaProS) Tool Interface ..................................................... 59
List of Tables

Table 1. D-SPS scope formal definitions and descriptions ......................................................... 30
Table 2. D-SPS Universality pattern formal definitions and descriptions ................................... 35
Table 3. D-SPS Absence pattern formal definitions and descriptions ......................................... 35
Table 4. D-SPS Existence pattern formal definitions and descriptions ....................................... 36
Table 5. D-SPS Precedence pattern formal definitions and descriptions .................................... 37
Table 6. D-SPS Response pattern formal definitions and descriptions ...................................... 40
Table 7. D-SPS Minimum Duration pattern formal definitions and descriptions ......................... 42
Table 8. D-SPS Maximum Duration pattern formal definitions and descriptions ....................... 44
Table 9. D-SPS Bounded Recurrence pattern formal definitions and descriptions .................... 45
Table 10. D-SPS Bounded Response pattern formal definitions and descriptions ...................... 46
Table 11. D-SPS Bounded Invariance pattern formal definitions and descriptions ..................... 48
Table 12. Extended Backus-Naur Form syntax description used to build D-SPS properties ........... 52
Table 13. High-level D-SPS specification for a dry-bulb temperature property .......................... 55
Table 14. High-level D-SPS specification for a diagnostic sensor property ............................... 56
Table 15. DaProS DNL grammar for validating property representations ................................. 68
Table 16. DaProS disciplined natural language representation for an experimental reading property ... 70
Table 17. DaProS disciplined natural language representation for a data relationship property ........ 70
Table 18. Data properties that could not be specified using DaProS ......................................... 72
Table 19. Tolerance ranges used in DART to assess hyperspectral data .................................. 83
Table 20. Tram data assessment results for the 2008 season ...................................................... 87
Table 21. Sensors included in the Jornada Eddy covariance tower used for the experiment ......... 89
Table 22. Data properties used in the case study, with property types and descriptions ............... 92
Table 23. Data properties used in the experiment to verify Eddy Covariance data using SDVe .... 95
Table 24. CSAT diagnostic flag definitions and data properties used to capture the flags .......... 107
Table 25. Agc confidence levels for specified data properties ................................................... 108
Table A1. Active database systems operators summary ............................................................ 119
Table B1. SPS scope descriptions .............................................................................................. 121
Table B2. SPS pattern descriptions ............................................................................................ 122
Table C1. Composite propositions definitions ........................................................................... 123
Table E1. Eddy covariance total experiment results ................................................................. 135
Table E2. Total number of anomalies as identified by data properties ...................................... 136
Table E3. Total number of anomalies as identified by data categories ...................................... 138
Table E4. Results of using SDVE to identify anomalies in EC winter sensor data ................. 138
Table E5. Number of anomalies as identified by data properties for winter season ................. 140
Table E6. Number of anomalies as identified by data categories for the winter season ......... 141
Table E7. Results of using SDVE to identify anomalies in EC Summer sensor data ............... 142
Table E8. Number of anomalies as identified by data properties for the summer season ......... 143
Table E9. Number of anomalies as identified by data categories for the summer season ......... 143
List of Figures

Figure 1. Groupings of data checks and analysis using the terminology of the projects.........................20
Figure 2. Scientific sensor data properties categorization .................................................................21
Figure 3. Property categorization results ..........................................................................................25
Figure 4. Experimental readings properties categories distributions ...............................................26
Figure 5. Experimental conditions properties categories distributions .............................................26
Figure 6. The relationship between the categorization, patterns and Boolean statements...................33
Figure 7. D-SPS specification derivation tree for a dry-bulb temperature property ..........................55
Figure 8. Pseudo-code representation for a dry-bulb temperature property ......................................56
Figure 9. D-SPS specification derivation tree for a diagnostic property ...........................................57
Figure 10. Pseudo-code representation for a dry-bulb temperature property with diagnostic flags ......58
Figure 11. Data property category decision tree for experimental readings ........................................63
Figure 12. Data property category decision tree for experimental condition ......................................64
Figure 13. Disciplined natural language specification derivation tree for a datum property ...............70
Figure 14. Disciplined natural language specification derivation tree for a data relationship property .71
Figure 15. Verification summary file snapshot ..................................................................................76
Figure 16. Verification file snapshot ...............................................................................................77
Figure 17. Typical reflectance graph as used by environmental scientists ...........................................79
Figure 18. Data Assessment Run-time (DART) framework ...............................................................82
Figure 19. DART data assessment results distribution for North tramline for the 2008 season ..........86
Figure 20. DART data assessment results distribution for Central tramline for the 2008 season ......86
Figure 21. DART data assessment results distribution for South tramline for the 2008 season .......86
Figure 22. Ratio of total data checks to total anomalies founds for the winter season ........................96
Figure 23. Ratio of total data checks to total anomalies founds for the summer season .....................97
Figure 24. Ratio of total anomalies by data property category for the winter season ..........................98
Figure 25. Ratio of total anomalies by data property category for the summer season .....................99
Figure 26. Number of hourly data properties violations for the winter season ..................................100
Figure 27. Number of hourly data properties violations for the summer season ...............................101
Figure 28. Rain event captured as abrupt changes in anomaly trend lines during the winter season ....102
Figure 29. Reflection of trend lines due to conflicting properties .......................................................105
Figure 30. Characterization of a rain event as a graphical representation .........................................106
Figure 31. Cyclic diurnal behavior captured by specified data properties ..........................................109
Figure E1. Total distribution of experimental loading time and distribution time ................................135
Figure E2. Ratio of total data evaluations to total anomalies found ..................................................136
Figure E3. Ratio of total anomalies by data property category .........................................................138
Figure E4. Data property contribution to detection of data anomalies ..............................................139
Figure E5. Distribution of experimental loading time and distribution time for the winter season ......140
Figure E6. Daily ratio of total loading times to verification times for the winter season .................140
Figure E7. Distribution of experimental loading time and distribution time for the summer season ....142
Figure E8. Daily ratio of total loading times to verification times for the summer season ...............142
## List of Illustrations

**Illustration 1:** DaProS graphical user interface ................................................................. 60
**Illustration 2:** DaProS main property specification graphical user interface ........................ 61
**Illustration 3:** DaProS property selection oracle ................................................................... 62
**Illustration 4:** DART graphical user interface ....................................................................... 84
Chapter 1: Introduction

Current trends and fluctuations in the Earth’s climate have resulted in an increased focus by scientists to study changes in environmental conditions to better understand climate change and associated impacts. Scientists use advanced sensor technology such as meteorological towers, wireless sensor networks, and robotic trams equipped with sensors to collect data at remote research sites. As the amount of data collection instruments introduced in the field increases, so does the volume of environmental sensor data acquired by such instruments. The measurements taken by sensor are discrete samples of physical phenomena and are subject to review of their accuracy dependent on their location. For some scientific projects, instrumentation typically does not include mechanisms to detect anomalies as data are collected. In this context, an anomaly is a deviation from an expected sensor data value. An anomaly in data does not always represent errors: it may represent environmental variability requiring further analysis. A common practice for environmental scientists is to collect sensor data for extended periods of time, possibly creating numerous individual data files that might not be verified for assurance that the data adhere to predefined data quality standards until the files are transferred to the database. A challenge with this practice is that a large amount of incorrect data can be collected due, for example, to a faulty sensor, and this can be undetected for extended periods of time. Furthermore, if data are identified as incorrect during analysis at a much later time, the data gathering process may have to be repeated.

Repeating data collection is expensive especially when the site is at a remote location. Sensor technology needs to be redeployed and possibly recalibrated; the amount of time required to gather the data can be significant. For time sensitive data that is required for policy decision-making, it might not be even possible to repeat the data gathering process. Because environmental sensor data can be non-reproducible entities, i.e. the observations at a given time and set of conditions cannot be repeated, the knowledge that could have been obtained from the correct data is not captured.

The importance of data for the study of the environment and their value to society supports the need to develop mechanisms and procedures to verify the integrity of the data at near-real time or post-collection time. Corrupted sensor data can cause miscalculations that might have a major impact on the
interpretation of results. For example, the team studying the data obtained from the Ozone Mapping Spectrometer (TOMS) on board of the Nimbus-7 satellite failed to detect the Earth’s stratospheric ozone depletion in some areas [1]. The team failure occurred because the TOMS data analysis software had been programmed to flag and set aside data points that deviated greatly from expected measurements. As a result, the Spring-time measurements showing a 10% drop in ozone concentrations, which should have set off alarms, were overlooked and marked as errors. In another example, the U.S National Snow and Ice Data Center project underestimated for several weeks the extent of Arctic sea ice by five hundred thousand square kilometers due to an undetected sensor drift [2]. The sensor drift went undetected because complete checks on the data had not been carried out. Also, a recent study to determine the quality of temperature measurements, which were obtained from the 1,221 continental United States climate-monitoring stations overseen by the National Weather Service, showed that only 11 percent of surveyed stations are of acceptable quality [3]. Furthermore, scientists measuring net radiation of Earth’s atmosphere to determine how much energy remains in the Earth system concluded that even though existing observing systems can capture all the required environment variables, it is still not possible to obtain closure of the energy budget due to either inadequate measurement accuracy or inadequate data processing [4].

1.1 Research Problem

In environmental studies, the data collection process can be divided into three phases: the instrument-calibration phase, the data-gathering phase, and the post-processing phase. In the instrument-calibration phase, the scientist may calibrate the data-sensing equipment based on knowledge about the site being studied. Anomalies may be introduced into the collected data. Some causes of anomalies in sensor data include noise from external sources, hardware noise, inaccuracies and impressions in sampling methods and derived data, and various environmental effects [5]. In the data-gathering stage the scientist collects the data using calibrated equipment; at this stage, scientists typically rely on their knowledge and experience in the field to manually identify and distinguish one type of anomaly from another given the equipment and prevailing climatic conditions. Some limitations
to this approach exist. The following examples illustrate the difficulty of making decisions in the field concerning the quality of the sensor data being collected [6]:

- The data may not be readily available for analysis and interpretation from the electronic device recording the data over time, i.e., from the data logger.
- Problems with the equipment, such as low battery voltage, extreme differences between the temperature of the instrument and the external temperature, and dark current drifts make it difficult to distinguish errors from the data themselves unless the problem has previously occurred and the origin of the problem has been properly documented in the past.
- The scientist or technician in the field may not have the depth of knowledge or experience to identify potential problems.
- As the complexity of the equipment increases, so does the difficulty to determine the cause of equipment malfunction.

In the *post-processing* stage, data are generally processed and stored in a database by the scientist. In this stage, scientists typically define and use site-specific anomaly detection processes to identify and differentiate anomalies in the processed data. Even though anomaly detection in sensor data is important to draw reliable environmental conclusions, scientists rarely share or reuse knowledge about their data processes with other scientists until a process is established by the scientific community mostly because of the lack of a well-defined methodology for doing so and lack of tool support for such sharing.

The challenges that contribute to the limited sharing of scientific data processes and knowledge include [6]:

- The lack of documented properties and associated information, e.g., statements about who contributed the property and the applicability of the property. Properties of interest include checks on the following: upper and lower limits of variables; limit on the rate of change between data; detection of subsequence of data with the same value; analysis of two or several parameters at the same point in time; spatial continuity or consistency checks in which the values of adjacent
stations are allowed to differ within a certain bound; and diagnostic equations to which data are expected to adhere;

• Differences in checking criteria and processes across the scientific community;
• Ambiguity in natural languages when describing properties;
• Complexity in properties when dealing with time and multiple criteria;
• Technical knowledge required by scientists, especially when dealing with relationships among properties of various types and issues related to data repositories; and
• Use of embedded or hard-coded property checking in many existing systems, making it difficult to reuse and refine properties.

Also, data properties’ requirements for anomaly detection at collection time and post-processing are distinct. Data properties for the \textit{data-gathering} stage address behaviors associated with collection-time measurements and equipment malfunction. These properties are frequently used to flag issues that may require immediate action from an expert. The advantage of collection time anomaly detection is that corrective action can be taken immediately after an anomaly is found, preventing anomaly propagation through time or with data reduction, and thereby reducing the need to repeat the experiment. The disadvantages are the difficulty to manually analyze large amounts of data as they are collected, the inability of practitioners to have a global view of the dataset being analyzed due to lack of tool support, and the addition of a new process that has to be incorporated into the scientist’s daily activities.

Data properties for the \textit{post-processing} stage rely on historical data and trends, as well as statistical analysis. The advantage of such an approach is that it provides a larger overview of the data behavior and allows scientists to identify anomalies at a higher level. The disadvantages are as follows:

• Anomalies can be masked and propagated because of the practice of condensing datasets with a large number of samples through data reduction;
• Modification to raw data performed during the processing and analysis stage can accidentally introduce new anomalies into a dataset;
• Custom data anomaly detection software that has not been verified may introduce anomalies;
• Corrective action cannot be taken if anomalies are found, possibly requiring that data collection be repeated.

1.2 Research Goal

The process of detecting anomalies in scientific sensor data is divided into two stages, a property specification stage and a verification stage. In the property specification stage, a scientist specifies a set of properties to be used to detect anomalies in the data. In the verification stage, a system checks that the data adheres to the specified properties. Indeed, the quality of data verification can be affected by the quality of the properties specified. The primary focus of this dissertation is data property specification for collection time data.

The overarching research goal of this dissertation is to define an environmental-scientist-centered approach for specifying data properties that defines temporal and data relationships associated with sensor data to identify anomalies. The research questions are as follows:

• What types of data properties are commonly used to check anomalies in sensor data collected by environmental scientists?
• What factors must be considered for properties related to time and location?
• What are the differences between properties that check experimental conditions and those that check instrumentation?
• How can a specification-pattern-system approach to data property specification be applied?
• How effective are scientist-specified properties at identifying anomalies?

1.3 Significance of the Research

The importance of data to the study of environmental sciences emphasizes the need to develop mechanisms and processes to verify the integrity of the data obtained by field-based instrumentation. As part of the process, scientists need to: document the data anomaly detection processes used to verify the collected data, be able to troubleshoot equipment at collection time as it becomes dysfunctional, be able to recognize environmental variability events, and train new scientists to identify and analyze anomalies in data.
The proposed approach will allow scientists or a group of scientists to specify and customize properties for anomaly detecting, based on the environment, instrument, and available knowledge. The proposed approach will allow an environmental scientist to specify data properties that can be used as input to data verification tools that can check for anomalies as the data are collected. The work has the potential to impact a scientist’s ability to provide and improve anomaly-detection in data for field-based instruments used in environment science research.

The approach will allow scientists to share, reuse, and adapt knowledge captured as data properties to detect the anomalies in datasets. Once a data-property set is built and validated, the properties can be shared with the broader scientific community in a form that can be interpreted by data verification mechanisms. In some cases, the property may require modification, e.g., a change in threshold due to the location of the research site being different to the site used to generate the property. Metadata may be associated with the data properties to document the reasoning used to construct the property and the references used as the basis of the property.

A challenge in anomaly detection is determining if an anomaly occurred because of an error caused by the equipment or by an environmental event. A scientist can use his or her expertise and knowledge about a site and the instrumentation to combine data properties to characterize the causes for an anomaly. For example, scientists can define properties that compare the readings collected by redundant sensors to determine if both sensors have sensed an environmental event or if the reading was a product of a faulty sensor.

1.4 Broader Impacts

The approach presented in this dissertation pertains to any application that requires anomaly detection for large amounts of sensor data and variable properties. In addition, the proposed work can be used as an educational tool to assist students as they develop skills to interpret data and to identify and specify data properties. As an educational tool, it can be used to emphasize the notion of data assessment and its importance in determining the quality of results in other disciplines such as the geological sciences and engineering. Such an educational tool can be used to train new scientists to evaluate data
integrity in different data quality scenarios. It may be possible to look at historical data sets to determine thresholds, maxima, minima and variability in data that may help build data properties.

1.5 ORGANIZATION OF THE DISSERTATION

This dissertation is organized as follows. Chapter 2 provides the background of work that supports the research. The chapter discusses early work conducted to identify anomalies in data on various domains. It also includes a discussion of active databases (aDBS), the Specification and Pattern System (SPS), property specification tools based on the SPS, run-time monitoring, data categorizations based on classes of queries related to sensor networks, computational data analysis for error detection, and scientific categorizations of domain-specific errors.

Chapter 3 focuses on data properties related to environmental science studies. Chapter 3 contains a literature survey conducted to understand the causes of anomalies in data related to environmental science, in particular those obtained through sensors. The results from this literature survey were used to develop a data property categorization.

Chapter 4 presents the Data Specification and Pattern System (D-SPS). In addition, it discusses the Data Property Specification (DaProS) prototype tool developed based on the data property categorization and the D-SPS.

Chapter 5 presents a case study conducted in the Arctic to elicit the types of data properties of interest to environmental scientists and document how these properties can be captured. The chapter also describes how the Sensor Data Verification (SDVe) prototype tool was used to verify data properties specified using the DaProS tool on eddy covariance sensor data obtained from the Jornada Experimental Range research site.

The dissertation ends with a summary and discussion of future work, followed by references and five appendixes.
Chapter 2: Background

2.1 OVERVIEW

The specification of data properties to identify anomalies in sensor data requires an examination of efforts in the following areas: categorization of different aspects of data collected from ecological and environmental studies, error detection approaches, and property specification techniques. Specifically, the chapter examines efforts in the sensor network community to categorize data queries and uncertainty. It also presents work on detection of error anomalies from the viewpoint of databases, software engineering monitoring systems, and computational data analysis. The chapter ends with a summary of work to support the specification of properties that can be used to detect errors in data.

2.2 DATA CLASSIFICATION

In the ecology and environmental science communities, there have been numerous efforts to classify sources of errors in data. Two classification systems are described in this section. The first classification system groups the types of queries that are made to examine data collected from sensor networks. The second classification categorizes uncertainty in data collected for ecological studies.

2.2.1 Data Query Classification

Elnahrawy and Nath [5] categorized data properties into four categories. Single Source Queries properties return the value(s) of the attribute(s) of a specific sensor and no aggregation is involved. Sent Non-Aggregate Queries properties return the set of sensors that satisfy a given user-defined predicate. The predicates are assumed to be simple range queries on one or more attributes and are allowed to include AND and OR operations. Summary Aggregate Queries are queries performed using one of the following aggregate functions: SUM, COUNT, and AVG. Exemplary Aggregate Queries are queries performed using one of the following aggregate functions: MIN and MAX. This approach is based only on sensor data and does not capture properties to detect instrument malfunctioning.

Bonnet et al. [7] suggest classifying the queries as historical, snapshot and long-running queries. Historical queries aggregate queries over historical data obtained from the device network. Snapshot queries relates only to those measurements at a given point in time. Long-running queries relates to data collected from the device network over recurrent time interval. In this approach, queries are formulated
in Structured Query Language (SQL) with minimal additions to the language. This approach is tied to a specific distributed query-processing model not available for all wireless sensor networks.

Madden et al. [8] classify data aggregates according to their state requirements, tolerance of loss, duplicate sensitivity, and monotonicity. Duplicate sensitivity implies restrictions on network properties and on certain optimizations. Exemplary aggregates return one or more representative values from the set of all values, and summary aggregates check whether a certain property is satisfied for all the measurement values. Monotonic aggregates are used to determine whether some properties can be evaluated in the network before the final sensor value aggregate is known. Finally, the state requirements refer to the amount of space required to store partial aggregate states. The classification is tailored to match sensor networks properties.

In their work, Cheng et al. [9] present a classification of probabilistic queries. The authors identify two dimensions for classifying database queries, by nature of the answer or by aggregation. Value-based Non-Aggregate queries return an attribute value of an object as the only answer and involve no aggregate operators. Entity-based Non-Aggregate queries return a set of objects, each of which satisfies the condition(s) of the query, independent of other objects. Entity-based aggregate queries return a set of objects that satisfy an aggregate condition. Value-based aggregate queries involve aggregate operators that return a single value. This approach requires a deep understanding of the data and the different probabilistic measurements associated with the data.

2.2.2 Scientific Uncertainty Classification

Many scientific communities focus on understanding the sources and effects associated with errors in collected data. In ecology, a classification [10] divided uncertainty into two main categories: epistemic uncertainty and linguistic uncertainty.

Epistemic uncertainty is associated with knowledge of the state of a system, and it includes uncertainty due to limitation of measurement devices, insufficient data, extrapolations and interpolations, and variability over time or space. Epistemic uncertainty can be classified into six main types: measurement errors, systematic errors, natural variations, inherent randomness, model uncertainty, and subjective judgment.
Measurement errors result from imperfections in measuring equipment and observational techniques and include operator error and instrument error. Measurement errors manifest as random variations in the measurement of a quantity and are influenced by the number of measurements taken, the variation among measurements, the accuracy of the equipment used to take the measurements, and the training and skill of the observers [10]. Systematic errors occur as a result of a bias in the measuring equipment associated with a given sampling procedure. Systematic errors result from the deliberate judgment of a scientist to exclude or include data that are should not the excluded or included, unintentional erroneous calibration of measuring equipment or consistent incorrect recording of measurements. Natural variations occur in systems that change in a way that are difficult to predict with respect to time, space, or other independent variables. Inherent randomness occurs when a system is irreducible to a deterministic state, thus limiting the understanding of the system. Model uncertainty results from a misrepresentation of physical and biological systems due to variables and processes under-representation. Subjective judgment uncertainty is the result of a misinterpretation of data, especially when data are scarce and error prone.

Linguistic uncertainty is associated with the natural language used in the scientific vocabulary, and can be classified as: under-specificity, ambiguity, vagueness, context dependence, and theoretical indeterminacy of theoretical terms. Under-specificity occurs when there is unwanted generality in a statement, i.e., the statement does not provide the degree of details necessary. Ambiguity occurs when a word can have more than one meaning and it is not clear which meaning is intended. Vagueness occurs when the borderline cases for a word are imprecise. Context dependence arises from a failure to specify the context in which a proposition is to be understood. Indeterminacy of theoretical terms relates to the problem that the future usage of theoretical terms is not completely fixed by past usage, so some existing theoretical terms, although not ambiguous, have the potential for ambiguity.

2.3 ERROR DETECTION APPROACHES

2.3.1 Active Databases

Event-Condition-Action (ECA) systems are frequently used to detect errors in data. ECA systems evaluate rules that satisfy the following semantics: when an event occurs, evaluate associated
conditions, and for each satisfied condition execute an associated action [11]. Active databases (aDBS) [11] are the most common ECA system.

In active databases, the database content and context is monitored for changes in data entities of interest (events). Once an event is raised, conditions about the data entity are evaluated. If any condition is satisfied, an action on the data entity itself, the database data, or the database state is executed. Supported formalisms for active databases are categorized based on the tool for what they were developed. The major active database systems are described next.

2.3.1.1 Swiss Active Mechanism-Based Object-Oriented Database System

The Swiss Active Mechanism-Based Object-Oriented Database System (SAMOS) [12] is a combination of active and object-oriented database systems. It is based on the principles of aDBS: rule specification, rule execution, and rule management. In the SAMOS system, the user defines event patterns using a high level specification language, then the system translates the event definitions, stores the events into the event-rule-base and “programs” the event detector. The event detector identifies occurrences of primitive events, builds the internal representation of composite events, inserts occurring “event of interest” in the event register, and finds rule(s) and executes them. SAMOS introduces constructs for the specification of events based on an event algebra, integrates time specification facilities into event definitions, uses an object-oriented environment for the internal representation of events, and uses Petri nets to detect event occurrences.

In SAMOS, events are divided into two types, primitive events and composite events, both of which can be timed. A primitive event describes a point in time specified by an occurrence in the database (method events), the DBMS (transaction events), or in its environment (time and abstract events). A composite event is used to model complex occurrences as events. Six event constructors are used to specify composite events.

Timed events describe events as an explicit point in time that can be specified as absolute or as periodically reappearing. SAMOS uses method events to capture the points in time when the object begins, and when the object finishes the execution of the method.
2.3.3.2 Ode

Ode [13] is an object oriented (OO) database developed by AT&T Bell Labs that is based on O++. O++ extends C++ by facilitating the creation of persistent objects. O++ is the programming language for the Ode object database. In Ode, the conditions are part of the event specification and triggers are used to indicate the presence of events.

In Ode events are “happenings” of interest. Events happen instantaneously at specific point in time. In object-oriented databases, events are related to actions that happen to objects and the state of the objects. Events have “scope”. In OO systems, most events are local to a particular object. Ode supports three types of events: Basic, Logical and Composite.

Basic events include object state events that capture the creation, deletion, update/read/access of an object, method execution events that capture specified member function applied to objects, time events that assume a global system clock where each tick of the clock is an event treated as a local event at each object in the database, and transaction events that capture the start, commit attempts, commit, abort of a transaction.

In Ode, every basic event is a logical event that is used to hide or “mask” some occurrences of an event. Logical events are combined to create composite events using logical operators and special event specification operators. In composite events, the events are said to occur at the point of occurrence of the last logical event that was needed to make it occur. An optional mask predicate can be applied to a composite event to obtain a logical-composite event. A logical-composite event is the general notion of an event. Any mask predicate applied to a composite event, unlike the mask predicates of logical events, can only be evaluated in terms of the “current” state of the database.

2.3.3.3 Snoop

Snoop [13] is a model independent event specification language used to specify ECA properties. Snoop distinguishes between events and conditions, classifies the types of events required, identifies primitive events, and introduces event modifiers and operators for constructing composite, complex and abstract event expressions. Snoop supports temporal, explicit, and composite events in addition to traditional database events. Snoop distinguishes between event expressions and events. An event
expression is an expression that defines an interval on a timeline. Event expressions are used to denote the occurrence of an event within a closed interval. An event is an atomic occurrence.

In Snoop, events can be either logical or physical. Logical events correspond to the specification of an event at the conceptual level and do not specify the physical or internal event to which it will be mapped. Events at the conceptual levels are mapped to the physical/internal events either by using a mapping specification or by choosing an implementation that corresponds to mapping depending upon whether the mapping of the event is done by a compiler or manually by the practititioner. Logical events are mapped uniquely to physical events whereas a physical event may correspond to one or more logical events. Physical events are the ones actually detected by the system. Events may precede or follow one another, or events may be unrelated. Snoop establishes ordering relationships to capture behavior.

Events are organized in the basis of structure and behavior. Events can be broadly classified into: primitive events, events that are pre-defined in the system using primitive event expressions and event modifiers; and composite or complex events, events formed by applying a set of operators to primitive and composite events.

Primitive events are further classified into database events, temporal events, and explicit events. Database events are related to database operations. Temporal events are related to time and are of two types: absolute and relative. Absolute events map to discrete points along the timeline. Relative events are defined with respect to an explicit reference point. Explicit events are events that are detected and signaled along with their parameters by application programs and are only managed by the system. Prior to their usage, explicit events need to be registered with the system. For complex applications, composite events can be used to specify ECA rules. A composite event is an event obtained by the application of an event modifier to a complex event expression.

Conditions are also part of Snoop specifications. A condition is a Boolean function of data values. A condition does not produce any side effects. A condition may be valid over an interval of time, while an event is atomic by definition. A database state of interest can be defined in terms of a condition; conversely, being in a state is a condition. Conditions define ‘states’ and hence are used in
ECA rules as guards on transitions. A guarded transition fires when its event occurs but only if the guard condition is also true.

2.3.3.4 High Performance Active Database System

The High Performance Active (HiPAC) database system [15] addresses the management of time-constrained data. In HiPAC “triggering” events cause conditions to be evaluated over a predefined subset of data. HiPAC supports selection conditions that evaluate single attributes of a single object, restriction conditions for multiple attributes of a single object, aggregation conditions for single attributes aggregated over multiple instances, join conditions for single common attributes of multiple homogeneous objects, i.e., sensor correlation, or several attributes of heterogeneous objects, and application-specific operators such as time normalization or interpolation of data used in condition evaluation. Specific temporal constructs in HiPAC include events that recur every predefined amount of time, timing constraints to indicate hard time deadlines of tasks, task urgencies to order tasks based on time, transition conditions to verify time-dependent post-conditions of data entities, and historical predicates to check time-dependent data trends.

For a summary of active databases operators and their associated semantics for the formalisms mentioned in this section refer to Appendix A.

2.3.4 Computational Data Analysis

Numerical approaches to identify errors in data are frequently used. Such numerical approaches usually rely on statistical analysis to identify errors and trends in the data, especially on time series data. The following subsection describes some of these approaches.

2.3.4.1 Quality Control on Time Series Data

The Intelligent Outlier Detection Algorithm (IODA) [16, 17] is a patented technique designed to perform quality control on time series data. IODA was developed at the National Center for Atmospheric Research (NCAR). The algorithm uses statistics, graph theory, image processing and decision trees to determine whether the data are correct. The algorithm was implemented in MATLAB and may be expanded and translated to other programming languages such as C. The IODA algorithm
compares incoming data, which are treated as images, to common patterns of failure. The algorithm identifies problems with data points such as jumps and intermittency, identifies patterns in the data, and characterizes data inaccuracies. The algorithm arranges data points into clusters, both in a domain that represents the data points over time and in a delay space. The delay space is a technique that pairs a data point in the times series with the previous value. Using the clustering technique, bad data are separated into their own cluster, clearly distinct from the cluster of accurate data. The IODA algorithm also calculates quality scores indicating whether each individual data point is good or bad. The algorithm was successfully applied to wind readings from anemometers in Alaska that contained errors.

### 2.3.4.2 Dynamic Bayesian Network

Dereszynski & Dietterich [18] use a Dynamic Bayesian Network (DBN) [19] approach to automatic data cleaning for individual air temperature data streams. The DBN combines discrete and conditional linear-Gaussian random variables to model the air temperature as a function of diurnal, seasonal, and local trend effects. The approach uses a general fault model to classify simple anomalies (observations outside the range of acceptable temperatures), medium anomalies (malfunctions in the sensor hardware or changes in functionality of the sensor), and complex anomalies (anomalies that are so subtle that they cannot be captured without the use of additional sensors).

### 2.3.5 Model Checkers and Run-Time Monitoring

For critical software systems, software assurance mechanisms, such as model checkers and run-time monitoring systems, can be used to check that the data being processed by software is correct with respect to a set of properties. Model checking [20] is a formal technique for verifying finite-state concurrent systems and relies on building a finite model of a system and an algorithm that automatically traverses the system model to verify if a desired property (or a set of properties) holds in the model.

Run-time monitoring allows practitioners to observe a software execution trace and check that a software property specification holds over the given trace [22]. In most run-time monitoring frameworks, the application’s code is instrumented by injecting checks at appropriate places in the source code. Usually, the application to be monitored has to be recompiled to include the checks, and
monitoring is performed at the code level to ensure that the code is behaving as intended. Delgado et al. [22] have compiled a taxonomy of run-time monitoring frameworks.

Model checkers and runtime monitoring require the specification of properties using a formal language that is well defined syntactically and semantically. Once the properties are specified, they are used as input to software assurance mechanisms that can verify if the software execution adheres to the expected behavior provided by the specified properties.

2.4 Property Specification Approaches

While formal languages eliminate the inherent ambiguity of natural languages, they introduce the following challenges: the need for practitioners to have strong mathematical or software engineering background, the language expressiveness that may limit the types of properties that can be specified, the difficulty to understand and validate properties specified in a formal language. Tool support has been created to guide and help practitioners overcome the challenges associated with the use of formal languages to specify software properties. Some of these tools are described in this subsection.

2.4.1 Specification and Pattern System

Dwyer et al. [23] introduced the Specification and Pattern System (SPS) to assist practitioners to formally specify software and hardware properties. The authors of the system analyzed a wide range of properties from multiple domains, i.e., hardware systems, network protocols, security protocols, and user interfaces, to identify common patterns. Currently, the SPS is available in a website maintained by the Department of Computer Information Systems at Kansas State University [23]. The website provides a repository of all raw survey data, i.e. properties, that led to the creation of the patterns and scopes used in the SPS. The SPS supports the translation of a property specification to several formalisms that capture discrete-time systems [24].

In the SPS, a specification consists of scopes and patterns. Scopes define the portion of a program over which a property holds. Patterns describe the structure of specific behaviors and define relationships among patterns. Propositions are used to represent Boolean expressions. There are five types of scopes defined in the SPS: Global, Before R, After Q, Between Q and R, and After Q until R. Appendix B further describes each scope.
There are seven types of patterns defined in SPS. The patterns are divided into two groups: Occurrence and Order. Occurrence patterns deal with single event or condition and specify the rate at which that condition or event occurs. The Occurrence patterns are Absence, Existence, and Universality. Order patterns relate two conditions or events and specify the order in which they occur. The Order patterns are Response, Precedence, Chain Response, and Chain Precedence. Appendix A1 provides a short description of each pattern.

The notion of Composite Proposition (CP) was introduced by Mondragon et al. [25] as an extension to SPS to support the specification of properties requiring sequential and concurrent behavior. CPs, which are composed of conditions and events, extend the type of properties that can be specified and generated in a formal language. Conditions are propositions that hold in one or more consecutive states. Events are instants at which a proposition changes values in two consecutive states. A CP defined as a condition is used to describe concurrency, while one defined as an event is used to describe activation or synchronization of processes or actions. CPs can define boundaries on scopes and patterns with multiple propositions. Salamah et. al. [26] used CP classes with patterns and scope to generate templates that assist in deriving specifications in Linear Temporal Logic (LTL) [27]. Appendix C provides a short description of each CP class.

2.4.2 Property Specification Tool (Prospec)

Prospec [28] is a tool that guides a user in the development of formal specifications. It includes patterns and scopes, and it uses decision trees to assist users in the selection of appropriate patterns and scopes for a given property. Prospec 2.0 extends the capability of SPS by supporting the specification of CP classes for each parameter of a pattern or scope that is comprised of multiple conditions or events. The use of CP classes allows practitioners using Prospec to specify ordered sequences, non-deterministic sequences, and concurrency. By using CPs, a practitioner is directed to clarify requirements, which leads to reduced ambiguity and incompleteness of property specifications. Prospec generates formal specifications in Linear Temporal Logic.
2.4.3 Overview of Other Tools

Propel [29] allows practitioners to write and understand properties by using property templates, provided by the tool, that explicitly capture details as options for commonly occurring property patterns based on SPS. Propel’s property templates are represented using both disciplined natural language (DNL) and finite-state automata (FSA). The practitioner can view both representations simultaneously and select from which representation to elucidate the desired property.

The Timeline Editor [30] allows the formalization of certain type of requirements. To formalize these requirements a series of events and required system responses are placed on a timeline. The tool automatically converts the timeline specification automatically into a test automaton. The timeline specification can then be used directly by a logic model checker or by a test-sequence generator. Timeline Editor can neither capture group of events occurring in arbitrary order nor provide visual feedback for validation purposes.

SPIDER [24] generates specification properties using natural language representations. This process is based on a natural language grammar and specification pattern system to derive a natural language sentence. This sentence is then mapped to the temporal logic that can be analyzed formally by a tool such as SPIN. The structured language grammar supports translations of untimed and timed properties to multiple temporal logics.
Chapter 3: Data Property Categorization

3.1 Overview

In order to define an environmental-scientist-centered approach for specifying data properties based on temporal and data relationships associated with sensor data, several factors need to be understood. This chapter explores what data properties are relevant to sensor data collected by environmental scientists, what data aspects must be considered for properties related to time and location, and what are the differences between properties that check experimental conditions and those that check instrumentation. This chapter describes a literature survey conducted over scientific projects that collect data using sensors and the data property categorization derived from the literature survey.

3.2 Data Property Categorization

Checking the quality of sensor data is an essential step in data processing and requires identifying and analyzing data anomalies. In the literature, however, there is no consensus about checking criteria, which requires scientists to identify their own criteria. Scientist design different data quality systems to best meet their needs [32] and there is a need to agree on common criteria. For example, the same, quality levels for the MODIS aerosol sensor measurements over ocean and over land actually mean different quality, but this difference is not captured anywhere but in scientific folklore [33]. Several efforts have tried to standardize criteria. The Open Archival Information System (OAIS) [34] allows each discipline to apply criteria in the way that best fit its need, thus making OAIS criteria ambiguous in interdisciplinary applications. The ISO11179 [35] standard deals with metadata catalogs, but still allows levels of data granularity to differ, i.e., the standard does not take into consideration that one discipline’s data granule is a data collection to another [32].

3.2.1 Literature Survey

A literature survey was conducted to review and analyze current efforts in evaluating data quality documented by a total of 15 projects [Appendix D] focused on environmental sensor data collection. The projects illustrate how data quality is incorporated into sensor data collection systems and processes at field sites and data centers.
The reviewed projects were in one or more of the following fields: atmospheric studies (6), oceanography (9), meteorology (6), hydrology (1) and land productivity (1). The data collected through the projects include CO$_2$ concentration, carbon balance, energy balance, spectral data, bathymetric and thermography, water salinity, tide measurements, vessels data, and temperature and wind profiles.

![Figure 1](image)

**Figure 1.** Groupings of data checks and analysis using the terminology of the projects.

The groupings of data checks and analysis gleaned from the projects are summarized in Figure 1. The number of projects in which they occurred is given with each bar. As shown, the most frequently specified check is range limit (i.e., those that capture sensor readings thresholds that are environmentally sound according to the scientists’ expertise), followed by checks associated with instrument behavior (i.e., those related to conditions associated with the instrument during the data collection processes) and checks that use statistical analysis (i.e., checks performed after the data is processed and analyzed). Time continuity checks, which denote checks that have a time-dependent relationship among sensor data readings, also play an important role. Spatial estimates refer to checks that identify expected data values corresponding to physical conditions that influence experiment results such as rain, snow, etc.
Of the projects studied, different projects use different terminology to describe similar data checks or properties. For example, range checking can apply to checks that identify outliers and spikes. Those classified as data continuity include checks that could be referred to as data gaps, data relationships, or persistence in various projects. It is assumed that the measurements were performed at sequential moments of time, e.g. every second. It is also assumed that there are duplicate sensors that sense the same information, so that sensor errors can be detected by comparing the measurements collected by the sensors.

3.2.2 Data Property Categories

From analysis of the checks described in the previous subsection, the property categorization shown in Figure 2 resulted.

![Scientific Sensor Data Properties](image)

**Figure 2.** Scientific sensor data properties categorization.
The categorization was initially divided into two categories taking into consideration that anomalies can originate from environmental events or equipment malfunctioning to be consistent with the literature [6] [36] [37]. The data property groups identified in the literature survey made a distinction between those data properties that are related to time and those that are not. As a result, the two major data property categories were further subdivided to reflect these two sub-categories. Finally, the literature survey revealed distinctions between data properties that only analyzed one sensor a time, those that analyzed two or more sensor at a time, actions of an instrument based on the readings from another sensor, and values of a single sensor that varied depending on the conditions of an instrument.

The categorization divided the properties into two major types: experimental readings and experimental conditions. Experimental readings properties specify expected values and relationships related to field data readings and can be used to identify anomalies in a dataset, as well as random data errors, i.e., those errors that can be detected, estimated, and minimized by examining the convergence of calculations with increasing size of data sets [36, 37]. Experimental conditions properties specify expected instrument behavior and relationships by defining instrument attributes (e.g., low voltage). This type of properties can identify systematic errors, i.e., persistent offsets or multipliers that can affect the whole or a portion of the dataset [37]. The values being checked may be sensor readings, derived values based on one or more sensor readings, pre-defined values, and historical values.

Properties labeled experimental readings are divided into the following five subcategories:

Datum\(^1\): A datum (D) property specifies the expected value of a single sensor reading. A sensor reading is compared against a pre-defined or historical value. Example: *The relative humidity should always be greater than or equal to 0 and less than or equal to 1* [38].

Time-Dependent Datum: A time-dependent datum (TDD) property specifies the expected value(s) of a single type of sensor, where the readings are filtered by date and time. The selected sensor readings are compared against a predefined value or a historic value. Example: *During daylight on May 12th, the dry bulb temperature should be less than or equal to 103°F* [39].

\(^1\) For this work “Datum” is used to denote the singular form of “Data”
Datum Relationship: A datum-relationship (DR) property specifies the relationship between two or more types of sensor readings. A DR property can be used to compare sensor readings against readings from other types of sensors, or against a predefined constant value or historic value. Example: $\text{Temperature} < \text{Wet-Bulb-Temperature} < \text{Dew-Point-Temperature}$ [40].

Time-Dependent Datum Relationship: A time-dependent datum relationship (TDDR) property specifies the relationship between two or more related sensor readings that are filtered based on time. The selected readings may be compared against each other, against a predefined value, or an historic value. TDDR properties capture relationships within time series data and dataset behaviors dependent on time. Example: No two measurements of the consensus subset can differ by more than 1/8 of the maximum measurable velocity, where the consensus subset is created each hour by applying the consensus algorithm from the ten 6-minute radial velocity measurements on each antenna beam [41].

Instrument-Dependent Datum: An instrument-dependant datum (IDD) property is one that specifies a property about an instrument that influences behavior of the sensor readings. Example: If the profile lies close to land and the depth is smaller than 50 meters, the observed value should lie within 5 standard deviations from the mean value [42].

Experimental conditions properties are divided into the following five subcategories:

Instrument: An instrument (I) property specifies the expected behavior of an instrument by describing an attribute of the instrument. The attribute is compared against either a predefined value or an historic value. Example: The collection-time voltage sensor should fall inside the expected range [43].

Time-Dependent Instrument: A time-dependent instrument (TDI) property captures the expected behavior of a single instrument that is dependent on time. The instrument reading is compared against a predefined constant value, a historic value, or a time entity in a given time constraint. Example: Based on the time when the last scan took place, each radar must scan a 360-degree sector at the lowest two elevations every 2.5 minutes [44].

Instrument Relationship: An instrument relationship (IR) property captures the relationship between one or more related instruments. An IR property can be used to compare the behavior of the
instrument. Example: *If a current meter is used, at least one of the HCSP/HCDT or NSCT/EWCT sensor couples must be present* [45].

**Time-Dependent Instrument Relationship:** A time-dependent instrument relationship (TDIR) property captures the relationship between two or more related instruments and expected behavior based on time. A TDIR property can be used to compare instrument behavior dependent on a time. Example: *Based on the time when the last scan took place, perform sector scans of storms with 2 or more radars every 1-minute* [44].

**Data-Dependant Instrument:** A datum-dependant instrument (DDI) property captures a known datum or datum relationship whose value influences instrument behavior, or causes an instrument’s action. DDI properties capture continuity problems. Example: *If there is no change in current direction data [for 2 minutes], the system must generate an error alert* [46].

Using the data property categorization given in Figure 3, a total of 532 properties from the aforementioned projects were analyzed and classified. The process took three refinement iterations of the categorization. These iterations are labeled as “initial categorization,” “revised categorization,” and “tool categorization.”

The initial categorization had eight categories: datum, time-dependent datum, datum relationship, time-dependent datum relationships, instrument, time-dependent instrument, instrument relationship, and time-dependent datum relationship. As shown in Figure 3, the initial categorization classified 386 properties as experimental readings, 93 properties were classified as experimental conditions, and 53 properties were not classifiable. Figures 3 and 4 compare the number of properties classified for each subcategory of type experimental readings and experimental conditions.
The initial categorization was refined and extended to increase its coverage by adding the *instrument-dependent datum* and the *data-dependent instrument* categories. The categorization was revised because it was realized that the 53 data properties that could not be classified captured data values that depended on the conditions of the instrumentation or captured instrument functioning that depended on the numerical values being obtained from other sensors. The classification of properties under the revised categorization is shown in Figure 4. The new categorization resulted in a discrepancy between the number of initial properties placed in a particular category and those placed in a category using the revised categorization properties (other than the unclassified properties). There were properties in the initial categorization that were unclassified. Once the categorization was revised, some of the unclassified properties were placed in one of the new categories. A data property specification tool, which uses the second iteration of the categorization to guide the specification process, was developed to allow scientists to specify data properties. The tool was used to specify and categorize the data properties from the literature survey. The results are shown in Figures 3-5. The data property specification tool is further described in Chapter 5.
Figure 4. Experimental readings properties categories distributions.

Figure 5. Experimental conditions properties categories distributions.
3.3 **Data Property Categorization Findings**

The results obtained by the categorization revealed that the studied environmental projects captured more *experimental* than *systematic* properties. Scientist have concentrated less on instrument malfunctions even though the latter can be a source of anomalies in the data. In addition, with *experimental readings* properties, *datum properties* and *data relationships* were specified more frequently than *time-related* properties. In the *systematic errors* category, *datum-dependent* instrument properties ranked higher than the other categories, an indication that scientists use data inspection to determine instrument malfunctions instead of specifying separate instrumentation properties.

Several factors that can limit the effectiveness of the categorization for data property specification were identified during the property categorization process. Some data properties are described at such an abstract level that it is difficult to translate the property into a specification that can be automatically verified. Other data properties were complex, requiring them to be decomposed into several simpler properties. Due to the inherently ambiguous nature of natural languages it is difficult to determine the intended meaning of some data properties descriptions.

A number of specifications are a combination of data verification and data steering properties. Combined property specifications require both verifying that the properties adhere to predefined behaviors, the verification aspect, and guaranteeing that a reaction occurs in response to a data or instrument stimulus, the steering aspect. As a result, combined property specifications must be decomposed into separate data verification properties and data steering properties.
Chapter 4: Data Specification and Pattern System

4.1 Overview

Scientists need the means to specify error detection properties for field-based data gathering instruments. A frequent practice for instrument manufacturers is to embed error detection properties into the instrument’s source code. The main drawback for this approach is that error detection properties are limited by the implementation and what the manufacturer had in mind in term of error detection properties when the system was conceived. Further changes to existing error detecting properties and the addition of new properties require modifications to the existing source code. Due to frequent changes in scientific procedures and the large amount of variables in field experiments, there is a need for a dynamic way to specify error detection properties.

A common practice for scientists is to use a natural language to specify data properties; however, the ambiguity of natural languages can cause properties to be misinterpreted by other scientists. An alternative to specifying data properties while mitigating ambiguity is for the scientist to specify the properties using a formalism based on mathematical logic. However, it is not realistic to expect scientists to learn new formalisms or logics. Software engineering techniques have been used to address this challenge. For example, the Specification and Pattern System (SPS) [23] was developed to facilitate the specification of critical software properties; SPS was unable to capture timing-based requirements. Konrad and Cheng [24] addressed this limitation by extending the SPS to support specification of collection-time properties. However, neither SPS nor the extensions by Konrad and Cheng permit the specification of data properties. The Data Property Specification and Pattern System developed as a component of this dissertation expands the concepts from both the original SPS and the collection-time extension. SPS and D-SPS properties are specified using a scope and pattern; however, while SPS properties are verified against execution traces, D-SPS properties are evaluated against sequences of data readings of single sensors extracted from datasets.

A data reading, \( \delta \), is a key-value pair, \( \langle \alpha, \beta \rangle \), where \( \alpha \) denotes a unique indexing value, e.g. time-stamp, and \( \beta \) denotes a sensor reading value. In addition, D-SPS uses the data property categorization to define applicable patterns and scopes that can be evaluated in a data property.
This chapter shows how a specification pattern system approach to specification can facilitate the ability of environmental scientists to specify data properties based on temporal and data relationships associated with sensor data.

4.2 DATA PROPERTY SPECIFICATION AND PATTERN SYSTEM (D-SPS)

Data property specification using the Data Property Specification and Pattern System (D-SPS) is composed of the following components: patterns, scopes, Boolean statements, and data property categories. D-SPS data property specifications use patterns, which are common occurring data properties, to define how Boolean statements are evaluated over data subsequences defined by scopes. A data property category, as defined in Chapter 3, refines the pattern selection based on a data property’s dependency on time, number of sensors, and sensor relationships. The following subsections describe the components of D-SPS.

4.2.1 D-SPS Scopes

A scope is a sequence from a dataset of interest. Table 1 presents the formal definitions and descriptions for all D-SPS scopes. The formal definitions are defined using Z notation as described in [74]. The definitions use the following functions:

- #D denotes the cardinality of a sequence D.
- index(D,e): Given a non-empty sequence D and element e ∈ D, index(D,e) returns the index of element e in sequence D, where e is the first element of a pair in the sequence.
Table 1. D-SPS scope formal definitions and descriptions.

<table>
<thead>
<tr>
<th>Definition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>$D' = &lt; (\alpha_1, \beta_1), \ldots, (\alpha_{#D}, \beta_{#D})&gt;$</td>
</tr>
<tr>
<td>Before R</td>
<td>$((\exists x \in \text{dom } D : (R = x)) \rightarrow D' = &lt; (\alpha_1, \beta_1), \ldots, (\alpha_{#D}, \beta_{#D})&gt;) \land ((\forall x \in \text{dom } D : (R \neq x)) \rightarrow D' = &lt;&gt;)$</td>
</tr>
<tr>
<td>After L</td>
<td>$((\exists x \in \text{dom } D : (L = x)) \rightarrow D' = &lt; (L, \beta_{\text{index}(D,L)}), \ldots, (\alpha_{#D}, \beta_{#D})&gt;) \land ((\forall x \in \text{dom } D : (L \neq x)) \rightarrow D' = &lt;&gt;)$</td>
</tr>
<tr>
<td>Between L and R</td>
<td>$((\exists x \in \text{dom } D, \exists y \in \text{dom } D : (L = x \land R = y)) \rightarrow D' = &lt; (L, \beta_{\text{index}(D,L)}), \ldots, (R, \beta_{\text{index}(D,R)})&gt;) \land ((\forall x \in \text{dom } D : (L \neq x)) \rightarrow D' = &lt;&gt; \land ((\forall y \in \text{dom } D : (R \neq y)) \rightarrow D' = &lt;&gt;)$</td>
</tr>
<tr>
<td>After L until R</td>
<td>$((\exists x \in \text{dom } D, \exists y \in \text{dom } D : (L = x \land R = y)) \rightarrow D' = &lt; (L, \beta_{\text{index}(D,L)}), \ldots, (R, \beta_{\text{index}(D,R)})&gt;) \land ((\exists x \in \text{dom } D, \exists y \in \text{dom } D : (L = x \land R \neq y)) \rightarrow D' = &lt; (L, \beta_{\text{index}(D,L)}), \ldots, (\alpha_{#D}, \beta_{#D})&gt;) \land ((\forall x \in \text{dom } D : (L \neq x)) \rightarrow D' = &lt;&gt;)$</td>
</tr>
</tbody>
</table>

### 4.2.2 D-SPS Boolean Statements

Boolean statements use relational operators ($<, \leq, =, \neq, \geq, >$) applied to data readings to establish relationships between sensors. For this work, a Boolean statement is classified as type “single sensor reading” (denoted by S) when a sensor reading is compared to a constant numerical value. A Boolean statement is classified as type “multiple sensor reading” (denoted by M) when a sensor reading is
compared to another sensor reading. For example, $temp < 20$ denotes a single sensor reading type and $temp = age_temp$ denotes a multiple sensor reading type. It is assumed that in both cases that the sensor readings are of the same measurement type. Enforcement of this assumption is described in Section 5.3.3.

### 4.2.3 D-SPS Patterns

The D-SPS uses *patterns* and *Boolean statements* to specify data properties. A *property pattern* is a high-level abstraction describing a commonly occurring property of a scientific dataset. Property patterns for this work were defined based on the commonly occurring properties extracted from the data property categorization. The SPS definitions, without the collection-time extension, were adapted to create the initial D-SPS. The original D-SPS included the quantitative, not time-constrained, patterns: *Universality, Absence, Existence, Precedence* and *Response*. The formal definitions for the quantitative patterns are provided in Tables 2 to 6. The original D-SPS was used to specify the properties from the data property categorization; however, time-dependent properties could not be specified with the original D-SPS.

### 4.2.4 Extended D-SPS Patterns from Data Property Categories

The data property categorization makes a distinction between properties that are dependent on time and those that are not, and within these two categories, there are sub-categories that capture the dependencies based on the number of sensors and sensor relationships associated with the property. Given the similarities between the categories in the data property categorization and the patterns supported by the D-SPS, it is possible to relate the data patterns from the D-SPS to the categories in the data property categorization. For example, the data property category of type “time dependent datum” captures an expected behavior of a single sensor readings dataset that depend on time. The D-SPS provides patterns to evaluate a single data property’s occurrence (*Maximum Duration, Minimum Duration*) or recurrence (*Bounded Recurrence*) that are dependent on time. Similar relationships can be built for all of the data property categories and data patterns. The relationships are constructed based on the type and number of Boolean statement(s) to be evaluated and on whether the Boolean statement(s) evaluation depend(s) on time. Time dependent properties are defined in Tables 7 to 11.
**Time Dependent:** Data property categories that depend on time are of two types: properties that evaluate a single sensor readings dataset over time, i.e., *Time-Dependent Datum* and *Time-Dependent Instrument*, and properties that evaluate multiple sensor readings datasets over time, i.e., *Time-Dependent Datum Relationship* and *Time-Dependent Instrument Relationship*.

For *time-dependent datum* and *time-dependent instrument* properties, the D-SPS patterns are restricted to *Minimum Duration*, *Maximum Duration*, and *Bounded Recurrence*; these patterns assess a single Boolean statement of type *S* over time.

For time-related categories with multiple sensor readings datasets over time, i.e. *time-dependent datum relationship* and *time-dependent instrument relationship*, the D-SPS patterns are restricted to: *Minimum Duration*, *Maximum Duration*, and *Bounded Recurrence* for which a pattern that assess a single Boolean statement of type *M* over time is used, and *Bounded Response* and *Bounded Invariance* that assess a combination of two Boolean statements of any combination of type *S* and *M*.

For timed patterns, units of time are based on a sequence of readings and indexing values in a dataset of interest are assumed to be ordered time stamps, with equal constant time resolution, i.e., the time stamps are all of the same time measurement unit and change at a same constant time rate. The scientist is expected to align the unit of time used in the dataset to the unit of time used in the data property specification. For example, consider a scientist that wants to specify a property *P* that captures a Boolean statement *B* that must be evaluated every 5 minutes over dataset *D*. If the time resolution for *D* is minutes, *B* is evaluated every 5 units (data readings) in *D*. If the time resolution for *D* is seconds, *B* is evaluated every 300 units (data readings) in *D*. The formal definitions for timed patterns are provided in Table 2.

**Not Time Dependent:** Similarly, data property categories that do not depend on time are of three types: properties that evaluate a dataset for a single sensor, i.e., *Datum* and *Instrument*; properties that evaluate multiple sensor readings within the same data property category, i.e., *Datum Relationship* and *Instrument Relationship*; and properties that evaluate multiple sensor readings from different data property categories, i.e., *Instrument-Dependent Datum* and *Datum-Dependent Instrument*.  

32
For *datum* and *instrument* properties, D-SPS patterns are restricted to: *Absence, Universality, and Existence*; these patterns assess a single Boolean statement of type $S$ over the sensor readings dataset.

For *datum relationship* and *instrument relationship*, D-SPS patterns are restricted to: *Absence, Universality, Existence, Precedence, and Response*. The *Absence, Universality* and *Existence* patterns evaluate a single Boolean statement of type $M$ over the sensor readings datasets. The *Precedence* and *Response* patterns combine two Boolean statements over the datasets. The Boolean statements can be one combination of type $S$ and $M$. For *Instrument-Dependent Datum* and *Datum-Dependent Instrument*, the D-SPS patterns are restricted to *Precedence and Response*; these properties assess a combination of two Boolean statements over the sensor readings dataset. The Boolean statements can be of any combination of type $S$ and $M$. Figure 6 illustrates the relationship between the data property categorization, the patterns, and the type and number of Boolean statements associated with each pattern.

![Diagram](image)

**Figure 6.** The relationship between the categorization, patterns and Boolean statements.
A different number of Boolean statements are evaluated by the patterns over a set of scopes. For pattern that are evaluated over a minimum of two scopes and a maximum of four scopes, the single scope \((s)\) subscript represents the evaluation of two scopes, the multiple scopes to single scope relationship \((ms)\) subscript represents the relation between two initial scopes that have an effect on a third scope, the single scope to multiple scopes relationship \((sm)\) subscript represents the relation between an initial scope that have an effect on two subsequent scopes, and the multiple scope to multiple scope relationship \((mm)\) subscript captures the relation between two initial scopes which evaluation have implications for a second pair of scopes to be evaluated. The number of scopes associated to every pattern corresponds to the number of sensor datasets that can be evaluated by the pattern’s Boolean statement(s). For this work, it is assumed that for patterns that require two or more scopes, the sensor readings are indexed by the same \(\alpha\) value.

Every pattern definition includes the name of the pattern, a pattern type, the Boolean statement type associated with the property, a natural language description of the pattern, the data categories from the data categorization that use the pattern, a formal representation of the pattern, and a pseudo-code representation of the pattern. The definitions used to formally characterize D-SPS patterns are described in Tables 2 to 11.

Let:

\[ S_i \] be the subsequence of a dataset \(D_i\) as defined by some scope \(<(\alpha_{i,1}, \beta_{i,1}), \ldots,(\alpha_{i,\#S_i}, \beta_{i,\#S_i})>\), where \(i \in \{1,\ldots,4\}\).

\( P \) be a Boolean statement to be evaluated over the sensor readings \(\{\beta_{i,1},\ldots,\beta_{i,\#S_i}\} \in S_i\).

\( T \) be a Boolean statement to be evaluated over the sensor readings \(\{\beta_{i,1},\ldots,\beta_{i,\#S_i}\} \in S_i\).

\( C \) be a number of readings.
### Table 2. D-SPS Universality pattern formal definitions and descriptions.

**Universality Patterns: The Boolean Statement $P$ Always Holds.**

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Universality$_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Type:</td>
<td>Occurrence</td>
</tr>
<tr>
<td>Boolean Statement Type:</td>
<td>Single Sensor Reading ($S$)</td>
</tr>
<tr>
<td>English Description:</td>
<td>The Boolean statement $P$ always holds over scope $S_1$.</td>
</tr>
<tr>
<td>Data Category Used By:</td>
<td>- Data</td>
</tr>
<tr>
<td>- Instrument</td>
<td></td>
</tr>
<tr>
<td>Formal Definition:</td>
<td>$\forall i \in {1, \ldots, #S_1}: P\left(\beta_{1,i}\right)$</td>
</tr>
</tbody>
</table>
| Pseudo-Code Representation: | for (i=0; i<=size of S_1[]; i++) {  
  if not P(S_1[β_{1,i}]){
    Print to Screen “Anomaly Found”
  }
} |

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Universality$_M$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Type:</td>
<td>Occurrence</td>
</tr>
<tr>
<td>Boolean Statement Type:</td>
<td>Multiple Sensor Reading ($M$)</td>
</tr>
<tr>
<td>English Description:</td>
<td>The Boolean statement $P$ always holds over scopes $S_1$ and $S_2$.</td>
</tr>
<tr>
<td>Data Category Used By:</td>
<td>- Data Relationship.</td>
</tr>
<tr>
<td>- Instrument Relationship.</td>
<td></td>
</tr>
<tr>
<td>Formal Definition:</td>
<td>$(#S_1 = #S_2) \land (\forall i \in {1, \ldots, #S_1}: (\alpha_{1,i} = \alpha_{2,i}) \land P\left(\beta_{1,i}, \beta_{2,i}\right)$)</td>
</tr>
</tbody>
</table>
| Pseudo-Code Representation: | if size of S_1[] == size of S_2[]{
  for (i=0; i<=size of S_1[]; i++) {
    if S_1[α_{1,i}]!= S_2[α_{2,i}] {
      Print to Screen “Different indexing value”
    }
    if not P(S_1[β_{1,i}], S_2[β_{2,i}]) {
      Print to Screen “Anomaly Found”
    }
  }
} |

### Table 3. D-SPS Absence pattern formal definitions and descriptions.

**Absence Patterns: The Boolean Statement $P$ Never Holds.**

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Absence$_S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Type:</td>
<td>Occurrence</td>
</tr>
<tr>
<td>Boolean Statement Type:</td>
<td>Single Sensor Reading ($S$)</td>
</tr>
<tr>
<td>Formal Definition:</td>
<td>$(#S_1 = #S_2) \land (\forall i \in {1, \ldots, #S_1}: (\alpha_{1,i} = \alpha_{2,i}) \land \neg P\left(\beta_{1,i}, \beta_{2,i}\right)$)</td>
</tr>
</tbody>
</table>
| Pseudo-Code Representation: | if size of S_1[] == size of S_2[]{
  for (i=0; i<=size of S_1[]; i++) {
    if S_1[α_{1,i}] == S_2[α_{2,i}] {
      Print to Screen “Same indexing value”
    }
    if not P(S_1[β_{1,i}], S_2[β_{2,i}]) {
      Print to Screen “Anomaly Found”
    }
  }
} |
Table 4. D-SPS Existence pattern formal definitions and descriptions.

<table>
<thead>
<tr>
<th>Pattern Name</th>
<th>Pattern Type</th>
<th>Boolean Statement Type</th>
<th>Data Category Used By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Existence</td>
<td>Occurrence</td>
<td>Single Sensor Reading (S)</td>
<td>- Data</td>
</tr>
<tr>
<td>English Description</td>
<td>The Boolean statement P holds at least once over the scope S_i.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal Definition</td>
<td>$\exists i \in {1, \ldots, #S_i} : P \left( \beta_{1,i} \right)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo-Code Representation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>English Description: The Boolean statement P never holds over scopes $S_i$ and $S_2$.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal Definition: $(#S_1 = #S_2) \wedge (\forall i \in {1, \ldots, #S_i}: (\alpha_{1,i} = \alpha_{2,i}) \wedge \lnot P (\beta_{1,i}, \beta_{2,i}))$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo-Code Representation:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pattern Name: Absence_M</td>
<td>Occurrence</td>
<td>Multiple Sensor Reading (M)</td>
<td>- Data Relationship.</td>
</tr>
<tr>
<td>English Description: The Boolean statement P never holds over scope $S_i$.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Formal Definition: $\forall i \in {1, \ldots, #S_i} : \lnot P (\beta_{1,i})$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo-Code Representation:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Pseudo-Code Representation:**

```plaintext
Bool flag=false

for (i=0; i<=size of S1[]; i++){
    if P(S1[β1,i]){
        flag=true;
    }
}

if flag==false{
    Print to Screen "Anomaly Found"
}
```

**Pattern Name:** Existence<sub>M</sub>

<table>
<thead>
<tr>
<th>Pattern Type:</th>
<th>Boolean Statement Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Occurrence</td>
<td>Multiple Sensor Reading (M)</td>
</tr>
</tbody>
</table>

**English Description:**
The Boolean statement P holds at least once over scopes S<sub>1</sub> and S<sub>2</sub>.

**Formal Definition:**
\((\#S_1 = \#S_2) \wedge (\exists i \in \{1, \ldots, \#S_1\}: (\alpha_{1,i} = \alpha_{2,i}) \wedge P(\beta_{1,i}, \beta_{2,i}))\)

**Pseudo-Code Representation:**

```plaintext
Bool flag=false

if size of S1[] == size of S2[]{
    for (i=0; i<=size of S1[]; i++){
        if S1[α<sub>1,i</sub>] != S2[α<sub>2,i</sub>] {
            Print to Screen "Different indexing value"
        }
        if P(S1[β<sub>1,i</sub>], S2[β<sub>2,i</sub>]) {
            flag=true
        }
    }
}

if flag==false{
    Print to Screen "Anomaly Found"
}
```

---

**Table 5. D-SPS Precedence pattern formal definitions and descriptions.**

<table>
<thead>
<tr>
<th>Pattern Name: Precedence&lt;sub&gt;SS&lt;/sub&gt;</th>
<th>Boolean Statement Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Single Sensor Reading (S)</td>
</tr>
<tr>
<td></td>
<td>Single Sensor Reading (S)</td>
</tr>
</tbody>
</table>

**English Description:**
The Boolean statement T holds over scope S<sub>1</sub> before Boolean statement P eventually holds over scope S<sub>2</sub>.

**Data Category Used By:**
- Data Relationship.
- Instrument Relationship.
- Instrument Dependent Data.
- Data Dependent Instrument.
Formal Definition:

\((\#S_1 = \#S_2) \land (\forall i \in \{1, \ldots, \#S_1\}: (\alpha_{1,i} = \alpha_{2,i}) \land T(\beta_{1,i}) \rightarrow (\exists j \in \{i, \ldots, \#S_2\}: P(\beta_{2,j})))\)

Pseudo-Code Representation:

```java
Bool flag=false
if size of S_1[] == size of S_2[]{
    for (i=0; i<=size of S_1[]; i++){
        if S_1[α_{1,i}] == S_2[α_{2,i}] {
            Print to Screen “Different indexing value”
        }
        if T(S[β_{1,i}]) {
            for (j=1; j<=size of S_2[]; j++){
                if P(S_2[β_{2,j}]){
                    flag=true
                }
            }
        }
        if (flag == false){
            Print to Screen “Anomaly Found”
        }
    }
}
```

Pattern Name: Precedence_{MS}

<table>
<thead>
<tr>
<th>Pattern Type:</th>
<th>Boolean Statement Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>-Multiple Sensor Reading (M)</td>
</tr>
<tr>
<td></td>
<td>-Single Sensor Reading (S)</td>
</tr>
</tbody>
</table>

English Description:
The Boolean statement T holds over scopes S_1 and S_3 before Boolean statement P eventually holds over scope S_2.

Data Category Used By:
- Data Relationship.
- Instrument Relationship.
- Instrument Dependent Data.
- Data Dependent Instrument.

Formal Definition:

\((\#S_1 = \#S_2 \land \#S_2 = \#S_3) \land (\forall i \in \{1, \ldots, \#S_1\}: (\alpha_{1,i} = \alpha_{2,i} \land \alpha_{2,i} = \alpha_{3,i}) \land T(\beta_{1,i}, \beta_{3,i}) \rightarrow (\exists j \in \{i, \ldots, \#S_2\}: P(\beta_{2,j})))\)

Pseudo-Code Representation:

```java
Bool flag=false
if size of S_1[] == size of S_2[] == size of S_3[] {
    for (i=0; i<=size of S_1[]; i++){
        if S_1[α_{1,i}] == S_2[α_{2,i}] == S_3[α_{3,i}] {
            Print to Screen “Different indexing value”
        }
        if T(S_1[β_{1,i}], S_2[β_{3,i}]) {
            for (j=1; j<=size of S_2[]; j++){
                if P(S_2[β_{2,j}]){
                    flag=true
                }
            }
        }
        if (flag == false){
            Print to Screen “Anomaly Found”
        }
    }
}
```
### Pattern Name: Precedence_{SM}

<table>
<thead>
<tr>
<th>Pattern Type:</th>
<th>Boolean Statement Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Single Sensor Reading (S)</td>
</tr>
<tr>
<td></td>
<td>Multiple Sensor Reading (M)</td>
</tr>
</tbody>
</table>

**English Description:**
The Boolean statement T holds over scopes $S_1$ before Boolean statement P eventually holds over scope $S_2$ and $S_4$.

**Data Category Used By:**
- Data Relationship.
- Instrument Relationship.
- Instrument Dependent Data.
- Data Dependent Instrument.

**Formal Definition:**
\[
(#S_1 = #S_2 \land #S_2 = #S_4) \land (\forall i \in \{1, \ldots, #S_1\}: (\alpha_{i_1} = \alpha_{i_2} \land \alpha_{i_2} = \alpha_{i_4}) \land T(\beta_{i_1})) \rightarrow (\exists j \in \{i, \ldots, #S_2\}: P(\beta_{j_2}, \beta_{j_4}))
\]

**Pseudo-Code Representation:**
```python
Bool flag=false
if size of $S_1[] == size of S_2[] == size of S_4[] {
    for (i=0; i<=size of S_1[]; i++){
        if $S_1[\alpha_{i_1}] != S_2[\alpha_{i_2}] != S_3[\alpha_{i_3}] != S_4[\alpha_{i_4}]$
            Print to Screen "Different indexing value"
        }
        if T($S_1[\beta_{j_1}]$) {
            for (j=i; j<=size of S_2[]; j++)
                if P($S_2[\beta_{j_2}], S_4[\beta_{j_4}]$) {
                    flag=true
                    }
        }
        if (flag == false){
            Print to Screen "Anomaly Found"
        }
    }
}
```

### Pattern Name: Precedence_{MM}

<table>
<thead>
<tr>
<th>Pattern Type:</th>
<th>Boolean Statement Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td>Multiple Sensor Reading (M)</td>
</tr>
<tr>
<td></td>
<td>Multiple Sensor Reading (M)</td>
</tr>
</tbody>
</table>

**English Description:**
The Boolean statement T holds over scopes $S_1$ and $S_3$ before Boolean statement P eventually holds over scope $S_2$ and $S_4$.

**Data Category Used By:**
- Data Relationship.
- Instrument Relationship.
- Instrument Dependent Data.
- Data Dependent Instrument.

**Formal Definition:**
\[
(#S_1 = #S_2 \land #S_2 = #S_4) \land ((\forall i \in \{1, \ldots, #S_1\}: (\alpha_{i_1} = \alpha_{i_2} \land \alpha_{i_2} = \alpha_{i_3} \land \alpha_{i_3} = \alpha_{i_4}) \land T(\beta_{i_1})) \rightarrow (\exists j \in \{i, \ldots, #S_2\}: P(\beta_{j_2}, \beta_{j_4}))
\]

**Pseudo-Code Representation:**
```python
Bool flag=false
if size of $S_1[] == size of S_2[] == size of S_3[] == size of S_4[] {
    for (i=0; i<=size of S_1[]; i++){
        if $S_1[\alpha_{i_1}] != S_2[\alpha_{i_2}] != S_3[\alpha_{i_3}] != S_4[\alpha_{i_4}]$
            Print to Screen "Different indexing value"
    }
}
```
Table 6. D-SPS Response pattern formal definitions and descriptions.

<table>
<thead>
<tr>
<th>Pattern Name:</th>
<th>ResponseSS</th>
<th>Boolean Statement Type:</th>
<th>Data Category Used By:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Type:</td>
<td>Order</td>
<td>-Single Sensor Reading (S)</td>
<td>-Data Relationship.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Single Sensor Reading (S)</td>
<td>-Instrument Relationship.</td>
</tr>
</tbody>
</table>

**English Description:**
The Boolean statement P holds over scope $S_1$ on the same, or the immediately following, sensor reading after Boolean statement T holds over scope $S_2$.

**Formal Definition:**
$$(\#S_1 = \#S_2) \land (\forall i \in \{1, \ldots, \#S_1\}: ((\alpha_{1,i} = \alpha_{2,i}) \land T(\beta_{1,i})) \rightarrow (P(\beta_{2,i}) \lor P(\beta_{2,i+1})))$$

**Pseudo-Code Representation:**
```plaintext
Bool flag=false
if size of $S_1[]$ == size of $S_2[]$ {
    for (i=0; i<=size of $S_1[]$-1; i++){
        if $S_1[\alpha_{1,i}]$ != $S_2[\alpha_{2,i}]$ {
            Print to Screen “Different indexing value”
        }
        if T($S_1[\beta_{1,i}]$) {
            if P($S_2[\beta_{2,i}]$) {
                flag=true
            }
            if P($S_2[\beta_{2,i+1}]$) {
                flag=true
            }
        }
    }
    if (flag == false){
        Print to Screen “Anomaly Found”
    }
}
```

<table>
<thead>
<tr>
<th>Pattern Name:</th>
<th>ResponseMS</th>
<th>Boolean Statement Type:</th>
<th>Data Category Used By:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Type:</td>
<td>Order</td>
<td>-Multiple Sensor Reading (M)</td>
<td>-Data Relationship.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-Single Sensor Reading (S)</td>
<td>-Instrument Relationship.</td>
</tr>
</tbody>
</table>

```
The Boolean statement P holds over scope \( S_2 \) on the same, or the immediately following, sensor reading after Boolean statement T holds over scopes \( S_1 \) and \( S_3 \).

**Formal Definition:**

\[
(#S_1 = #S_2 \land #S_2 = #S_3) \land (\forall i \in \{1, \ldots, #S_1\}: ((\alpha_{1,i} = \alpha_{2,i} \land \alpha_{2,i} = \alpha_{3,i}) \land T(\beta_{1,i}, \beta_{3,i})) \rightarrow (P(\beta_{2,i}) \lor P(\beta_{2,i+1})))
\]

**Pseudo-Code Representation:**

```plaintext
Bool flag=false
if size of S_1[] == size of S_2[] == size of S_3[]{
    for (i=0; i<=size of S_1[]-1; i++){
        if S_1[\alpha_{1,i}] != S_2[\alpha_{2,i}] != S_3[\alpha_{3,i}]{
            Print to Screen “Different indexing value”
        }
        if T(S_1[\beta_{1,i}], S_3[\beta_{3,i}]){
            if P(S_2[\beta_{2,i}]){
                flag=true
            }
            if P(S_2[\beta_{2,i+1}]){
                flag=true
            }
        }
        if (flag == false){
            Print to Screen “Anomaly Found”
        }
    }
}
```

**Pattern Name:** ResponseSM

**Pattern Type:** Order

**Boolean Statement Type:**
- Single Sensor Reading (S)
- Multiple Sensor Reading (M)

**English Description:**
The Boolean statement P holds over scopes \( S_2 \) and \( S_4 \) on the same, or the immediately following, sensor reading after Boolean statement T holds over scope \( S_1 \).

**Formal Definition:**

\[
(#S_1 = #S_2 \land #S_2 = #S_4) \land (\forall i \in \{1, \ldots, #S_1\}: ((\alpha_{1,i} = \alpha_{2,i} \land \alpha_{2,i} = \alpha_{4,i}) \land T(\beta_{1,i})) \rightarrow (P(\beta_{2,i}) \lor P(\beta_{2,i+1}, \beta_{4,i+1}))
\]

**Pseudo-Code Representation:**

```plaintext
Bool flag=false
if size of S_1[] == size of S_2[] == size of S_4[]{
    for (i=0; i<=size of S_1[]-1; i++){
        if S_1[\alpha_{1,i}] == S_2[\alpha_{2,i}] == S_4[\alpha_{4,i}]{
            Print to Screen “Different indexing value”
        }
        if T(S_1[\beta_{1,i}]){
            if P(S_2[\beta_{2,i}], S_4[\beta_{4,i}]){
                flag=true
            }
        }
        if P(S_2[\beta_{2,i+1}], S_4[\beta_{4,i+1}]){
```
Table 7. D-SPS Minimum Duration pattern formal definitions and descriptions.

<table>
<thead>
<tr>
<th>Pattern Name: Minimum Duration patterns</th>
<th>Boolean Statement Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum Duration</td>
<td>Single Sensor Reading (S)</td>
</tr>
</tbody>
</table>
### English Description:
The Boolean statement P holds for a minimum of c consecutive readings over scope $S_1$.

### Formal Definition:
c denotes a time unit count (constant value).

$$\exists i \in \{1, \ldots, \#S_1\} : (\forall j \in \{i, \ldots i+c\} : P(\beta_{i,j}))$$

### Pseudo-Code Representation:
```
for (i=0; i<=size of $S_1[]$; i++) {
    if $P(S_1[\beta_{i,j}])$ {
        for (j=i; j<=i+c; j++) {
            if not $P(S_1[\beta_{i,j}])$ {
                Print to Screen "Anomaly Found"
            }
        }
    }
}
```

### Pattern Name: Minimum Duration$_M$

### Boolean Statement Type:
**Multiple Sensor Reading (M)**

### Pattern Type: Duration

### English Description:
The Boolean statement P holds for a minimum of c consecutive readings over scopes $S_1$ and $S_2$.

### Formal Definition:
c denotes time unit count (constant value).

$$\left(\#S_1 = \#S_2 \land (\exists i \in \{1, \ldots, \#S_1\} : (\alpha_{1,i} = \alpha_{2,i}) \land (\forall j \in \{i, \ldots i+c\} : (\alpha_{1,j} = \alpha_{2,j}) \land P(\beta_{1,j}, \beta_{2,j}))\right)$$

### Pseudo-Code Representation:
```
if size of $S_1[]$ == size of $S_2[]$ {
    for (i=0; i<=size of $S_1[]$; i++) {
        if $S_1[\alpha_{1,i}] != S_2[\alpha_{2,i}]$ {
            Print to Screen "Different indexing value"
        }
        if $P(S_1[\beta_{1,j}].S_2[\beta_{2,j}])$ {
            for (j=i; j<=i+c; j++) {
                if not $P(S_1[\beta_{1,j}].S_2[\beta_{2,j}])$ {
                    Print to Screen "Anomaly Found"
                }
            }
        }
    }
}
```
Table 8. D-SPS Maximum Duration pattern formal definitions and descriptions.

<table>
<thead>
<tr>
<th>Pattern Name: Maximum Duration&lt;sub&gt;S&lt;/sub&gt;</th>
<th>Boolean Statement Type: Single Sensor Reading (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pattern Type:</strong> Duration</td>
<td><strong>English Description:</strong> The Boolean statement P holds for a maximum of c consecutive readings over scope S&lt;sub&gt;i&lt;/sub&gt;.</td>
</tr>
</tbody>
</table>
| | **Formal Definition:**
c denotes a time unit count (constant value).
∃i ∈ {1, ..., #S<sub>1</sub>}: (∀j ∈ {i, ..., i+c}: P(β<sub>1j</sub>) ∧ ¬P(β<sub>1,i+c+1</sub>)) |
| | **Pseudo-Code Representation:**
Bool holds=false
for (i=0; i<=size of S<sub>i</sub>[]; i++){
  if P(S<sub>i</sub>[β<sub>1i</sub>]) {
    for (j=i; j<=i+c; j++){
      if not P(S<sub>i</sub>[β<sub>1j</sub>]) {
        holds=true
      }
    }
  }
  if holds==false{
    if P(S<sub>i</sub>[β<sub>1,i+c+1</sub>]){
      Print to Screen “Anomaly Found”
    }
  }
}

<table>
<thead>
<tr>
<th>Pattern Name: Maximum Duration&lt;sub&gt;M&lt;/sub&gt;</th>
<th>Boolean Statement Type: Multiple Sensor Reading (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pattern Type:</strong> Duration</td>
<td><strong>English Description:</strong> The Boolean statement P holds for a maximum of c consecutive readings over scopes S&lt;sub&gt;i&lt;/sub&gt; and S&lt;sub&gt;2&lt;/sub&gt;.</td>
</tr>
</tbody>
</table>
| | **Formal Definition:**
c denotes time unit count (constant value).
(#S<sub>1</sub> = #S<sub>2</sub>) ∧ (∃i ∈ {1, ..., #S<sub>1</sub>}: (α<sub>1i</sub> = α<sub>2i</sub>) ∧ (∀j ∈ {i, ..., i+c}: (α<sub>1j</sub> = α<sub>2j</sub>) ∧ (P(β<sub>1j</sub>, β<sub>2j</sub>) ∧ ¬P(β<sub>1,i+c+1</sub>, β<sub>2,i+c+1</sub>))) |
| | **Pseudo-Code Representation:**
Bool holds=false
if size of S<sub>i</sub>[] == size of S<sub>2</sub>[]{ 
  for (i=0; i<=size of S<sub>i</sub>[]; i++){
    if S<sub>i</sub>[α<sub>1i</sub>] != S<sub>i</sub>[α<sub>2i</sub>]{
      Print to Screen “Different indexing value”
    }
    if P(S<sub>i</sub>[β<sub>1i</sub>], S<sub>2</sub>[β<sub>2i</sub>]){
      for (j=i; j<=i+c; j++){

    }
  }
}
if not \( P(S_1[\beta_{1i}], S_2[\beta_{2j}]) \) {
    holds=true
}
if holds==false{
    if \( P(S_1[\beta_{1i+c+1}], S_2[\beta_{2i+c+1}]) \) {
        Print to Screen “Anomaly Found”
    }
}

Table 9. D-SPS Bounded Recurrence pattern formal definitions and descriptions.

<table>
<thead>
<tr>
<th>Pattern Name: Bounded Recurrence,</th>
<th>Boolean Statement Type: Single Sensor Reading (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Type: Periodic</td>
<td></td>
</tr>
<tr>
<td>English Description:</td>
<td>The Boolean statement P holds every c counts of readings over scope ( S_1 ).</td>
</tr>
<tr>
<td>Formal Definition:</td>
<td>( \exists i \in [1, #S_1] ): (( \forall j \in {0...((#S_1-i)/c)} : P(\beta_{1i+c+j}) ) )</td>
</tr>
</tbody>
</table>
| Pseudo-Code Representation:       | for \( i=0; i<=\text{size of } S_1[]; i++\) { |}
|                                   | \hspace{1em} if \( P(S_1[\beta_{1i}] \) { |}
|                                   | \hspace{2em} \text{for } j=0; j<=\text{size of } S_1[]-i/c; j++ { |}
|                                   | \hspace{3em} if not \( P(S_1[\beta_{1i+c+j}] \) { |}
|                                   | \hspace{4em} Print to Screen “Anomaly Found” |}
|                                   | } |}
|                                   | for \( i=0; i<=\text{size of } S_1[]; i++ \) { |}
|                                   | \hspace{1em} if \( P(S_1[\beta_{1i}] \) { |}
|                                   | \hspace{2em} \text{for } j=0; j<=\text{size of } S_1[]-i/c; j++ { |}
|                                   | \hspace{3em} if not \( P(S_1[\beta_{1i+c+j}] \) { |}
|                                   | \hspace{4em} Print to Screen “Anomaly Found” |}
|                                   | } |}

<table>
<thead>
<tr>
<th>Pattern Name: Bounded Recurrence,m</th>
<th>Boolean Statement Type: Multiple Sensor Reading (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern Type: Periodic</td>
<td></td>
</tr>
<tr>
<td>English Description:</td>
<td>The Boolean statement P holds every c counts of readings over scopes ( S_1 ) and ( S_2 ).</td>
</tr>
<tr>
<td>Formal Definition:</td>
<td>( #S_1 = #S_2 ) ( \land \ (\exists i \in {1, \ldots, #S_1} : (\alpha_{1j} = \alpha_{2j}) \land (\forall j \in {0...((#S_1-i)/c)} : (\alpha_{1j} = \alpha_{2j}) \land P(\beta_{1i+c+j}, \beta_{2i+c+j}) ) )</td>
</tr>
</tbody>
</table>
| Pseudo-Code Representation:       | for \( i=0; i<=\text{size of } S_1[]; i++ \) { |}
|                                   | \hspace{1em} if \( P(S_1[\beta_{1i}] \) { |}
|                                   | \hspace{2em} \text{for } j=0; j<=\text{size of } S_1[]-i/c; j++ { |}
|                                   | \hspace{3em} if not \( P(S_1[\beta_{1i+c+j}] \) { |}
|                                   | \hspace{4em} Print to Screen “Anomaly Found” |}
|                                   | } |
**Pseudo-Code Representation:**

```plaintext
if size of S1[] == size of S2[] {
    for (i=0; i<=size of S1[]; i++){
        if S1[α1,i] != S2[α2,i]{
            Print to Screen “Different indexing value”
        }
        if P(S1[β1,i],S2[β2,i]){
            for (j=0; j<=(size of S1[])-i)/c; j++)
                if not P(S1[β1+i+c],S2[β2+i+c]){
                    Print to Screen “Anomaly Found”
                }
        }
    }
}
```

**Table 10.** D-SPS Bounded Response pattern formal definitions and descriptions.

---

**Bounded Response Patterns: The Boolean Statement **T** holds After Boolean Statement **P** holds At No More Than C Count of Readings.**

<table>
<thead>
<tr>
<th>Pattern Name:</th>
<th>Bounded Response_SS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pattern Type:</strong></td>
<td>Timed-Order</td>
</tr>
<tr>
<td><strong>Boolean Statement Type:</strong></td>
<td>- Single Sensor Reading (S) - Single Sensor Reading (S)</td>
</tr>
<tr>
<td><strong>English Description:</strong></td>
<td>The Boolean statement <strong>T</strong> holds over scope S_2 after Boolean statement <strong>P</strong> holds over scope S_1 at no more than t + c consecutive readings, where t is the stamp when <strong>P</strong> holds.</td>
</tr>
<tr>
<td><strong>Data Category Used By:</strong></td>
<td>- Time Dependent Data Relationship. - Time Dependent Instrument Relationship.</td>
</tr>
</tbody>
</table>
| **Formal Definition:** | c denotes time unit count (constant value). 
(\#S_1 = \#S_2) \land ((\exists i \in \{1, \ldots, \#S_1\}; (\alpha_{1,i} = \alpha_{2,i}) \land P(\beta_{1,i})) \rightarrow (\exists j \in \{i, \ldots, i + c\}; (\alpha_{1,j} = \alpha_{2,j}) \land T(\beta_{2,j})) |

**Pseudo-Code Representation:**

```plaintext
Bool flag=false
if size of S1[] == size of S2[] {
    for (i=0; i<=size of S1[]; i++){
        if S1[α1,i] != S2[α2,i]{
            Print to Screen “Different indexing value”
        }
        if P(S1[β1,i],S2[β2,i]){
            for (j=i; j<=i+c; j++)
                if not P(S1[β1+i+c],S2[β2+i+c]){
                    flag=true
                }
        }
    }
}
if (flag == false){
    Print to Screen “Anomaly Found”
}
```
**Pattern Name:** Bounded Response

**Pattern Type:**
Timed-Order

**Boolean Statement Type:**
- Multiple Sensor Reading (M)
- Single Sensor Reading (S)

**English Description:**
The Boolean statement T holds over scope S after Boolean statement P holds over scopes S and S at no more than t + c consecutive readings, where t is the stamp when P holds.

**Data Category Used By:**
- Time Dependent Data Relationship.
- Time Dependent Instrument Relationship.

**Formal Definition:**
c denotes time unit count (constant value).

\[(\#S_1 = \#S_2 \land \#S_2 = \#S_3) \land ((\exists i \in \{1, \ldots, \#S_1\}: (\alpha_{1,i} = \alpha_{2,i} \land \alpha_{2,i} = \alpha_{3,i}) \land P(\beta_{1,i}, \beta_{3,i})) \rightarrow
(\exists j \in \{i, \ldots, i + c\}: (\alpha_{1,j} = \alpha_{2,j} \land \alpha_{2,j} = \alpha_{3,j}) \land T(\beta_{2,j}))\]

**Pseudo-Code Representation:**
```plaintext
Bool flag=false
if size of S_1[] == size of S_2[] == size of S_3[]{
   for (i=0; i<=size of S_1[]; i++){
      if S_1[\alpha_{1,i}] != S_2[\alpha_{2,i}] != S_3[\alpha_{3,i}]{
         Print to Screen “Different indexing value”
      }
      else{
         if P(S_1[\beta_{1,i}], S_3[\beta_{3,i}]){ 
            for (j=i; j<=i+c; j++){
               if T(S_2[\beta_{2,j}]){ 
                  flag=true
               }
            }
          }
      }
   }
   if (flag == false){
      Print to Screen “Anomaly Found”
   }
}
```

---

**Pattern Name:** Bounded Response

**Pattern Type:**
Timed-Order

**Boolean Statement Type:**
- Single Sensor Reading (S)
- Multiple Sensor Reading (M)

**English Description:**
The Boolean statement T holds over scopes S and S after Boolean statement P holds over scope S at no more than t + c consecutive readings, where t is the stamp when P holds.

**Data Category Used By:**
- Time Dependent Data Relationship.
- Time Dependent Instrument Relationship.

**Formal Definition:**
c denotes time unit count (constant value).

\[(\#S_1 = \#S_2 \land \#S_2 = \#S_4) \land ((\exists i \in \{1, \ldots, \#S_1\}: (\alpha_{1,i} = \alpha_{2,i} \land \alpha_{2,i} = \alpha_{4,i}) \land P(\beta_{1,i}, \beta_{3,i})) \rightarrow
(\exists j \in \{i, \ldots, i + c\}: (\alpha_{1,j} = \alpha_{2,j} \land \alpha_{2,j} = \alpha_{4,j}) \land T(\beta_{2,j}, \beta_{4,j}))\]

**Pseudo-Code Representation:**
```plaintext
Bool flag=false
if size of S_1[] == size of S_2[] == size of S_3[]{
   for (i=0; i<=size of S_1[]; i++){
      if S_1[\alpha_{1,i}] != S_2[\alpha_{2,i}] != S_4[\alpha_{4,i}]{
         Print to Screen “Different indexing value”
      }
   }
```
if \( P(S_1[β_{1,i}]) \) {
    for (j=i; j<=i+c; j++) {
        if \( T(S_2[β_{2,j}], S_4[β_{4,i}]) \) {
            flag=true
        }
    }
    if (flag == false) {
        Print to Screen “Anomaly Found”
    }
}

Table 11. D-SPS Bounded Invariance pattern formal definitions and descriptions.

<table>
<thead>
<tr>
<th>Pattern Name: Bounded Response\textsubscript{MM}</th>
<th>Pattern Type: Timed-Order</th>
<th>Boolean Statement Type:</th>
</tr>
</thead>
<tbody>
<tr>
<td>English Description:</td>
<td></td>
<td>- Multiple Sensor Reading (M)</td>
</tr>
<tr>
<td>The Boolean statement ( T ) holds over scopes ( S_2 ) and ( S_4 ) after Boolean statement ( P ) holds over scopes ( S_1 ) and ( S_3 ) at no more than ( t + c ) consecutive readings, where ( t ) is the stamp when ( P ) holds.</td>
<td></td>
<td>- Multiple Sensor Reading (M)</td>
</tr>
<tr>
<td>Data Category Used By:</td>
<td></td>
<td>- Time Dependent Data Relationship.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Time Dependent Instrument Relationship.</td>
</tr>
<tr>
<td>Formal Definition:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( c ) denotes time unit count (constant value).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( (#S_1 = #S_2 \land #S_3 = #S_4) \land (\forall i \in {1, \ldots, {#S_1}}: (α_{1,i} = α_{2,i} \land α_{2,j} = α_{3,j} \land α_{3,j} = α_{4,j}) \land P(β_{1,i}, β_{3,i})) \rightarrow (∃j \in {i, \ldots, i + c}: (α_{1,j} = α_{2,j} \land α_{2,j} = α_{3,j} \land α_{3,j} = α_{4,j}) \land T(β_{2,j}, β_{4,j})) ))</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pseudo-Code Representation:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bool ( flag = ) false</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if size of ( S_1[] ) == size of ( S_2[] ) == size of ( S_3[] ) == size of ( S_4[] )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for (i=0; i&lt;=size of ( S_1[] ); i++)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if ( S_1[α_{1,i}] \neq S_2[α_{2,i}] \neq S_3[α_{3,i}] \neq S_4[α_{4,i}] )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print to Screen “Different indexing value”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if ( P(S_1[β_{1,i}], S_3[β_{3,i}]) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>for (j=i; j&lt;=i+c; j++)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if ( T(S_2[β_{2,j}], S_4[β_{4,i}]) )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>flag=true</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>if (flag == false) {</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Print to Screen “Anomaly Found”</td>
<td></td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 11. D-SPS Bounded Invariance pattern formal definitions and descriptions.
**Timed-Order**

**Single Sensor Reading (S)**

**English Description:**
The Boolean statement T holds over scope S₁ for at least t + c consecutive readings before Boolean statement P holds over scope S₂, where t is the stamp when T holds.

**Data Category Used By:**
- Time Dependent Data Relationship.
- Time Dependent Instrument Relationship.

**Formal Definition:**
c denotes time unit count (constant value).

\[(\#S₁ = \#S₂) \land (\exists i \in \{1, \ldots, \#S₁\} : (α₁,i = α₂,i) \land (\forall j \in \{i, \ldots, i + c\} : (α₁,j = α₂,j) \land (T(β₁,j) \rightarrow (\neg P(β₂,j) \land P(β₂,j+c+1)))))]

**Pseudo-Code Representation:**

```plaintext
Bool holds=false
if size of S₁[] == size of S₂[]{
  for (i=0; i<=size of S₁[]; i++){
    if S₁[α₁,i] != S₂[α₂,i]{
      Print to Screen “Different indexing value”
    }
    if T(S₁[β₁,i]){
      for (j=i; j<=i+c; j++){
        if not T(S₁[β₁,j]){
          Print to Screen “Anomaly Found”
        }
      }
      for (k=i+c; k< size of S₁[]; k++){
        if P(S₂[β₂,k]) == true{
          holds=true
        }
      }
      if (holds == false){
        Print to Screen “Anomaly Found”
      }
    }
  }
}
```

**Pattern Name:** Bounded Invariance<sub>MS</sub>

**Pattern Type:**
Timed-Order

**Boolean Statement Type:**
- Multiple Sensor Reading (M)
- Single Sensor Reading (S)

**English Description:**
The Boolean statement T holds over scope S₂ after Boolean statement P holds over scopes S₁ and S₃ at no more than t + c consecutive readings, where t is the stamp when P holds.

**Data Category Used By:**
- Time Dependent Data Relationship.
- Time Dependent Instrument Relationship.

**Formal Definition:**
c denotes time unit count (constant value).

\[(\#S₁ = \#S₂ \land \#S₂ = \#S₃) \land (\exists i \in \{1, \ldots, \#S₁\} : (α₁,i = α₂,i \land α₂,i = α₃,i) \land (\forall j \in \{i, \ldots, i + c\} : (α₁,j = α₂,j \land α₂,j = α₃,j) \land (T(β₁,j, β₃,j) \rightarrow (\neg P(β₂,j) \land P(β₂,j+c+1))))]

---

49
### Pseudo-Code Representation:

```plaintext
Bool holds=false
if size of S_1[] == size of S_2[] == size of S_3[]{
    for (i=0; i<=size of S_1[]; i++){
        if S_1[α_1,i] != S_2[α_2,i] != S_3[α_3,i]{
            Print to Screen “Different indexing value”
        }
        if T(S_1[β_1,j], S_2[β_3,j]){
            for (j=i; j<=i+c; j++){
                if not T(S_1[β_1,j], S_3[β_3,j]) {
                    Print to Screen “Anomaly Found”
                }
            }
            for (k=i+c; k< size of S_1[]; k++){
                if P(S_2[β_2,k]){
                    holds=true
                }
            }
            if (holds == false){
                Print to Screen “Anomaly Found”
            }
        }
    }
}
```

### Pattern Name: Bounded Invariance_{SM}

#### Pattern Type: Timed-Order

#### Boolean Statement Type:
- Single Sensor Reading (S)
- Multiple Sensor Reading (M)

#### English Description:
The Boolean statement T holds over scopes S_1 for at least t + c consecutive readings before Boolean statement P holds over scopes S_2 and S_3 where t is the stamp when T holds.

#### Formal Definition:
c denotes time unit count (constant value).

\[ (#S_1 = #S_2 \land #S_2 = #S_4) \land (\exists i \in \{1, \ldots, #S_1\}: (\alpha_{1,i} = \alpha_{2,i} \land \alpha_{3,i} = \alpha_{4,i}) \land \\
(\forall j \in \{i, \ldots, i+c\}: ((\alpha_{1,j} = \alpha_{2,j} \land \alpha_{3,j} = \alpha_{4,j}) \land \\
T(\beta_{1,j}) \rightarrow (\neg P(\beta_{2,j}, \beta_{4,j}) \land \\
P(\beta_{2,j+c+1}, \beta_{4,j+c+1})))) \]

#### Data Category Used By:
- Time Dependent Data Relationship.
- Time Dependent Instrument Relationship.
for (k=i+c; k< size of \(S_1[]\); k++)
    if P(\(S_2[\beta_{2k}], S_4[\beta_{4k}]\))
        holds=true
    
for (j=i; j<=i+c; j++)
    if not T(\(S_1[\beta_{3j}], S_3[\beta_{3j}]\))
        Print to Screen “Different indexing value”

if T(\(S_1[\beta_{3j}], S_3[\beta_{3j}]\))
    for (j=i+1; j<=i+c; j++)
        if not T(\(S_1[\beta_{3j}], S_3[\beta_{3j}]\))
            Print to Screen “Anomaly Found”

for (k=i+c; k< size of \(S_1[]\); k++)
    if P(\(S_2[\beta_{2k}], S_4[\beta_{4k}]\))
        holds=true
    
if (holds == false)
    Print to Screen “Anomaly Found”

\(c\) denotes time unit count (constant value).
\((\#S_1 = \#S_2 = \#S_3 = \#S_4)\land (\exists i \in \{1, \ldots, \#S_1\}: (\alpha_{1,i} = \alpha_{2,i} \land \alpha_{2,i} = \alpha_{3,i} \land \alpha_{3,i} = \alpha_{4,i}) \land (\forall j \in \{i, \ldots, i+c\}: (\alpha_{1,j} = \alpha_{2,j} \land \alpha_{2,j} = \alpha_{3,j} \land \alpha_{3,j} = \alpha_{4,j}) \land (T(\beta_{1,j}, \beta_{3,j}) \rightarrow (\neg P(\beta_{2,j}, \beta_{4,j}) \land P(\beta_{2,j+c+1}, \beta_{4,j+c+1}))))\)
4.3 Specification of Data Properties Using D-SPS

Data Property Specification Syntax

A data property can be specified by selecting a data property category, and by defining a scope and a pattern. The data property specification can then be interpreted and used to automatically generate executable code that can be used to identify anomalies in the data. The syntax to specify a data property is described in Table 12; in the grammar, literal terminals are **bolded**, and non-terminals are given in *italics*. The literal terminals for data categories, scopes, and patterns, correspond to the definitions in the previous section.

**Table 12.** Extended Backus-Naur Form syntax description used to build D-SPS properties.

<p>| 1 | property ::= | data_category, scope, pattern_type |
| 2 | data_category ::= | Datum | Instrument | Data Relationship | Instrument Relationship | Time-Dependent Datum | Time-Dependent Instrument | Time-Dependent Data Relationship | Time Dependent Instrument Relationship | Instrument-Dependent Datum | Data-Dependent Instrument |
| 3 | pattern_type ::= | time_depend | not_time_depend |
| 4 | time_depend ::= | t_single_rel_cat | t_composite_rel_cat |
| 5 | not_time_depend ::= | nt_single_rel_cat | nt_composite_rel_cat | nt_hybrid_cat |
| 6 | t_single_rel_cat ::= | t_datum | t_istrument |
| 7 | t_composite_rel_cat ::= | t_data_relationship | t_instrument_relationship |
| 8 | nt_single_rel_cat ::= | nt_datum | nt_istrument |
| 9 | nt_composite_rel_cat ::= | nt_data_relationship | nt_instrument_relationship |
| 10 | nt_hybrid_cat ::= | nt_instrument_dep_datum | nt_data_dep_instrument |
| 11 | t_datum ::= | Minimum DurationS(boolstatssingle, number) | Maximum DurationS(boolstatssingle, number) | Bounded RecurrenceS(boolstatssingle, number) |
| 12 | t_istrument ::= | Minimum DurationS(boolstatssingle, number) | Maximum DurationS(boolstatssingle, number) | Bounded RecurrenceS(boolstatssingle, number) |
| 13 | scope ::= | Global | Before(del) | After(del) | Between(del,del) | AfterUntil(del,del) |
| 14 | del ::= | numeric_value | time_stamp |
| 15 | numeric_value ::= | number |
| 16 | time_stamp ::= | month-day-year hours:minutes:seconds:milliseconds |
| 17 | month ::= | 01/02/...09/10/11/12 |
| 18 | day ::= | 01/02/...09/10/11/31 |
| 19 | year ::= | 000000001...12050 |
| 20 | hours ::= | 01/02/...09/10/12/23 |
| 21 | minutes ::= | 01/02/...09/10/15/59 |
| 22 | seconds ::= | 01/02/...09/10/15/59 |
| 23 | milliseconds ::= | 01/02/...19 |
| 24 | t_data_relationship ::= | Minimum DurationM(boolstatssmult, number) | Maximum DurationM(boolstatssmult, number) | Bounded RecurrenceM(boolstatssmult, number) | Bounded ResponseSS(boolstatssmultss, number) | Bounded ResponseMS(boolstatssmultms, number) | Bounded ResponseSM(boolstatssmultsm, number) |</p>
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>t_instrument_relationship ::=</td>
</tr>
<tr>
<td>26</td>
<td>nt_datum ::=</td>
</tr>
<tr>
<td>27</td>
<td>nt_instrument ::=</td>
</tr>
<tr>
<td>31</td>
<td>nt_data_dep_instrument ::=</td>
</tr>
</tbody>
</table>
4.3.2 Data Property Specification Examples

To illustrate the strengths of the D-SPS consider the following natural language data properties obtained from the literature review:

“For May 12th at daytime, the dry bulb temperature should be smaller than or equal to 35.0 degrees Celsius” [48].

A scientist using the D-SPS to specify the data property selects Datum as the data property category because she is interested in evaluating only one set of sensor data. The scientist then specifies the scope for the data property. The desired scope encompasses the data readings recorded on May 12th during daytime, i.e., between 6:15:00 AM and 8:00:00 PM, as defined by the project’s documentation available to the scientist. The scientist uses Between L and R as the scope, where L stands for time stamp 05-12-2010 06:15:00.0 and R stands for time stamp 05-12-2010 20:00:00.0. The scientist examines the available documentation associated with the sensor readings of interest, and identifies temp as the variable containing the dry bulb temperature reading in degrees Celsius. The scientist defines a Boolean statement to be temp ° ≤ 35°C. The scientist then selects Universality s as the pattern to evaluate if it is always the case, over the scope of data reading values, that temp ° ≤ 35°C. The high level specification using D-SPS is presented in Table 13 and the derivation tree in Figure 7.
Table 13. High-level D-SPS specification for a dry-bulb temperature property.

<table>
<thead>
<tr>
<th>Document Specification</th>
<th>“On May 12th during the daytime, the dry bulb temperature should be smaller than or equal to 35.0 °C “[73].</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-SPS Representation</td>
<td>Datum, Between(05-12-2010 06:15:00.0, 05-12-2010 20:00:00.0), UniversalityS(temp; °C, ≤ 35; °C)</td>
</tr>
<tr>
<td>Assumptions</td>
<td>Daytime: time-frame between 05:12:2010:06:15:00 and 05:12:2010:20:00:00 from scientific expert knowledge. temp: dry bulb temperature sensor dataset.</td>
</tr>
</tbody>
</table>

Figure 7. D-SPS specification derivation tree for a dry-bulb temperature property.

The D-SPS specification can be represented using the pseudo-code definitions presented in the previous section. If it is assumed that the data structure \( D \), used to store the sensor datasets, provides the method `indexOf(alpha:String)` that returns the index of the first occurrence of the sensor reading indexed by the value `alpha`, then the pseudo-code representation in Figure 8 can be generated.

**Pseudo-Code Representation:**

```
D[]; // all temp readings
L= D[].indexOf(05-12-2010 06:15:00.0);
R= D[].indexOf(05-12-2010 20:00:00.0);
S_i[] = D[L]...D[R];
```
for (i=0; i<=size of S[i]; i++){
    if not P(S[i].temp ≤ 35){
        Print to Screen “Anomaly Found”
    }
}

**Figure 8.** Pseudo-code representation for a dry-bulb temperature property.

D-SPS also allows scientists to specify properties that capture problems in the equipment. For example, consider the data property that captures the effect of temperature over an equipment diagnostic value (agc) that is miscalculated whenever high temperatures occur during the summer:

“For all dry bulb temperature dataset values, it is always the case that, If temp>35 °C, then agc<70” [48].

The data property can be captured using D-SPS as follows. The scientist is interested in evaluating two related sensor datasets, thus she selects *Data Relationship* as the data type category for the property. The property will evaluate all of the *temp* readings in the dataset, thus the scientist selects *Global* as the scope. The scientist realizes that whenever dry bulb temperature temp is greater than 35°C, the *agc* diagnostic flag should be less than 70; otherwise, if temp is greater than 35 and the *agc* value is also greater than 70, then *agc* is likely being miscalculated possibly due to the effect of the high temperature over the equipment in the remote research location. The scientist uses the *Response(T,P)* pattern to capture the ordering of the Boolean statements *T* and *P*. Boolean statement *T* evaluates whether the *temp>*35 °C and Boolean statement *P* evaluates whether *agc>*70. The *Response(T,P)* pattern evaluates if is always the case that when *temp>*35 °C then *agc>*70 is also true. The high level specification using D-SPS is presented in table 14 and the derivation tree in Figure 9:

**Table 14.** High-level D-SPS specification for a diagnostic sensor property.

<table>
<thead>
<tr>
<th>Document Specification:</th>
<th>“For all dry bulb temperature dataset values, it is always the case that, If temp&gt;35 °C, then agc&lt;70.0.”[73]</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-SPS Representation:</td>
<td>Global,ResponseS(temp:°C,&gt;,35:°C, agc:n/a,&lt;,70.0:n/a)</td>
</tr>
<tr>
<td>Assumptions:</td>
<td><em>temp</em>: dry bulb temperature sensor dataset.</td>
</tr>
<tr>
<td></td>
<td><em>agc</em>: diagnostic flagging sensor dataset.</td>
</tr>
</tbody>
</table>
“For all dry bulb temperature dataset values, it is always the case that, if TEMP>35 °C, then AGC<70.0”

Data Relationship, Global, ResponseSS(TEMP:°C,>35:°C, AGC:n/a,<70.0:n/a)

---

**Figure 9.** D-SPS specification derivation tree for a diagnostic property.

Similarly to the property in Figure 10, The D-SPS specification can be represented using the pseudo-code definitions presented in the previous section.

**Pseudo-Code Representation:**

```c
D_1[]; //all temp readings
D_2[]; //all agc readings

S_1[] = D_1[];
S_2[] = D_2[];

Bool flag = false
if size of S_1[] == size of S_2[] {
    for (i=0; i<=size of S_1[]-1; i++){
        if S_1[i].alpha != S_2[i].alpha{
            Print to Screen “Different indexing value”
        }
    }
}
```
if T(S_1[i].temp >35){
    if P(S_2[i].agc<70.0){
        flag=true
    }
    if P(S_2[i+1].agc<70.0){
        flag=true
    }
} 
if (flag == false){
    Print to Screen “Anomaly Found”
}
}

Figure 10. Pseudo-code representation for a dry-bulb temperature property with diagnostic flags.

The pseudo-code representations can be interpreted, implemented, and used to identify anomalies in scientific sensor data.
Chapter 5: Data Property Specification and Verification Tool Support

5.1 Data Property Specification Tool Support Overview

The data property categorization and the D-SPS resulted in development of Data Property Specification (DaProS), a scientist-centered prototype tool that uses the data property categorization, developed as a component of this dissertation, to assist the user in specifying a data property. Through a series of guiding questions and selections, the user identifies the appropriate category and enters required information, and the tool yields the appropriate specification as well as a disciplined natural language representation of the specification for validation purposes. The DaProS prototype tool was successfully used to formally specify the data properties from the literature review and identify recurrent practices followed by scientists when specifying data properties.

5.2 Data Property Specification (DaProS) Tool Interface

The DaProS graphical user interface is composed of five interface windows as depicted in Illustration 1. The functionality of each screen is discussed next.

Main Window: The main window drives the data property specification process and is used to specify the property scope, the property pattern, and the Boolean statements. The main window also allows the scientists to document assumptions and comments about the data property, and to select a final property view as described earlier. In addition, the main window provides access to the master console and the property selection oracle.

Master Console: The master console window provides an overall summary of the system and the status of data property as it is being specified. The master console also provides the definitions associated with property scopes and patterns as they are being selected by the user.

Property Selection Oracle: The property selection Oracle window guides scientists in identifying the data property category that is best suited for specifying the desired data property. The property selection Oracle uses decisions trees and guiding questions to help scientists select a data property category for the property being specified.

Data Property Categorization Diagram: When called by the property selection oracle, the data property categorization window presents a visual representation of the relationships between the data
property categories, the patterns, and the number and types of Boolean statements evaluated by the
patterns as used by DaProS.

**Oracle Console:** The oracle console provides the scientists with hints associated with the current
state of the property selection process; such hints are automatically related by the Oracle to each stage of
the specification process.

Illustration 1: DaProS graphical user interface.
5.3 Data Property Specification Through DaProS

The property specification process in DaProS is composed of five steps as guided by the user interface of the main window shown in Illustration 2. The five steps followed to specify a property are: 1) property category selection, 2) property scope selection, 3) property pattern selection and specification, 4) assumptions and comments input, and 5) final property view. The steps are detailed in the following subsections.

Illustration 2: DaProS main property specification graphical user interface.

5.3.1 Data Property Category Selection

The initial step in the specification process is for the scientist to select a data property category. The data property categories help scientists determine if the purpose of the property is to define expected data behavior or instrument behavior. The data property categories also determine if the property is time-dependent or not. DaProS uses the defined relationships between data property categories and D-
SPS patterns to restrict the type and number of patterns and Boolean statements builders that are available to the user given a category selection. If the scientist is undecided over what data property to select, he or she can use the data property category selection oracle and decision trees to select the data property category.

The DaProS property selection oracle guidance to the user is based on data properties decision trees as presented in Illustration 3. The oracle asks the scientists a series of questions that help narrow the number of data properties options until a suitable data property category is identified.

**Illustration 3:** DaProS property selection oracle.

Figure 11 presents the data property decision tree for experimental readings, i.e., data readings, and Figure 12 presents the data property decision tree for experimental conditions, i.e., instrument functioning. In both trees, the decision tree begins by determining whether the intended property
specification is an experimental reading or an experimental condition. Once that is determined, it is necessary to determine if the property is time constrained or not. If the property is time constrained, then the scientist must decide whether he/she is interested in a evaluating a single sensor dataset or a relationship between two or more sensor datasets. If the property is not time constrained, then the scientist needs to determine if the property depends on other behavior, i.e., determine if the data depends on an instrument(s) functioning, if instruments depend on data behavior, or neither. If the property is free of any behavior dependencies then the scientist needs to decide if he/she is interested in a single sensor dataset or a relationship between two or more sensor dataset. Following the guidance, the scientist can decide a suitable data category type to specify the desired data property.

Figure 11. Data property category decision tree for experimental readings.
5.3.2 Data Property Scope Selection

The scientist selects a property scope and provides the attributes associated with the scope if needed. For time-indexed scopes requiring delimiters, the scientist provides the unique time-stamps indexing values needed to delimit the dataset of sensor readings of interest. For scopes indexed by numerical values and requiring delimiters, the scientist provides the unique indexing numerical values to delimit the dataset of interest. Because this work assumes that the data scopes to be evaluated are indexed by the same values, DaProS interface allows the scientists to only input a single left delimiter and a single right delimiter, if needed. It is assumed that the provided delimiters demarcate all the scopes in the data property specification. If the scientist is undecided on the format of the delimiters, he/she can use the \textit{find variables} functionality in the DaProS menu. Given a sample data file over which the data
property would be evaluated, the *find variables* functionality analyzes the data file’s header file, when available, and returns to the user a list of the sensor used to collect the data, the measurements units and the format of the indexing values. The scientist can then use the extracted format of the indexing values to define the delimiters for the scope.

### 5.3.3 Property Pattern Selection and Specification

The scientist specifies the pattern by first selecting the pattern type that describes how the Boolean statement(s) will be evaluated and by constructing the required Boolean statements.

If the selected data property category from Step 1 is indexed by timed values, the scientist must select a time-constrained property pattern; otherwise, the scientist selects a qualitative property pattern. DaProS’s interface guides the user in the property-pattern selection. If the property is not constrained by time, the scientist proceeds to *Step 3a*. If the property to be specified is time-constrained, the scientist proceeds to *Step 3b*.

In *Step 3a*, the scientist selects a property pattern that is not constrained by time. The data property patterns options available to the scientist are restricted by DaProS based on the scientists’ previous data property category selection and the D-SPS definitions. Given a pattern selection, the scientist builds either one or two Boolean statement(s) that can evaluate one to two sensor datasets, respectively. The number of sensor reading datasets for each Boolean statement is also limited by the scientist’s pattern choice and the D-SPS pattern definitions. The scientist uses the Boolean statement builder in DaProS to build the Boolean statement(s) required by the pattern definition. The scientists initially input the names of the sensors to be evaluated. The name of the sensors must match the names in the data file over which the data property will be evaluated. If the scientist is undecided on the name of the sensor to use, he/she can use the *find variables* functionality in the DaProS menu to retrieve the sensor names. The scientist must also provide the measurement units associated with the sensors evaluated. DaProS performs a syntactic comparison between the scientist-provided measurement units of the sensor datasets being related to ensure that the units are equivalent. If the scientist is undecided about what measurement units are used in the data file, he/she can use the *find variables* functionality in the DaProS menu to retrieve the measurement units for a given sensor. Finally, the scientist must bound
the sensor datasets in each Boolean statement by using one or two of the following basic set of relational operators: $<, \leq, =, \neq, \geq, >$.

In Step 3b, the scientist selects a property pattern that is constrained by time. The data property patterns options to the scientist are restricted by DaProS based on the scientists’ previous data property category selection and the D-SPS definitions. Given a pattern selection, the scientist builds either one or two Boolean statement(s) to evaluate the respective sensor datasets. The number of sensor reading datasets for each Boolean statement is also limited by the scientist’s pattern choice and the D-SPS pattern definitions. The scientist uses the Boolean statement builder in DaProS to build the Boolean statement(s) required by the pattern definition. The scientists initially input the names of the sensors to be evaluated. The name of the sensors must match the names in the data file over which the data property will be evaluated. If the scientist is undecided on the name of the sensor to use, he/she can use the *find variables* functionality in the DaProS menu to retrieve the sensor names. The scientist must also provide the measurement units associated with the sensors evaluated. DaProS performs a syntactic comparison between the scientist-provided measurement units of the sensor datasets being related to ensure that the units are equivalent. If the scientist is undecided about what measurement units are used in the data file, he/she can use the *find variables* functionality in the DaProS menu to retrieve the measurement units for a given sensor. In addition, the scientist must bound the sensor datasets in each Boolean statement by using one or two of the following basic set of relational operators: $<, \leq, =, \neq, \geq, >$.

Finally, the scientist provides a numerical value to represent the time constant associated with time-related patterns.

### 5.3.4 Assumptions and Comments (optional step)

Some scientific communities require data-related artifacts to include metadata that document the processes and sources associated with such artifacts. Metadata, i.e., well-defined and structured data that documents information about other data, can be used to capture sources, methodologies, and assumptions associated with a data property. Even though metadata formats for data properties are not in place yet, it is still beneficial to capture assumptions and comments associated with the data properties created using DaProS. DaProS allows scientists to input the sources, methodologies and assumptions...
considered when specifying data properties. The term metadata is used in this work to denote the documentation of sources, methodologies, and assumptions associated with the data property. The capture of assumptions and comments is optional and has no effect on the data property specification components other than helping the scientist better understand the data property. The captured assumptions and comments can be used to document the context for a data property, to mitigate the linguistic ambiguity and vagueness of concepts associated with the sensor readings in a data property, and to document sources and reasoning supporting a data property. The data verification mechanisms do not evaluate the assumptions and comments definitions.

5.3.5 Final Property View

In this step, the scientist selects a format to display the property. The scientist can generate a property summary, a disciplined natural language description, or as an Extensible Markup Language (XML) representation. The property summary provides an overall view of the property that includes the selected data property category, the property scope along with the delimiters, if needed, and the selected pattern along with the corresponding Boolean statement(s). The disciplined natural language (DNL) [29] description is a natural language description that can be used by scientists to review and validate the property by determining if the natural language captures the intended meaning of the property. The use of disciplined natural language intends to mitigate the ambiguity inherent to natural languages. DaProS tool uses disciplined natural language (DNL) templates to generate the natural language descriptions of specifications. A DNL grammar was constructed based on the work initiated by Konrad and Chen [24]. Table 15 presents the DaProS grammar used to derive natural language property representations, and Tables 5 and 6 present examples along with the respective derivation trees in Figures 11 and 12. In the grammar, literal terminals are bolded, and non-terminals are given in italics.
Table 15. DaProS DNL grammar for validating property representations.

<table>
<thead>
<tr>
<th></th>
<th>property ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>data_category, scope, pattern_type</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>data_category ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Given a(n) “data_cat” category:</td>
</tr>
<tr>
<td>3</td>
<td>data_cat ::=</td>
</tr>
<tr>
<td></td>
<td>Datum</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>scope ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>global \ beforeR \ afterL \ betweenLandR \ afterLuntilR</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>global ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>for all dataset values</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>beforeR::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>for all dataset values before del</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>afterL::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>for all dataset values after del</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>betweenLandR::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>for all dataset values between betdel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>afterLuntilR::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>for all dataset values after aftdel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>del ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>numeric_value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>betdel ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>numeric_value and numeric_value</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>aftdel ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>numeric_value until numeric_value or until the end of the dataset if numeric_value does not occur time_stamp until time_stamp or until the end of the dataset if time_stamp does not occur</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>numeric_value ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>number.number</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>time_stamp ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>month-day-year hours:minutes:seconds.milliseconds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>number::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>month::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0102...09101112</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>day::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>0102...0910...31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>year::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>00000001...2050</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>hours::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>0102...0910...23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>minutes::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>0102...0910...59</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>seconds::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>0102...0910...59</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>milliseconds::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>123...9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>pattern_type::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>absence</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>absence::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>it is never the case that (boolstatingle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>universality::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>it is always the case that (boolstatingle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>existence::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>(boolstatingle</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>precedence::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>26</td>
<td>it is always the case that if boolstatingle holds then boolstatingle eventually holds.</td>
</tr>
<tr>
<td></td>
<td>it is always the case that if boolstatingle holds then boolstatmult eventually holds.</td>
</tr>
<tr>
<td></td>
<td>it is always the case that if boolstatmult holds then boolstatingle eventually holds.</td>
</tr>
<tr>
<td></td>
<td>it is always the case that if boolstatmult holds then boolstatmult eventually holds.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>response ::=</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>it is always the case that if boolstatingle holds then boolstatingle immediately holds.</td>
</tr>
<tr>
<td></td>
<td>it is always the case that if boolstatingle holds then boolstatmult immediately holds.</td>
</tr>
<tr>
<td></td>
<td>it is always the case that if boolstatmult holds then boolstatingle immediately holds.</td>
</tr>
<tr>
<td></td>
<td>it is always the case that if boolstatmult holds then boolstatmult immediately holds.</td>
</tr>
</tbody>
</table>
The grammar in Table 15 can be used to describe the properties obtained from the literature review. Table 16 and Figure 13 present the disciplined natural language representation and the specification derivation tree respectively for a datum property. Similarly, Table 17 and Figure 14 present the disciplined natural language representation and the specification derivation tree respectively for a data relationship property.
Table 16. DaProS disciplined natural language representation for an experimental reading property.

<table>
<thead>
<tr>
<th>Document Specification</th>
<th>“On May 12th during the daytime, the dry bulb temperature should be less than or equal to 35.0 °C “[73].</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-SPS Representation:</td>
<td>Datum, Between(05-12-2010 06:15:00.0, 05-12-2010 20:00:00.0), UniversalS(temp:°C, ≤ ,35:°C)</td>
</tr>
<tr>
<td>DaProS DNL Representation:</td>
<td>Given a(n) “Datum” category: for all dataset values between 05-12-2010 06:15:00.0 and 05-12-2010 20:00:00.0, it is always the case that temp °C ≤ 35.0 °C holds.</td>
</tr>
<tr>
<td>Assumptions:</td>
<td>Daytime: time-frame between 05:12:2010:06:15:00 and 05:12:2010:20:00:00 from scientific expert knowledge. temp: dry bulb temperature sensor dataset.</td>
</tr>
</tbody>
</table>

Figure 13. Disciplined natural language specification derivation tree for a datum property.

Table 17. DaProS disciplined natural language representation for a data relationship property.

<table>
<thead>
<tr>
<th>Document Specification:</th>
<th>“For all dry bulb temperature dataset values, it is always the case that, if temp&gt;35 °C, then agc&lt;70.0.” [73]</th>
</tr>
</thead>
<tbody>
<tr>
<td>D-SPS Representation:</td>
<td>Data Relationship,Global, ResponseS(temp:°C, &gt; ,35, °C, agc:n/a,&lt;,70,0:n/a)</td>
</tr>
</tbody>
</table>
**DaProS DNL Representation:**

Given a(n) "Data Relationship" category: for all dataset values, it is always the case that if \( \text{temp} \, ^{\circ}C > 35 \, ^{\circ}C \) holds, then \( \text{agc} < 70.0 \) immediately holds.

**Assumptions:**

- \( \text{temp} \): dry bulb temperature sensor dataset.
- \( \text{agc} \): diagnostic flagging sensor dataset.

---

"For all dry bulb temperature dataset values, it is always the case that, if TEMP>35 \(^{\circ}C\), then AGC<70.0"  

Given a(n) "Data Relationship" category, for all dataset values it is always the case that if TEMP:°C > 35:°C holds then AGC:n/a < 70.0:n/a immediately holds.

**Figure 14.** Disciplined natural language specification derivation tree for a data relationship property.

The XML representation is to be used to export the specified properties from DaProS to the data anomaly detection system described in Section 6.3.2. XML was selected because it is portable, extensible, and it is embraced by the scientific communities that participate of this work.

**5.3.6 Findings**

Properties obtained from the literature survey were specified using DaProS, and the DNL representation generated from the grammar in Table 3 was captured. DaProS was capable of specifying all of the data properties, except for six of them. Those six properties were ambiguous and, as a result, it
was difficult to identify the appropriate scope and pattern using the tool based only on the statement as provided in the documentation. However, further property refinement and expert-scientist knowledge would allow the scientist to specify the properties using DaProS. Table 18 presents the properties that were not specified and the issues associated to each property. In addition, the specification exercise also identified the need for the formalization and establishment of a standardized, well-structured, data property specification metadata schema to document data properties.

Table 18. Data properties that could not be specified using DaProS.

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flag as K if the data looks to have obvious errors, but no specific reason for the error can be determined.</td>
<td>“Obvious” is ambiguous.</td>
</tr>
<tr>
<td>Flag as M if a known instrument malfunction occurs.</td>
<td>This can be specified if the instruments are provided. In this, they were not given.</td>
</tr>
<tr>
<td>Flag as Z if data passed evaluation.</td>
<td>“Evaluation” was not specified.</td>
</tr>
<tr>
<td>A Fine Structure consists of step-like features or small interleaving observed in a profile over a range of depths, (usually 10-100 m) or the entire profile.</td>
<td>There is no specification for the features. Also, it is not clear if the data is to be verified over the range of depths, over the entire profile, or both. Is there a difference at 10m vs 100m?</td>
</tr>
<tr>
<td>The date and time must be sensible.</td>
<td>“Sensible” is ambiguous.</td>
</tr>
<tr>
<td>All data must be corrected with a calibration accuracy of +/-1.0% at up to 20 mm/hr.</td>
<td>This can be verified only if the original value is saved along with the corrected value. Should the verification include those corrected at 20mm/hr?</td>
</tr>
<tr>
<td>All data must be corrected with a calibration accuracy of MAX (+/-1.1 m/sec (2.4 mph), +/-4% of reading).</td>
<td>This can be verified only if the original value is saved along with the corrected value.</td>
</tr>
</tbody>
</table>

5.4 DATA PROPERTY VERIFICATION TOOL SUPPORT OVERVIEW

The sensor data verification (SDVe) prototype tool was developed to allow scientists to evaluate scientific sensor datasets against specified data properties at collection time, or as soon as the data is available from the data logger in the field. The SDVe tool takes as input a property specification file and a dataset file and verifies that the data in the dataset files adheres to the property specified in the property specification file. The tool supports reusable properties that do not require the evaluation tool’s source code to be modified or recompiled. Such reusable properties are of interests to scientists because quality control is always a combination of different levels of control and site-specific tests that will differ from one site to another because of the topography. The SDVe tool raises alarms whenever a
sensor reading does not satisfy the specified data property. The SDVe is not intended to characterize the reasons for an anomaly in the data, but to inform scientists about possible problems in the data so that the scientists can decide whether the raised alarm signals a legitimate environmental event, or if it is an error in the data caused by a malfunctioning instrument. The reason for this decision is that even though physically plausible environmental event and instrument problems can overlap, automated techniques could not equivocally pinpoint differences between the two [49]; subjective analyses do better than automated programs during unusual conditions. In addition, data flagged as anomalies, but later deemed physical after graphical inspection, are often found to be the most unusual and interesting situations worthy of closer investigation. The SDVe prototype tool was successfully used, as described in Section 6.3, to identify anomalies in scientific sensor data obtained from sensors in an Eddy covariance tower.

5.5 **SENSOR DATA VERIFICATION (SDVe) TOOL COMPOSITION**

The sensor data verification tool, as shown in Figure 15, is composed of five components: the *dataset parser*, the *specification parser*, the *scope processor*, the *Boolean function builder*, and the *pattern processor*. 
The **dataset parser** module encompasses a group of sensor dataset file parsers for supported data files. Even though SDVe is intended to support a number of different sensor data files, a dataset parser for such data file should be developed and placed in this module. However, this is the only change that has to be performed to the SDVe. The dataset parser reads a dataset file and transforms the dataset into an internal representation of the dataset that can be used by the scope processor to extract the data subsequences to be evaluated. The sensor data readings are stored as pairs indexed by an identifier, i.e. a unique time-stamp or numeric value associated to the sensor reading.

The **specification parser** module reads a DaProS-generated specification file and extracts the attributes needed to delimit the dataset scopes, to build the Boolean statements to be evaluated over the data, and to determine how to apply the pattern to the dataset.

The **scope processor** module uses the internal dataset representation populated by the dataset parser module and extracts the data subsequences over which the Boolean statements will be evaluated.
The **Boolean statement builder** module uses the Boolean statements attributes extracted by the specification parser module and determines the type of operation, the inputs, and the Boolean function code templates to evaluate the sensor readings in the scope(s). SDVe supports the mathematical operators \(<\), \(\leq\), \(=\), \(\neq\), \(\geq\), and \(>\).

The **pattern processor** module uses the Boolean functions created by the Boolean function builder module, the scopes from the scope processor module and the pattern attributes from the specification parser module to evaluate the pattern. The pattern processor evaluates the Boolean functions over the scopes of sensor data as specified by the pattern’s attributes, and raises alarms when the Boolean functions evaluations over the scopes do not satisfy the expected data property. To evaluate the Boolean statements over the scopes, SDVe interprets the DaProS specifications into code templates that evaluate the data. Each pattern code template in SDVe corresponds to a pattern in the D-SPS. The code templates have a predefined number of Boolean statements that can be evaluated by the pattern, and a predefined number of scopes that can be evaluated by each Boolean statement. Once the patterns have been evaluated, SDVe generates two output files to document the violations of properties identified in the data.

The **verification summary file** includes the total results of the verification process over several sensor data files. The verification summary is used to provide an overall overview of the verification process. Figure 16 provides a snapshot of a verification summary file. Each data property evaluated by SDVe is presented in the file along with a count of instances when sensor readings violate such property. The file also contains, a count of the total number of sensor reading evaluated, a count of the number of violations found, and a file detection rate calculated from the ratio between total violations found and the total number of checks. The verification summary file also provides total an aggregation of the total time in milliseconds to load to load the text file to be evaluated and the total time in milliseconds to verify the file.
The verification file includes the evaluated data properties along with the instances of the data that violate the data property. Figure 17 presents a snapshot of a verification file. Values indexed by the timestamp are violations of the property “Given a(n) Datum category, For all e_hmp dataset values, it is always the case that e_hmp:kPa > 0.463:kPa.” where e_hmp represents the vapor pressure measurements in kPa collected by a temperature and relative humidity probe.
Figure 17. Verification file snapshot.
Chapter 6: Evaluation

6.1 Overview

In order to demonstrate that scientist-specified properties are effective at identifying anomalies in sensor data, a series of case studies and experiments was conducted. An initial case study was conducted to determine how effective scientist-specified properties are at identifying anomalies that use historical and trend data on hyperspectral sensor data. A separate case study and two experiments were designed and conducted to determine how effective scientist-specified properties are at identifying anomalies related to experimental conditions in ecological data collected from an Eddy covariance tower and how effective scientist-specified properties are at identifying instrumentation anomalies in ecological data collected from an Eddy covariance tower. The results of the case studies and experiments are described in this chapter.

6.2 Hyperspectral Data

6.2.1 Robotic Tram Systems

Some environmental scientists conduct ground-based hyperspectral remote sensing studies to monitor surface properties when satellites cannot (e.g., when cloud cover prevails) and to formulate mathematical relationships between surface optical properties and environmental phenomenon. Robotic tram systems have been shown to be useful in conducting such ground-based research [50]. The robotic tram system used for this work consist of a tramline that is placed over an area of interest upon which a robotic cart is programmed to collect hyperspectral data using sensors that are activated at selected points along the tramline [51]. Hyperspectral data is obtained using a dual-detector field portable spectrometer [52]. The spectrometer has a nominal range of operation between 303 nanometers (nm) and 1148 nm in 256 contiguous bands with a spectral resolution of approximately 3 nm and a full-width-half maximum of approximately 10 nm. The optimal range of this detector (range with reasonable signal-to-noise) is approximately 400-1000 nm. Radiance (radiation from the target) and irradiance (radiation from the sky) are collected simultaneously, thereby permitting correction of surface reflectance under varying sky conditions [50]. The two detectors are cross-calibrated using a white panel
with 99% reflectance [53] at the beginning and at the end of a tramline. A typical reflectance graph as generated by the spectrometer is depicted in Figure 18.

![Reflectance graph](image)

**Figure 18.** Typical reflectance graph as used by environmental scientists.

Most robotic tramlines operate semi-autonomously and require careful attention to sensor calibration. Environmental conditions such as rain or fog, sun angle, and extreme cold and wind can affect the quality of data. At predefined increments of time or after the robotic cart has travelled a particular distance, the operator generally assesses the quality of the data by physically accessing the cart and visually inspecting the spectrometer that plots the data as they are collected. The operators use their experience-based knowledge and intuition to determine whether the collection process is operating as expected. If an anomaly or an unexpected value(s) in the data is suspected, the operator either restarts the measurement process, or persists with data collection and corrects data during post-processing.

### 6.2.2 Data Assessment Run-Time (DART) Monitoring Framework

The Data Assessment Run-Time (DART) monitoring framework was developed to address the data assessment challenges associated with the anomaly detection in hyperspectral data collected using robotic tram systems and to further understand the type of data properties associated with the collection process. DART is intended to detect deviations from an “ideal” set of data for a particular season (referred to as the *representative data set* hereafter) that is derived from expert knowledge and historical
data. The deviations identify points of interest in the data due to environmental variability or instrument malfunctioning.

DART allows environmental scientists using the tram system to specify and then verify data properties as data are streamed wirelessly from the tram system. DART works similarly to a software engineering run-time monitoring system with minor adaptations to accommodate for data processing. For the DART framework, a data property is a logical statement about data values associated with hyperspectral sensor readings. With DART, a user specifies a set of data properties of interest to capture the expected data values. DART is intended to detect deviations from an “ideal” set of data that is derived from expert knowledge and historical data, i.e., the representative data set.

DART can operate in collection time mode and post-collection mode. Collection time mode allows scientists to verify the data at collection time as the data are wirelessly streamed from the tram cart to a computer. In collection time mode, DART is given a path to a run-time folder, and DART continually checks data stored as new files in the folder, i.e., files sent by a spectrometer. Post-collection mode is used to verify data that has already been collected and stored.

The system takes three files as input: 1) a “Run-time Data File” (obtained from a run-time folder) that contains the raw data and metadata collected by the spectrometer on the tram cart at a given interval in time or position along the tramline; 2) an expert-validated “Representative Data Set File” (stored in the “Representative Data Set Repository”) that contains averaged historical data from the previous season; and 3) an expert-validated “Property Specification File” (stored in the “Specification Repository”) that contains the properties to be verified. The “Representative Data Set Repository” stores seasonal hyperspectral reflectance files, where a representative data set file contains averages of hyperspectral reflectance data considered representative of the data for a particular time of the season. A representative data set is used to compare its values to the data being collected by the tram system. In particular, the property specifications specify threshold values for the difference between the actual readings and the values of the representative data set. Specifications are described in XML files and are divided accordingly to the quality thresholds expected for each week of the season of interest.
The DART framework generates two outputs for every processed “Run-time Data File”, an “Assessment Log File” and an “Assessment Visual Representation.” The “Assessment Log File” contains the metadata associated with the spectrometer used to collect the data, a local data assessment value for each range of interest in the data, and a global data assessment value derived from local data assessment values from individual ranges. The “Assessment Visual Summary” is a graphical representation of the assessment results associated with the collected spectral readings.

A prototype version of DART was developed. The system is composed of six modules: the file parsers, the metadata handler, the run-time data reflectance calculator, the monitor, the specification mapper, and the output generator. The dataflow diagram in Figure 19 depicts the system.

The file parsers module extracts the run-time metadata and raw sensor data from the “Run-time Data File,” the sensor data from the “Representative Data Set File,” and property specifications from the “Property Specifications File.” The metadata handler module uses the run-time field data metadata extracted by the file parsers to create a metadata summary to be associated to the corresponding “Data Quality Log File”. The run-time data reflectance calculator module calculates a reflectance value for each of the 256 spectral wavelength bands recorded by the spectrometer in the “Run-time Data File”. The module calculates the reflectance value by dividing the radiances value by the irradiance value and associating the resulting reflectance value to the corresponding wavelength.

The specification mapper module maps the data property specifications extracted from the “Property Specification File” into a specification code template that is applied by the monitor module to the calculated reflectance data. The property specifications define the upper and lower wavelength limits to the intervals of interest in the data spectrum, and the maximum threshold value, which are determined from scientists’ expertise and data values expectations for which the normalized sensor data can differ from the “ideal” seasonal data.
The **monitor** module calculates the deviation of the normalized sensor data from the representative seasonal sensor data and assigns a data assessment value depending on how distant the deviation is from the specified threshold. The data assessment value for individual readings are used to determine the local data assessment for the predefined ranges in the spectra, and the local data assessment values are used to determine the global data assessment value for the “Run-time Data File” being processed.

The **output** generator module uses the results of the **monitor** module and the **metadata handler** module to create a quality log file for every processed “Run-time Data File” and creates a plot of the reflectance values calculated by the **run-time data reflectance calculator** module. Both artifacts are used to present the data assessment results in DART’s graphical user interface.

The scientists use DART’s graphical user interface to specify the locations of the various file folders, as well as to determine the type of data assessment to be performed, e.g., collection time or post-processing. Once the “Run-time Data File” is processed, the interface shows the data assessment flags associated with each wavelength value, the metadata summary, the data assessment summary for the file, and the spectra of the data.
DART performs the data assessment by assigning data assessment flags to individual reflectance values, ranges of interests, and “Run-time Data Files.” For individual reflectance values, DART determines the deviation of the reflectance value from the “ideal” seasonal data values. Given a set of predefined thresholds provided by the user, DART compares the reflectance deviations to such thresholds and classifies the severity of the deviation. For every wavelength value in a specific range, the absolute difference (AD) between the derived reflectance value and the representative seasonal value is calculated and compared to a predefined threshold (T); if AD is less than or equal to T, the derived reflectance value is not considered a deviation. If the AD deviates from T, the assessment flag changes accordingly (Table 19).

**Table 19.** Tolerance ranges used in DART to assess hyperspectral data.

<table>
<thead>
<tr>
<th>Flag</th>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00D</td>
<td>AD≤T</td>
<td>No deviations found.</td>
</tr>
<tr>
<td>01D</td>
<td>T&lt;AD≤T+0.1</td>
<td>Small deviation found.</td>
</tr>
<tr>
<td>02D</td>
<td>T+0.1&lt;AD≤T+0.2</td>
<td>Medium deviation found.</td>
</tr>
<tr>
<td>03D</td>
<td>AD&gt;0.2</td>
<td>Large deviation found.</td>
</tr>
</tbody>
</table>

For ranges of interest, a count of occurrences of the four types of data assessment flags in individual reflectance values is maintained. The range assessment flag is calculated by identifying the data flag with the largest count within the range. Similarly, the “Run-time Data File” data flag is that of the range assessment flag with the largest count. The graphical user interface is presented in Illustration 4.
6.2.3 Case Study

6.2.3.1. Description

A case study was conducted in collaboration with the System Ecology Laboratory at The University of Texas at El Paso and the University of Alberta to determine how effective are scientist-specified properties at identifying anomalies that use historical and trend data on hyperspectral sensor data collected by tram systems located on the Barrow Environmental Observatory near Barrow, Alaska. In particular, DART was used to interface with and identify anomalies in hyperspectral data files from a robotic tram system established as a component of the Barrow Biocomplexity flooding and draining experiment [51]. The tramline infrastructure for this site was a robotic tram system similar to that described by Gamon and others [50]. The three 300 meter-long tramlines were located in treatments in a large-scale hydrological manipulation experiment [51].

The tram cart setup was modified to include a laptop on which DART was executed. The laptop was remotely accessed through a wireless connection. The new setup was used to assess the data at collection time for three runs on the same date to show the feasibility of the approach; however, due to
limited wireless connectivity in the field, the remainder of the 2008 seasonal data was assessed using DART’s post-processing mode.

Representative data sets for the 2008 season were created using corrected historical data gathered in 2007. A series of representative data sets, which initially were selected based on the time of the season, were used to compare the representative values to the data being collected by the tram system. The representative data sets included average weekly reflectance data constructed from 2007 post-processed data for the three tramlines. This resulted in 27 representative data sets, nine per tramline, and one for every week of the 2007 season.

In collaboration with an environmental scientist, 13 specification files were created containing expected threshold values for different times of the season and delimiters to define ranges of scientific interest in the spectra. The specification files contained properties defined for three ranges of scientific interest within the optimal range in the spectra (i.e., 400-1000 nm) and two noise ranges outside the optimal range, and the initial thresholds used to evaluate the raw data against the “ideal” data. The thresholds and the delimiter selections were based on the scientist’s knowledge acquired over time.

6.2.3.2 Results

A total of 81 days of data were processed using DART, for a total of 24,690 spectral files, and 7,407,000 spectral readings for the 2008 season. Six additional days were unprocessed because the measurements were taken with a different spectrometer than the one used to create the seasonally representative data sets; thus, even though the specifications could be reused for these measurements, representative data sets for the new spectrometer were unavailable.

Figures 20-22 depict the results of using DART for data quality assessment. The x-axis presents the day of year when measurements were taken, and the y-axis presents the occurrences of each type of data quality assessment flags. Table 20 presents the 2008 data assessment generated by DART.

The hyperspectral data at the beginning of the season was expected to contain large deviations due to prevalent snow and melting snow. DART correctly identified the expected “largely deviated” (03D) data during this period, a transition period from June 18 (Day 1.) to June 30 (Day 181), and the
expected “no deviations” (00D) for the rest of the season for the North and Central tramlines. Technical problems appear to have influenced data obtained from the South tramline data throughout the season.

Figure 20. DART data assessment results distribution for North tramline for the 2008 season.

Figure 21. DART data assessment results distribution for Central tramline for the 2008 season.

Figure 22. DART data assessment results distribution for South tramline for the 2008 season.
On July 7, 2008 (Day 188), The North and Central tramlines were flagged as “no deviations” (298 files flagged as 00D) in each tramline, while the South tramline data were flagged as “large deviations” (03D). Metadata showed that the calibration equipment used by scientists in the South tramline to calibrate the spectrometer was dysfunctional and, thus, the integration time had to be changed to 25 milliseconds, which might have influenced the sensor reading.

Table 20. Tram data assessment results for the 2008 season.

<table>
<thead>
<tr>
<th>Tramline</th>
<th>00D</th>
<th>01D</th>
<th>02D</th>
<th>03D</th>
</tr>
</thead>
<tbody>
<tr>
<td>North</td>
<td>5422</td>
<td>20</td>
<td>32</td>
<td>2742</td>
</tr>
<tr>
<td>Central</td>
<td>5204</td>
<td>121</td>
<td>84</td>
<td>2808</td>
</tr>
<tr>
<td>South</td>
<td>6412</td>
<td>86</td>
<td>171</td>
<td>1588</td>
</tr>
</tbody>
</table>

On July 25, 2008 (Day 206), while data was being collected from the Central tramline, a heavy rain event occurred. It was captured by DART, which reported 291 00D files, two 01D files, five 02D, and two 03D files. The North and South tramlines reported 300 00D files, indicating an event that was unique to the Central tramline. Snowfall also occurred in August 1, 2008 (Day 213). The tramlines captured the event by showing a mixture of data quality assessment measurements. The North tramline reported 262 00D files, 14 01D files, 12 02D files, and 23 03D files. The Central tramline reported 91 00D files, 75 01D files, 39 02D files, and 105 03D files. The South tramline reported 35 00D files, 36 01D files, 14 02D files, and 225 03D files. Even though DART solely identified the snow event as a possible error, cross checking the results with climate data identified this environmental phenomenon.

In addition to the numeric findings, the case study identified two types of properties of interest to scientists: data properties and instrument properties. Data properties specify expected values and relationships related to field data readings i.e., noise ranges, and data values outside the specified thresholds for spectral ranges. Instrument properties specify expected instrument behavior and relationships by defining examining attributes and instrument functions based on reading (e.g., low voltage, bad fiber optic, and loose connections). The case study also showed that software engineering run-time verification techniques can be adapted to be used as data assessment techniques.
6.3 **Eddy Covariance Data**

6.3.1 **Eddy Covariance Towers**

Scientists working at the Jornada Basin Experimental Range (JER), which hosts the Jornada Basin Long-term Ecological Research program located in the northern Chihuahuan desert (+32.5 N, -106.8 W, elevation 1188 m) are interested in understanding how changes in ecosystems, land use, and climate change alter land atmosphere exchange of carbon, water and energy. Eddy covariance (EC) towers [54] are useful for collecting such measurements. Eddy covariance (EC) methods measure exchange of carbon dioxide (CO$_2$), water vapor, energy between land and atmosphere, and turbulent fluxes within the atmospheric boundary layer [54]. In particular, scientists use measurements collected by EC towers to identify what are the mechanisms by which ecosystems exchange carbon dioxide (CO$_2$) with the atmosphere. These measurements involve multiple sensors that record measurements up to approximately 10 times per second. The major goals of scientists studying the area are to estimate annual carbon budgets of ecosystems to improve and to understand the responses to present and future climate and to feed models to validate regional and continental carbon budgets [54].

For this dissertation, sensor data sets were obtained from an Eddy covariance station located in the JER site. The station consists of an extended open path Eddy Covariance system, mounted on a 10-meter high tower. The station contains more than 20 sensors, a data logger, a multiplexer, and a laptop computer. The station measures 14 parameters up to approximately 10 times per second and aggregates 107 variables in half hour intervals in addition to the collected raw data. The station also hosts a local wireless network that provides remote access to the laptop computer and loggers [54]. Every 24 hours the data is remotely accessed, collected and stored by scientists on servers. Table 21 describes the sensors from the Eddy covariance tower used for the experiment.
Table 21. Sensors included in the Jornada Eddy covariance tower used for the experiment.

<table>
<thead>
<tr>
<th>Instrument</th>
<th>Parameter</th>
<th>Description</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Campbell Scientific Data Logger (CR3000)</td>
<td>TIMESTAMP</td>
<td>Date and Time at which the measurement was taken</td>
<td>n/a</td>
</tr>
<tr>
<td>Campbell Scientific Data Logger (CR3000)</td>
<td>RECORD</td>
<td>Consecutive sensor reading indexing number</td>
<td>n/a</td>
</tr>
<tr>
<td>Campbell Scientific 3-D Sonic Anemometer (CSAT3)</td>
<td>Ux</td>
<td>Horizontal wind (x-axis)</td>
<td>m/s</td>
</tr>
<tr>
<td>Campbell Scientific 3-D Sonic Anemometer (CSAT3)</td>
<td>Uy</td>
<td>Horizontal wind (y-axis)</td>
<td>m/s</td>
</tr>
<tr>
<td>Campbell Scientific 3-D Sonic Anemometer (CSAT3)</td>
<td>Uz</td>
<td>Vertical wind</td>
<td>m/s</td>
</tr>
<tr>
<td>Campbell Scientific 3-D Sonic Anemometer (CSAT3)</td>
<td>Ts</td>
<td>Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>LI-COR Biosciences Open Path CO$_2$/H$_2$O Gas Analyzer (LI-7500A)</td>
<td>CO$_2$</td>
<td>Carbon Dioxide Mass Density</td>
<td>mg/m$^3$</td>
</tr>
<tr>
<td>LI-COR Biosciences Open Path CO$_2$/H$_2$O Gas Analyzer (LI-7500A)</td>
<td>H$_2$O</td>
<td>Water Vapor Mass Density</td>
<td>g/m$^3$</td>
</tr>
<tr>
<td>Campbell Scientific 3-D Sonic Anemometer (CSAT3)</td>
<td>fw</td>
<td>Thermocouple Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>LI-COR Biosciences Open Path CO$_2$/H$_2$O Gas Analyzer (LI-7500A)</td>
<td>press</td>
<td>Pressure</td>
<td>kPa</td>
</tr>
<tr>
<td>Campbell Scientific 3-D Sonic Anemometer (CSAT3)</td>
<td>diag_csat</td>
<td>Diagnostic Word</td>
<td>unitless</td>
</tr>
<tr>
<td>Campbell Scientific 3-D Sonic Anemometer (CSAT3)</td>
<td>agc</td>
<td>Automatic Gain Control</td>
<td>unitless</td>
</tr>
<tr>
<td>Campbell Scientific Temperature and Relative Humidity Probe (HMP45C-L)</td>
<td>t_hmp</td>
<td>Temperature</td>
<td>°C</td>
</tr>
<tr>
<td>Campbell Scientific Temperature and Relative Humidity Probe (HMP45C-L)</td>
<td>e_hmp</td>
<td>Vapor Pressure</td>
<td>kPa</td>
</tr>
<tr>
<td>Campbell Scientific Barometric Pressure Sensor (CS106)</td>
<td>atm_press</td>
<td>Atmospheric Pressure</td>
<td>kPa</td>
</tr>
</tbody>
</table>

6.3.3 Experiments

6.3.3.1 Background

Upon further analysis of the data properties obtained from the literature survey, it was determined that three data property types account for approximately 72.5% of the total number of data properties specified by scientists in the literature review; datum properties (32.5%), datum relationship properties (30.8%), and datum dependent instruments (9.2%). Based on these findings, a series of experiments were designed and conducted to measure the effectiveness of anomaly detection in sensor data through data properties. In particular, two were conducted to determine if DaProS–generated data
properties can be used by the SDVe tool to detect anomalies in Eddy Covariance sensor data for data properties of type datum, datum relationship, and datum dependent instrument.

Eddy covariance data was of interest to this study because quality control of Eddy covariance data requires instrument errors to be identified and the evaluation of how closely environmental conditions fulfill the theoretical assumptions underlying the method. These must be done automatically at collection time, or shortly after the measurements are sensed, to minimize data loss by reducing the time to detect and fix instrument problems. Additionally, screening of data must identify non-random periods of unsuitable data in the dataset. The quality control procedures, environmental conditions that fall outside of the assumptions of the methods, and instrument malfunctions, maintenance and calibration periods often remove 20 to 40% of the data [49].

Data quality assurance and quality control are outstanding problems that are not completely fulfilled in most of the Eddy Covariance networks such as FLUXNET [55], because most networks and stations using this method do not have a uniform scheme for quality control of Eddy covariance measurements. To address such a challenge, scientists need tool support to be able to capture, reuse and share their expertise to enable automatic detection of anomalies in scientific sensor data. Specifically, to detect anomalies in scientific sensor data, the data evaluation tool must be able to find data errors and uncertainty, both random and systematic, allow wrong or suspicious data to be found directly without accomplishing data analysis or without using a complete statistical analysis of time series, and check the data automatically without manual evaluation [6].

For the experiment, a set of data properties of interest was developed in collaboration with expert scientists from the University of Texas at El Paso Systems Ecology Lab working with Eddy covariance and Biomesonet towers’ data. The scientists contributing to this work were building their first tower and this study utilized amongst the first data to be collected by this system. As such, the scientists were interested in using a specification tool to capture data properties to flag problematic data based on expert knowledge captured by literature and other data collection systems in comparable landscapes. The data properties captured expert knowledge obtained from sensors reference manuals [56] [57], climate and
climatological variations in the Jornada Basin literature [58], Eddy covariance towers post-field data quality control literature [49], and scientists’ field experience.

6.3.3.2 Data Properties Seeding Experiment

The purpose of the experiment was to check the ability of the SDVe tool to detect anomalies. An error-free data file was randomly selected from the month of February and seeded with anomalies to evaluate the data properties. The experiment aimed to determine whether all seeded anomalies were found and that all events marked as anomalies were actually anomalies.

The data file was exported to Microsoft Excel to allow data manipulation and anomalies to be seeded independently by a scientist. A sample size calculator obtained from Creative Research Systems [59] was used to determine the number of anomalies to be seeded for each property in order to obtain statistical significance. The sample size results were reassured with a separate sample size calculator obtained from Raosoft [60]. To achieve a confidence level of 95%, with a confidence interval of five, for a sample population of 36000 measurements, the sample size was determined to be 380 points.

The experiment was constructed from a use case scenario built in collaboration with environmental scientists. The use case scenario deals with the atmospheric boundary layer, where Eddy covariance measurements take place, as it changes continuously in response to the heating and cooling of the Earth’s surface. Carbon dioxide, atmospheric pressure and temperature were evaluated through daily cycles drive. The scenario studies the changes on the carbon dioxide concentrations (CO₂) obtained from a LI-COR Biosciences Open Path CO₂/H₂O Gas Analyzer (LI-7500A), the relationship between pressures obtained from two pressure sensors from the Eddy covariance tower, here after also referred as press and atm_press, to differentiate between pressure reading taken from the LI-COR Biosciences Open Path CO₂/H₂O Gas Analyzer (LI-7500A), and the Campbell Scientific Barometric Pressure Sensor (CS106), respectively. The variable Press_diff is introduced to represent the dataset containing the absolute difference between press and atm_press. Even though Press_diff is evaluated as a datum property, conceptually the results of evaluating properties over Press_diff are considered as datum relationship. The adaptation had to be made because SDVe does not provide evaluation support for computational methods, e.g., Press_diff cannot be calculated internally. A separate module calculated Preff_diff values
externally as the data was read from the text files and processed by SDVe. Four data properties were used for this experiment. The data properties were related to the use case scenario and included two datum properties, a datum relationship property, and a datum dependent instrument property. The data properties descriptions, number of seeded anomalies and number of seeded anomalies identified by SDVe are shown in Table 22.

**Table 22.** Data properties used in the case study, with property types and descriptions.

<table>
<thead>
<tr>
<th>#</th>
<th>Property</th>
<th>Description</th>
<th>Seeded Anomalies</th>
<th>Anomalies Found</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Always, 626.5791&lt;CO₂&lt;632.2966 Datum, Global, <strong>Universality</strong>(CO₂: mg/m³, &lt;, 632.2966: mg/m³)</td>
<td>The carbon dioxide readings for this part of the season should be within the specified constant value.</td>
<td>380</td>
<td>380</td>
</tr>
<tr>
<td></td>
<td>Datum, Global, <strong>Universality</strong>(CO₂: mg/m³, &gt;, 626.5791: mg/m³)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Always, latm_press-press=0. Datum, Global, <strong>Universality</strong>(Press_diff: kPa, =, 0:kPa)</td>
<td>There should not be a difference between redundant pressure sensors located at different heights in the tower. <em>Note:</em> Even though the property is evaluated as a Datum property due to SDVe processing restriction, the results are classified as Data Relationship.</td>
<td>380</td>
<td>380</td>
</tr>
<tr>
<td>3</td>
<td>Datum Dependent Instrument, Global, <strong>Precedence</strong>(Ts:°C ≥ 5.795044:°C, agc:n/a , &lt;, 70:n/a)</td>
<td>The agc value has been documented to be influenced by high temperatures, the property ensures that if the temperature in the Ts sensor is high, the agc value remains unaffected.</td>
<td>380</td>
<td>380</td>
</tr>
</tbody>
</table>

For each of the three properties, 380 samples were selected using the RAND function in Excel to generate a random number between 0 and 100. Results were automatically sorted and the lowest 380 values were selected without replacement. The procedure was followed three separate times for each of the properties so that samples were different for the different properties.

Because the file is expected to be anomaly-free, maximum and minimum values were calculated and used as numerical thresholds in the data properties to be evaluated; all of the CO₂ readings in the CO₂ column in the data file are greater than or equal to the minimum value and less than or equal to the maximum value. Thus, the introduced anomalies should be values smaller than the minimum value or greater than the maximum value. A scientist seeded the data file with a number of data elements from
the sample size selected. The new values for property 1 were calculated by generating a random number between 0 and 1. If the random number was greater than or equal to 0.5, a number higher than the maximum was generated, otherwise a number lower than the minimum was generated. Then, the difference between maximum and minimum values was calculated to be 5.72, and the current value CO$_2$ was replaced by the result of calculating \( \text{original value} \pm 5.72 \pm \text{a random number between 0.1 and 10} \).

The new values of atmospheric pressure for property 2 were calculated by generating a random number between 0 and 1. If the random number was greater than or equal to 0.5, a new value for \text{press} was generated; otherwise a new value for \text{atm_press} was generated. Then, a random number between 0 and 1 was generated. If the new random number was greater than or equal to 0.5 an addition to the original value was performed, otherwise a subtraction was performed. Finally, the new value for \text{press} or \text{atm_press} was calculated by \( \text{original value} \pm 2.063324 \pm \text{a random number between 0.1 and 10} \).

For property 3, the new values of Ts were calculated by adding 7.858368 and a random number between 0.1 and 10. Similarly, the new values for the agc flag were calculated by adding 70 and a random number between 0.1 and 10. Once the values have been seeded, the data relationships were manually checked to ensure that the 380 seeded relationships had a temperature value greater than 7.858368 and an agc value greater than 70.

Once the data file was seeded, the data file and the specification were used as input to SDVe for verification purposes. The results in Table 25 show that the SDVe tool successfully identified the seeded anomalies in the data file. The indexing values of the results generated by the SDVe tool were also compared to the indexing values of the seeded readings to ensure that both readings corresponded. Thus, all of the indexing values of the anomalies founds corresponded to an indexing value in the subset of seeded readings.

\subsection*{6.3.3.3 Data Properties for Collection Time Data Experiment}

The second experiment was conducted to determine the feasibility of using SDVe to identify anomalies in Eddy Covariance sensor data and to illustrate how such anomalies can be identified and
documented by cross referencing the results obtained from SDVe to existing metadata about the collection process. The experiment does not quantify the improvement in the overall quality of the data.

The scientists developed a matrix of all the relationships between sensors in the Eddy covariance tower of interest for the specific site. The matrix included raw sensor measurements and derived data aggregated at different temporal resolutions and as part of the combination of measurements from two or more sensors. The sensor relationship matrix consisted of approximately 118 sensor readings along with their associated relationships. In collaboration with scientists some of the sensor relationships from the matrix were used to develop a scenario to determine if SDVe can be used to detect anomalies in collection time raw sensor data.

In the scenario, the scientists specified properties of interest of type \textit{Datum, Instrument,} and \textit{Datum Relationship}. The properties were intended to capture anomalies in raw data at collection time. The sensor readings of interest were selected based on the relationship to other sensor readings and the derivation of aggregated values from them. In collaboration with scientists, data properties of interest were specified, refined, and validated using DaProS. The numeric values used in the data properties were defined following algorithms and protocols used by different scientific networks and groups (e.g., CarboEurope [61], Canada Flux [62], Ameriflux [55]). Table 23 shows the data properties used in the experiment to detect anomalies in EC sensor data. The data properties are described based on the output provided by DaProS and are intended to be semi-formal description of the properties. Table 21 provides the definitions and units of measurement of the sensors used in the data properties. For presentation purposes, the descriptions for the Boolean statements in Table 23 are presented in a more general form than the ones generated by the D-SPS.
Table 23. Data properties used in the experiment to verify Eddy Covariance data using SDVe.

<table>
<thead>
<tr>
<th>#</th>
<th>Data Property</th>
<th>Season</th>
</tr>
</thead>
<tbody>
<tr>
<td>1dp</td>
<td>Given (a)n Datum category: for all e_hmp dataset values, it is always the case that 1.1 kPa &lt; e_hmp &lt; 2.0 kPa</td>
<td>Summer</td>
</tr>
<tr>
<td>2dp</td>
<td>Given (a)n Datum category: for all e_hmp dataset values, it is always the case that 0.463 kPa &lt; e_hmp &lt; 0.476 kPa</td>
<td>Winter</td>
</tr>
<tr>
<td>3dp</td>
<td>Given (a)n Instrument category: for all e_hmp dataset values, it is always the case that e_hmp != 0 kPa</td>
<td>All</td>
</tr>
<tr>
<td>4dp</td>
<td>Given (a)n Datum category: for all atm_press dataset values, it is always the case that 85.89 kPa &lt; atm_press &lt; 85.94 kPa</td>
<td>Summer</td>
</tr>
<tr>
<td>5dp</td>
<td>Given (a)n Datum category: for all atm_press dataset values, it is always the case that 50.0 kPa &lt; atm_press &lt; 50.034 kPa</td>
<td>Winter</td>
</tr>
<tr>
<td>6dp</td>
<td>Given (a)n Datum Relationship category: for all atm_press dataset values, it is always the case that latm_press-press &lt; 2.0 kPa</td>
<td>All</td>
</tr>
<tr>
<td>7dp</td>
<td>Given (a)n Datum category: for all t_hmp dataset values, it is always the case that 25°C &lt; t_hmp &lt; 35°C</td>
<td>Summer</td>
</tr>
<tr>
<td>8dp</td>
<td>Given (a)n Datum category: for all t_hmp dataset values, it is always the case that 9°C &lt; t_hmp &lt; 10.2°C</td>
<td>Winter</td>
</tr>
<tr>
<td>9dp</td>
<td>Given (a)n Datum Relationship category: for all t_hmp dataset values, it is always the case that lt_hmp-templ &lt; 1.5°C</td>
<td>All</td>
</tr>
<tr>
<td>10dp</td>
<td>Given (a)n Datum category: for all H2O dataset values, it is always the case that 8.57 g/m³ &lt; H2O &lt; 9.04 g/m³</td>
<td>Summer</td>
</tr>
<tr>
<td>11dp</td>
<td>Given (a)n Datum category: for all H2O dataset values, it is always the case that 4.48 g/m³ &lt; H2O &lt; 4.63 g/m³</td>
<td>Winter</td>
</tr>
<tr>
<td>12dp</td>
<td>Given (a)n Datum category: for all UZ dataset values, it is always the case that 2.0 m/s &lt; Uz &lt; 2.0 m/s</td>
<td>Summer</td>
</tr>
<tr>
<td>13dp</td>
<td>Given (a)n Datum category: for all UZ dataset values, it is always the case that 1.3 m/s &lt; Uz &lt; 0.75 m/s</td>
<td>Winter</td>
</tr>
<tr>
<td>14dp</td>
<td>Given (a)n Datum category: for all CO₂ dataset values, it is always the case that 480 mg/m³ &lt; CO₂ &lt; 580 mg/m³</td>
<td>Summer</td>
</tr>
<tr>
<td>15dp</td>
<td>Given (a)n Datum category: for all CO₂ dataset values, it is always the case that 600 mg/m³ &lt; CO₂ &lt; 625 mg/m³</td>
<td>Winter</td>
</tr>
<tr>
<td>16dp</td>
<td>Given (a)n Instrument category: for all agc dataset values, it is always the case that agc &lt; 70 n/a</td>
<td>Both</td>
</tr>
<tr>
<td>17dp</td>
<td>Given (a)n Instrument category: for all diag_csat dataset values, it is never the case that diag_csat==NAN</td>
<td>Both</td>
</tr>
<tr>
<td>18dp</td>
<td>Given (a)n Instrument category: for all diag_csat dataset values, it is never the case that diag_csat==99999</td>
<td>Both</td>
</tr>
<tr>
<td>19dp</td>
<td>Given (a)n Datum Dependent Instrument category: for all Ts dataset values, it is always the case that, If Ts&gt;35°C, then agc&lt;70 mg/m³</td>
<td>Summer</td>
</tr>
</tbody>
</table>

The Eddy Covariance (EC) data verified using SDVe were collected from July 06, 2010 to July 13, 2010 to capture EC summer behavior and from February 09, 2010 to February 16, 2010 to capture EC winter behavior. The sensors at the tower collected the EC data continuously, and a scientist manually split the data into 1-hour interval files to ease the verification process. The detailed results of
evaluating the 349 data files are provided in Appendix D. The remainder of this subsection discusses the overall results.

SDVe performed a total of 219,800,854 evaluation calls of which 50,857,351 were anomalies, i.e., approximately 23% of the evaluation calls identified an anomaly. The evaluation process took approximately 20 hours to complete, of which approximately 1 hour was spent loading the files into the system and 19 hours were spent verifying the data. Assuming a data file takes 15 minutes on average to be manually processed and evaluated by a scientist manually processing the 349 files would take approximately 87 hours. SDVe automatically evaluates the data in one fourth of the time that it would take a scientist to manually evaluate the same amount of data.

Figures 23 and 24 compare for each season the total number of checks performed by SDVe to the total number of data properties violations found.

![Winter Season](image)

**Figure 23.** Ratio of total data checks to total anomalies founds for the winter season.
Figure 24. Ratio of total data checks to total anomalies found for the summer season.

Figure 25 and Figure 27 depict the distribution of anomalies detected by each data property category for the summer and winter season respectively. For winter data, 94,842,448 evaluations calls took place, of which 26,439,560 were anomalies, i.e., 28 percent of the evaluation calls identified anomalies. Datum properties identified the most anomalies (25,344,929), followed by Data Relationship (6,117,955), Instrument (162,410) and Data Dependent Instrument (0). The sensor datasets with the most anomalies included atmospheric pressure (atm_press), vapor pressure (e_hmp) and temperature (Ts) as described in Figure 27.
Figure 25. Ratio of total anomalies by data property category for the winter season.
Figure 26. Number of hourly data properties violations for the winter season.
For summer data, 124,958,406 evaluations calls took place, of which 24,417,791 were anomalies, i.e., 20 percent of the evaluation calls identified anomalies. *Datum* properties identified the most anomalies (21,429,802), followed by *Data Relationship* (2,639,985), *Instrument* (348,004) and *Data Dependent Instrument* (0). The sensor datasets with the most anomalies included water vapor mass density ($H_2O$), atmospheric pressure (atm_press), carbon dioxide ($CO_2$), and temperature (Ts) as described in Figure 28.

![Summer Season Total Anomalies Identified by Data Property Category](chart.png)

**Figure 27.** Ratio of total anomalies by data property category for the summer season.
Figure 28. Number of hourly data properties violations for the summer season.
Once the datasets were evaluated using SDVe, the anomalies found by the tool were cross-referenced with the available data-collection process metadata compiled by the scientists in the field site and with historical meteorological data obtain from the *Weather Underground* website [72]. SDVe was able to identify environmental variability and instrument malfunctioning in the datasets. Abrupt changes in the number of violations detected in the summer season by the data properties correspond to rain events that occurred in the area. Specifically, rain events occurred in July 8, 2010 (07/08/2010) and July 11, 2010 (07/11/2010) represented as high concentrations and abrupt changes in the “anomaly-found” trend lines at the time of event occurrence. A similar rain event, depicted in Figure 29, occurred on February 10, 2010 (02/10/2010) during the winter season. Such rain events were identified as both, true environmental variability and as a cause for instrument malfunctioning. Rain events affected data properties monitoring environmental variability such has water vapor mass density ($H_2O$), carbon dioxide mass density ($CO_2$), vapor pressure ($e_{hmp}$), and data properties monitoring instrument functioning such as the automatic gain control (agc) flag.
Figure 29. Rain event captured as abrupt changes in anomaly trend lines during the winter season.
6.3.4 Discussion

During the specification of data properties, many aspects need to be considered to maximize the amount of data anomalies to be identified. In some instances, the efficiency of the data anomaly detection mechanism depended on the quality of the specified properties. Such influencing aspects are described in the subsections below.

6.3.4.1 Conflicting Data Properties

Conflicting properties can be detected at the specification stage or at the verification stage. In this approach, the conflicting properties can be identified by the scientist at the verification stage once the data has been processed. Capturing conflicts at the specification stage is difficult because it requires a deep understanding of the data being analyzed and requires the scientist to keep track of the data properties and the relationship and side effects among them. Also, potential property conflicts are harder to validate because there is no immediate way to quantitatively compare the expected output for the data properties that can isolate the conflict. If the conflicts are captured at the verification stage, the scientist has access to the verification results and can cross-reference the results of the verification with the specified data properties, and decide if properties conflict with each other. For example, Figure 30 depicts two data properties that conflict with each other. An H₂O value might be greater than 9.04 and greater than 8.57, thus violating the former and satisfying the latter. Or an H₂O value might be less than 8.57 and less than 9.04, thus violating the former and satisfying the latter. Such property is represented in the graph as a reflection of the trend lines.
Threshold Selection

Another challenge is identifying the most appropriate threshold to be used in a data property. Threshold values, when needed, are usually site specific, thus it is difficult to identify universal threshold values that could use by all of the scientific communities. The correct selection of threshold values is very important for the evaluation of the sensor data. Consider for example, the datum property P1: “Given a Datum category, for all CO\textsubscript{2} dataset values, it is always the case that 600 mg/m\textsuperscript{3} < CO\textsubscript{2} < 625 mg/m\textsuperscript{3}” and the data property P2: “Given a Datum category, for all CO\textsubscript{2} dataset values, it is always the case that 500 mg/m\textsuperscript{3} < CO\textsubscript{2} < 700 mg/m\textsuperscript{3}.” P1 has a stricter criterion for identifying an anomaly. In addition, if the scientist is more interested in finding the CO\textsubscript{2} readings that highly deviate from the expected measurements, P2 would be a better property. Scientists must consider the intent of the property in addition to the site specific information when selecting the thresholds for the data properties. Determining the property that should be used is left to the discretion of the scientist, although the system should notify the scientist of properties that are subsumed by others.

Characterization of Environmental Variability and Instrument Malfunctioning

Scientists are interested in identifying the critical set of properties needed to identify environmental variability or instrument malfunctioning from the data. In this context, a critical set of properties is the group containing the minimum number of data properties that will allow a scientist to identify a weather or instrument feature in the data. Figure 31 illustrates a rain event that could be characterized by grouping together properties associated to water vapor mass density (H\textsubscript{2}O), carbon dioxide mass density
(CO₂), vapor pressure (e_hmp) and the automatic gain control (agc) flag that seem highly correlated during such events.

Figure 3.1. Characterization of a rain event as a graphical representation.
Data property clusters can also be used to document and alert that instrument maintenance is needed by capturing data properties associated with error flags from instruments. Consider the case of the diagnostic word diag_csat associated to the CSAT3 sonic anemometer. The diag_csat reports the error flags in Table 24. The error flags can be captured as data properties, also in Table 24. The error diagnostic flag [31] process can be characterized by the group of specified data properties and can be used as part of the fault tree.

**Table 24.** CSAT diagnostic flag definitions and data properties used to capture the flags.

<table>
<thead>
<tr>
<th>Error Flag</th>
<th>Description</th>
<th>Data Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>61502</td>
<td>NaN</td>
<td>Given a(n) Instrument category, for all dataset values, it is never the case that diag_csat==61502 holds</td>
</tr>
<tr>
<td>61440</td>
<td>Lost trigger</td>
<td>Given a(n) Instrument category, for all dataset values, it is never the case that diag_csat==61440 holds</td>
</tr>
<tr>
<td>61503</td>
<td>No data</td>
<td>Given a(n) Instrument category, for all dataset values, it is never the case that diag_csat==61503 holds</td>
</tr>
<tr>
<td>61441</td>
<td>SDM error</td>
<td>Given a(n) Instrument category, for all dataset values, it is never the case that diag_csat==61441 holds</td>
</tr>
<tr>
<td>61442</td>
<td>Wrong CSAT3</td>
<td>Given a(n) Instrument category, for all dataset values, it is never the case that diag_csat==61442 holds</td>
</tr>
</tbody>
</table>

6.3.4.6 Confidence Levels Capture

Data properties can also be used to assign confidence levels of quality to collected data. Data properties can be defined in terms of the diagnostic measurements in the data collection instruments that are affected by the conditions surrounding the experiment. Consider the clean window value (agc) from the Li7500 instrument. Typical clean windows values are between 55% and 65%. As dirt or water accumulates on the windows, or anywhere in the optical path, the agc value increases. Changes in the agc value can result from dust, pollen, vibrations, dew, and rain/snow [48].

Given that the higher the value of the agc, the higher the probability that the collected data results in bad/ not-expected data, the scientist can specify data properties for which confidence levels can be assigned. For instance, Table 25 presents a series of data properties that can be used to assign confidence levels to data collected at different readings given by the agc.
Table 25. Agc confidence levels for specified data properties.

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Confidence Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>For all status readings in the Li7500 sensor, agc value&lt;=40%</td>
<td>Good Data</td>
</tr>
<tr>
<td>For all status readings in the Li7500 sensor, 40%&lt;agc value&lt;=50%</td>
<td>Probably Good Data</td>
</tr>
<tr>
<td>For all status readings in the Li7500 sensor, 50%&lt;agc value&lt;=60%</td>
<td>Probably Bad Data</td>
</tr>
<tr>
<td>For all status readings in the Li7500 sensor, agc value&gt;60%</td>
<td>Bad Data</td>
</tr>
</tbody>
</table>

The confidence levels assigned to the data are subjective and dependent on the scientists’ mental model. The goal of the confidence levels is to identify those measurements that require the most analysis and attention at the post-processing stage to determine if an anomaly occurred due to environment variability or to an instrument malfunction.

6.3.4.7 Data Granularity

Data properties can be used to identify anomalies at different data granularities, i.e., the level of fineness to which a dataset is sub-divided. The challenge is to determine at which data granularity to verify the data such that the number of anomalies found is maximized. If the data is verified at the highest level, i.e., the smallest sub-division of a dataset, every data point can be examined and verified; however, it requires more computational power to do so. Lowering the data granularity relieves the computation power needed to verify the data, but can miss anomalies in the data. For this work, the data granularity used was the highest available--data measurements were taken every millisecond and stored in half hour files.

6.3.4.8 Data Property Precision

The results of the data anomaly detection process are dependent on how specific and precise the data properties are. Due to differences in data types and data behavior according to seasonal and diurnal variability, in some cases it is necessary to define the data properties to be as specific as possible to an expected data behavior or time-related variability. Consider the EC tower used in the experiment. The expected data values measured by the sensors differ depending on the season, the diurnal cycle and the
time of the day. The effectiveness of the data anomaly detection process depends not only on the data granularity and the data property quantitative values, but also on the precision with which the seasonal and diurnal cycles are modeled by the data properties. Different data property sets, with specific data granularity and quantitative values, have to be built for specific parts of the season and diurnal cycles. Figure 32 depicts how the effectiveness of the data properties for many of the sensor readings in the EC tower vary depending on a cyclical behavior correlated to the diurnal cycles in the winter season. In Figure 32, the drop in temperature at nighttime generating a diurnal pattern of anomalies indicates the need for a separate data property to be specified for temperatures at nighttime.

![Summer Temperature Fluctuation](image)

**Figure 32.** Cyclic diurnal behavior captured by specified data properties.

### 6.3.4.9 Data Property Specification Types

The experiment identified two purposes for which data properties can be specified; data properties can be specified to document scientific knowledge about processes or to identify anomalies in scientific sensor data. Data properties specified to detect anomalies in sensor data are typically specific about the sensor names and thresholds over which the data should be evaluated, can be interpreted and used to evaluate data by data anomaly detection mechanisms without further manipulation to the property, and can be used to document the scientific processes. Data properties that are specified for the sole purpose of documenting processes are typically general in their descriptions, e.g., only describe which sensor reading will be evaluated and how it will be used, but do not include the specific name or threshold values to be evaluated. Data properties for documenting processes might also include computational methods that need to be applied to data before the data can be evaluated. These types of data properties are not suitable for the current version of SDVe.
6.3.4.10 Data Files Parsing

The current implementation of SDVe has limited scalability because it requires a new data file parser to be created every time a new type of sensor data will be evaluated. However, this limitation can be addressed by standardizing the format of the sensor reading files generated by the different scientific communities. If the sensor reading files are standardized, the number of parsers needed to be implemented for the SDVe tool will be reduced and will allow SDVe to be usable by broader communities. In addition, it will foster the reuse of data properties by scientists.
Chapter 7: Conclusion

7.1 Summary of Work

The amount of sensor technology that is being introduced at remote research sites to collect environmental data is rapidly increasing, and the associated instrumentation typically does not include mechanisms to identify anomalies during data collection. It would be beneficial if scientists could have assurance that the collected data sets are of high quality. The development of mechanisms and procedures to verify the integrity of the data motivates the research. To address this, the research defined an environmental-scientist-centered approach for specifying data properties that defines temporal and data relationships associated with sensor data to identify sensor-data anomalies. The approach, which is based on software-engineering techniques, supports the scientists’ ability to specify data properties through guidance using property classifications. The scientist-specified properties can be used to detect anomalies in sensor data and capture temporal and data relationships regarding experimental conditions and instrumentation.

A literature review found 15 projects that collect environmental data from sensors, and documented the type of analyzed data and the type of data properties as defined by each project. More than 500 published data properties were extracted from the surveyed projects. A data property categorization was created. A case study conducted near Barrow Alaska also documented the types of data properties of interest to environmental scientists and how these properties can be specified.

An outcome of the research is a specification-pattern-system approach called Data Specification and Pattern System (D-SPS) and the Data Property Specification (DaProS) tool that assists scientists in specifying sensor data properties. DaProS is based on the aforementioned data property categorization and D-SPS. The DaProS tool uses decision trees to guide scientists through the data property specification process, and it provides a disciplined natural language description of specified properties to allow scientists to validate the intended meaning of the specification. The generated properties from DaProS can be exported to data verification tools such as the Sensor Data Verification tool.

Case studies and experiments were conducted to evaluate the use of data property specifications and tools to detect environmental variability and instrument malfunctions during the collection of
scientific sensor data. In collaboration with scientists working with Eddy covariance data, a set of data properties of interest was assembled, specified, refined, and validated using the DaProS tool. Resulting DaProS specifications and a seeded data file were used as input to the Sensor Data Verification (SDVe) tool to determine the tool’s ability to identify anomalies on sensor data obtained from an Eddy covariance tower from the Jornada Basin Experimental Range (JER).

SDVe was successfully used to identify and distinguish anomalies caused by environmental events from those caused by instrument malfunction. The experiment’s results also identified many factors that play a role in the effectiveness of the data anomalies detection process. In addition, the work on this dissertation is applicable to any application that requires anomaly detection for large amounts of sensor data and variable properties and has the potential to be used as an educational tool to assist students as they develop skills to interpret data and to identify and specify data properties.

7.2 Future Work

The work provides an approach to specify data properties based on a categorization comprised of 10 data properties types. However, the experiment included only the three most frequently used data property types: Datum properties, Datum Relationship properties, and Datum Dependent Instruments properties. Future work includes extending the experiment to include the remaining data property categories. Also, the SDVe tool at this point only supports patterns associated with the Datum properties, Datum Relationship properties, and Datum Dependent Instruments properties. Work is still needed to extend the SDVe functionality to support the remaining patterns. Also, further SDVe support for other types of data, such as spectral data obtained from the tram system, is desired.

D-SPS can be improved through the inclusion and extension of Mondragon’s composite propositions [25]. In particular, composite propositions can help reduce ambiguity when specifying data properties that capture concurrent reading and instrument behavior. Another approach that can benefit the D-SPS is the use of fault tree analysis to combine properties at a higher level of abstraction than how they are being specified at this point. By defining and grouping the data properties at a higher abstraction level, fault trees can provide scientists with guidance and define best practices for considering errors in sensor data and potential causes of the errors.
The DaProS and SDVe tools will be extended to be remotely accessible by wireless and cellular devices. In particular, SDVe will be improved to increase the data processing capabilities of the tool. To speed up the data anomaly detection process, SDVe will use emerging large dataset distributed processing technologies such as Hadoop [67] to divide and distribute the data to different clusters of computers, which would process smaller amounts of data at every cluster, and which data anomaly detection results can be aggregated to a central application.

The approach used for this dissertation work assumes that the data are available in the form of a text file generated shortly after a sensor acquires the data. The reason for this assumption is that most instrumentation has proprietary software that does not allow access to the data until it is gathered and logged, at which point the proprietary software allows the generation of the text file. Recent deployment of open-source data collection and logging systems powered by middleware that support real-time streaming of data such as Data Turbine [68] would allow access to real-time data streams. SDVe can also be improved to allow the identification of anomalies in streams of scientific data at near real-time. Due to the nature of real-time data, new research opportunities will be explored to determine how to evaluate data properties over streaming data that may be streamed in parallel, sequence, or in recurrent periods of time.

Finally, future work for DaProS tool and the SDVe tool includes bundling them with sample data files, sample data property specifications, and best practices manuals for use as a pedagogical solution to help train new scientists into understanding and identifying errors in sensor data.
References


115


[64] Canada Federal Department of Fisheries and Oceans. “Data Quality Assurance (QC) at the Marine Environmental Data Service (MEDS).” DFO ISDM Quality Control. Internet: http://www.meds-sdmm.dfo-mpo.gc.ca/meds/Prog_Int/WOCE/WOCE_UOT/qcproces_e.htm [January 21, 2009].


Appendix A: Active Database Systems Operator Summary

Operators to specify data behaviors have been introduced by every major aDBS. This section provides a summary of the introduced operators, along with their semantics. The operator column depicts the different operator representations as used by the different formalisms. The name column specifies the name associated for the given operator. The semantics column presents the semantic associated to each operator as described in the literature. Finally, the used by column specifies the languages that use the specified operator with the semantics described. Even though some operators might look similar, there still some subtleties that differentiate them, thus the large amount of operators presented in this section. Operators that were not strictly equivalent among formalisms were placed separately on its own category.

Table A1. Active database systems operators summary.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Name</th>
<th>Semantics</th>
<th>Used by</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V, \parallel$</td>
<td>OR operator</td>
<td>Disjunction of two events $E_1$ and $E_2$, occurs when $E_1$ occurs or $E_2$ occurs.</td>
<td>SNOOP ODE SAMOS</td>
</tr>
<tr>
<td>$\Delta, &amp;&amp;$</td>
<td>AND operator</td>
<td>Conjunction of two events $E_1$ and $E_2$, occurs when both $E_1$ and $E_2$ occur, irrespective of their order of occurrence</td>
<td>SNOOP ODE SAMOS</td>
</tr>
<tr>
<td>ANY($m, E_1,E_2,...,E_n$)</td>
<td>ANY operator</td>
<td>Conjunction event, where $m\leq n$, occurs when $m$ events out of the $n$ distinct events specified occur, ignoring the relative order of their occurrence.</td>
<td>SNOOP</td>
</tr>
<tr>
<td>$E_1; E_2$, sequence($E_1,E_2$)</td>
<td>SEQ operator</td>
<td>Sequence of two events $E_1$ and $E_2$, occurs when $E_2$ occurs provided $E_1$ has already occurred.</td>
<td>SNOOP ODE SAMOS</td>
</tr>
<tr>
<td>$A(E_1,E_2,E_3)$</td>
<td>Non-Comulative Aperiodic operator</td>
<td>Occurrence of an aperiodic event within a closed time interval. Occurs when $E_2$ occurs within the time interval started by $E_1$ and ended by $E_3$.</td>
<td>SNOOP</td>
</tr>
<tr>
<td>$A^*(E_1,E_2,E_3)$</td>
<td>Cumulative Aperiodic operator</td>
<td>Occurrence of a cumulative aperiodic event within a closed time interval. Occurs only once when $E_3$ occurs and accumulates the occurrences of $E_2$ within the time interval started by $E_1$ and ended by $E_3$.</td>
<td>SNOOP</td>
</tr>
<tr>
<td>$P(E_1,\mbox{TI:parameters},E_3)$</td>
<td>Periodic operator</td>
<td>$P$ occurs for every $\mbox{TI}$ interval, starting after $E_1$ and ceasing after $E_3$. Parameters specified are collected each time $P$ occurs. If not specified,</td>
<td>SNOOP</td>
</tr>
<tr>
<td>Operator</td>
<td>Description</td>
<td>Example</td>
<td>Notes</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>$P^*(E1,TI[:parameters],E3)$</td>
<td>Cumulative Periodic operator</td>
<td>$P^<em>$ occurs only once when $E3$ occurs. Also, specified parameters are collected and accumulated at the end of each period and made available when $P^</em>$ occurs. Parameter specification is mandatory in $P^*$.</td>
<td>SNOOP</td>
</tr>
<tr>
<td>$\neg(E2) [E1,E3]$, $!$</td>
<td>Not operator</td>
<td>Detects the non-occurrence of the event $E2$ in the closed interval formed by $E1$ and $E3$</td>
<td>SNOOP ODE SAMOS</td>
</tr>
<tr>
<td>after keyword</td>
<td>After qualifier</td>
<td>Specifies an event that occurs immediately after the execution of a function specified by the keyword</td>
<td>ODE SAMOS</td>
</tr>
<tr>
<td>before keyword</td>
<td>Before qualifier</td>
<td>Specifies an event that occurs just before the execution of a function specified by the keyword</td>
<td>ODE SAMOS</td>
</tr>
<tr>
<td>prior $(E,F)$</td>
<td>Prior operator</td>
<td>Holds if the last logical event of $E$ occurs before the last logical event of $F$. The order in which the other events occur is irrelevant.</td>
<td>ODE</td>
</tr>
<tr>
<td>relative $(E,F)$</td>
<td>Relative operator</td>
<td>Requires the last logical event of $E$ occur prior to the first logical event of $F$.</td>
<td>ODE</td>
</tr>
<tr>
<td>relative const $(E)$</td>
<td>Relative operator</td>
<td>Specifies the composite event that consist of the $ith$ and any subsequent “$E$” event.</td>
<td>ODE</td>
</tr>
<tr>
<td>choose const $(E)$</td>
<td>Choose operator</td>
<td>Species the $ith$ occurrence of an event to be selected</td>
<td>ODE</td>
</tr>
<tr>
<td>every const $(E)$</td>
<td>Every operator</td>
<td>Used to specify events that occur every $const$ occasion</td>
<td>ODE SAMOS</td>
</tr>
<tr>
<td>$fa (E, F, G)$</td>
<td>Fa operator</td>
<td>Defines the first occurrence of event $F$ (at some logical operation $p$) relative to an event $E$, with no intervening event $G$ relative to $E$ taking place prior to logical event $p$.</td>
<td>ODE SAMOS</td>
</tr>
<tr>
<td>$faAbs (E, F, G)$</td>
<td>FaAbs operator</td>
<td>Defines the first occurrence of event $F$ (at some logical operation $p$) relative to an event $E$, with no intervening event $G$ relative to the whole history taking place prior to logical event $p$.</td>
<td>ODE</td>
</tr>
<tr>
<td>$(TIMES (n,E) IN 1)$</td>
<td>History operator</td>
<td>Is signaled each time $n$ events of $E$ have occurred during the time interval $I$.</td>
<td>SAMOS</td>
</tr>
</tbody>
</table>
Appendix B: Specification Pattern System

In SPS, scopes define the portion of a program over which the property holds. Patterns describe the structure of specific behaviors and define relationships between patterns. There are five types of scopes defined in SPS as shown in Table B1.

Table B1. SPS scope descriptions.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global</td>
<td>The scope consists of all the states of program execution.</td>
</tr>
<tr>
<td>Before R</td>
<td>The scope consists of the states from the beginning of the program execution until the state immediately before the state in which proposition R first holds.</td>
</tr>
<tr>
<td>After Q</td>
<td>The scope consists of the state in which proposition Q first holds and includes all the remaining states of program execution.</td>
</tr>
<tr>
<td>Between Q and R</td>
<td>The scope consists of all intervals of states where the start of each interval is the state in which proposition Q holds and the end of the interval is the state immediately prior to one in which proposition R holds.</td>
</tr>
<tr>
<td>After Q until R</td>
<td>Same as Between Q and R except, if there is a state in which Q holds and proposition R does not hold, then the interval of the scope will include all states from and including the state where Q last holds until the end of program execution.</td>
</tr>
</tbody>
</table>

Similarly, there are seven types of patterns defined in SPS as shown in Table B2. The patterns are divided into two groups: Occurrence and Order patterns. Occurrence patterns deal with single event or condition and specify the rate at which that condition or event occurs. The Occurrence patterns are Absence, Existence, and Universality. Order patterns relate two conditions or events and specify the order at which they occur. The Order patterns are Response, Precedence, Chain Response, and Chain.
Precedence. The Precedence pattern encompasses Precedence and Chain Precedence. Similarly, the Response pattern encompasses Response and Chain Response.

**Table B2. SPS pattern descriptions.**

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Absence</td>
<td>The pattern describes a portion of a system’s execution that is free of certain events or states.</td>
</tr>
<tr>
<td>Universality</td>
<td>The pattern describes a portion of a system’s execution which contains only states that have a desired property.</td>
</tr>
<tr>
<td>Existence</td>
<td>The pattern describes a portion of a system’s execution that contains an instance of certain events or states.</td>
</tr>
<tr>
<td>Precedence</td>
<td>The pattern describes a relationship between a pair of events/states where the occurrence of the first is a necessary precondition for an occurrence of the second.</td>
</tr>
<tr>
<td>Response</td>
<td>The pattern describes a cause-effect relationship between a pair of events/states. An occurrence of the first, the cause, must be followed by an occurrence of the second, the effect.</td>
</tr>
</tbody>
</table>
Appendix C: Composite Propositions

Composite Propositions (CP) extend SPS to capture sequential and concurrent behavior. CP can be defined as sets or sequences. CP can be composed of conditions or events. Conditions are propositions that hold in one or more consecutive states. Events are instants at which a proposition changes values in two consecutive states. CP defined as conditions are used to describe concurrency, while those defined as events are used to describe activation or synchronization of processes or actions. CP can be used to define boundaries or scopes and patterns with multiple propositions. The composite proposition taxonomy has twelve classes. The following are the descriptions of CP classes as defined by Mondragon:

Table C1. Composite propositions definitions.

<table>
<thead>
<tr>
<th>CP Name</th>
<th>CP Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AtLeastOneC(GS)</td>
<td>At least one of the propositions in the set holds.</td>
</tr>
<tr>
<td>AtLeastOneE(GS)</td>
<td>At least one of the propositions in the set becomes true.</td>
</tr>
<tr>
<td>ParallelC(GS)</td>
<td>All propositions in the set hold.</td>
</tr>
<tr>
<td>ParallelE(GS)</td>
<td>All propositions in the set become true simultaneously.</td>
</tr>
<tr>
<td>ConsecutiveC(GQ)</td>
<td>Each proposition in the sequence is asserted to hold in a specified order, one at each successive state.</td>
</tr>
<tr>
<td>ConsecutiveE(GQ)</td>
<td>Each proposition in the sequence becomes true in a specified order, one at each successive state. Once they become true, their truth value in subsequent states does not matter.</td>
</tr>
<tr>
<td>ConsecutiveC*(GS)</td>
<td>Each proposition in the set is asserted to hold, one at each successive state, and the ordering is non-deterministic.</td>
</tr>
<tr>
<td>ConsecutiveE*(GS)</td>
<td>Each proposition in the set becomes true, one at each successive state, and the ordering is non-deterministic. Also, once they become true, their truth value in subsequent states does not matter.</td>
</tr>
<tr>
<td>EventuallyC(GQ)</td>
<td>Each proposition in the sequence is asserted to hold in a specified order and in distinct and possibly non-consecutive states.</td>
</tr>
<tr>
<td>EventuallyE(GQ)</td>
<td>Each proposition in the sequence becomes true in a specified order and in distinct and possibly non-consecutive states. Once they become true, their truth value in subsequent states does not matter.</td>
</tr>
<tr>
<td>EventuallyC*(GS)</td>
<td>Each proposition in the set is asserted to hold in distinct and possibly non-consecutive states and the ordering is non-deterministic.</td>
</tr>
<tr>
<td>EventuallyE*(GS)</td>
<td>Each proposition in the set becomes true in distinct and possibly non-consecutive states and the ordering is non-deterministic. Also, once they become true, their truth value does not matter.</td>
</tr>
</tbody>
</table>
In the above definitions, the subscripts C and E indicate whether the propositions in the CP class are asserted as Conditions or as Events. The letters S and Q inside the parentheses indicate whether the propositions form a set of a sequence.
Appendix D: Description of Scientific Projects Included in Literature Survey

National Data Buoy Center (NDBC)

The NDBC [63] is part of the National Oceanographic and Atmospheric Administration (NOAA) National Weather Service (NWS). The NDBC operates 77 moored buoys and 57 Coastal-Marine Automated Network stations. The real-time NDBC data is used for preparing weather warnings, analyses and forecasts. The NDBC Quality Control (QC) program performs two types of real-time automated QC checks. The first category includes gross error checking that detects communication transmission errors and total sensor failure. Data flagged by these checks are virtually certain to be erroneous. The second category identifies data that may not be grossly in error, but for some reason, suspect. Data quality analysts further examine data falling in the second category.

Diagnostic measurements are acquired each hour to monitor station performance. Equipment remotely transmitted message data are checked for errors as a result of truncated or garbled messages. Also, equipment related data such as battery voltage and charge current are monitored by data quality analysts.

The most extensively used data quality check is the range limit check. A range limit check is a comparison between a measurement with pre-established upper and lower limits. The limits can be either hard or soft. A hard limit is set at three standard deviations from mean climatology values taken from the U.S Navy Marine Climatic Atlas of the World. A soft limit varies according to geographic area and season. Soft limits are usually set at two standard deviations from the mean climatology value for a specific area and month.

Time continuity checks track the change over time of a particular variable. NDBC has empirically derived limits that are used to check the time rate of change of several meteorological variables. NDBC uses two different time continuity algorithms; one on which a hard time continuity check is used, and a second one where the checks are derived using statistical formulations unique to the type of measurement.

Time continuity and range limit checks are highly coupled. Time continuity is always checked first. A measurement is hard flagged as a result of a failing limit check only if it has first failed the time
continuity check. The order prevents loss of data due to erroneous range limit flagging during severe weather conditions, since it is assumed that the onset of severe conditions will be gradual, and time continuity limits will not be exceeded.

The NDBC also performs Duplicate Sensor Checks. Weather stations often have duplicate sensors. Sensor degradation is sudden or is of such short duration that it might go undetected by the analyst. The NDBC identifies data from the primary sensor as degraded when it exceeds region and season specific differences in relation to the secondary sensor in a particular way. When such a condition occurs, the sensor hierarchy is reversed. Also the measurements from redundant sensors are flagged when either the difference between the measurements or the difference between time-continuity measurements exceed regional and seasonal limits.

Internal-consistency checks are also performed on data. Consistency checks are based on physical relationships between measurements. There are both hard and soft internal consistency checks.

**Marine Environmental Data Service (MEDS)**

The Marine Environmental Data Service (MEDS) [64] is a branch of Canada's federal Department of Fisheries and Oceans (DFO). The MEDS manages and archives ocean data collected by DFO, or acquired through national and international programs conducted in ocean areas adjacent to Canada. The MEDS disseminates: data, data products, and services to the marine community.

In MEDS, data quality assurance is a procedure of verification and validation. To validate, the data are formatted to an internal processing format. The data are checked to determine readability and interpretability. The data are verified for format errors in the original source form, or for invalid values. The system also captures inaccuracies caused by instrument noise or signal processing algorithms, and determines whether the observations are representative of the ambient conditions.

The MEDS verifies three main components over the ocean profile data. The first component examines the characteristics of the platforms track looking to identify errors in either position or time at real time. The data are ordered by call sign and within each cruise by date and time. Each cruise is verified for valid dates, latitude and longitude, station location and inferred speed.
The second component examines profiles of observations to identify suspected erroneous values. Profile checking verifies global ranges, bathymetry, single valued profiles and monotonically increasing depths for known parameter types (e.g., temperature, salinity, and oxygen). A set of statistical tests is applied, including regional range and global profile envelopes. Other tests look for spikes, pronounced gradients, density inversions, and temperature inversions.

The third component identifies duplication of profiles. Duplicate data records are often found to be the same observation, but differ in their method of analysis or reporting. The MEDS save all duplicate records, and flags all but the best one as duplicates.

**Tropical Atmosphere Ocean (TAO) Project**

The TAO [46] array consists of approximately 70 moorings in the Tropical Pacific Ocean, telemetering oceanographic and meteorological data at real time via the Argos satellite system. The array is a major component of the El Niño/Southern Oscillation (ENSO) Observing System, the Global Climate Observing System (GCOS), and the Global Ocean Observing System (GOOS).

Real-time data quality control is performed on a daily, weekly, and monthly basis. For daily, quality analysis data is automatically flagged against broad error specifications and narrower range of error specifications. Broad error specifications are preliminary gross automated error checking for data outside physically realistic ranges. Narrower range specifications are applied to specific weather parameters such as wind direction, relative humidity, air temperature and barometric pressure. Every week, the National Center for Environmental Prediction (NCEP) compiles statistics of transmitted TAO data and compares these statistics to numerical weather predictions. The statistics used include mean, median, standard deviation, variance, and minimum and maximum. Based on the statistics, suspect observations for wind, sea surface, and air are reported. Daily averaged data are plotted by site for the most recent 12 months, and continuity between deployments is checked. Plots of daily mean data are also verified. Trained analysts examine individual time series and statistical summaries. Data that have passed gross error checks but are unusual relative to neighboring data in the time series, and/or are statistical outliers, are further examined. Mooring deployment and recovery logs are searched for corroborating information about malfunctioning equipment.
Data Assembly Center (DAC) for Surface Meteorology

The DAC [40] for Surface Meteorology at Florida State University is charged with collecting, controlling quality, archiving, and distributing all underway-surface meteorological data from World Oceans Circulation Experiment vessels. Types of data collected include standard ship bridge observations, advanced automated systems, and all other practically obtained surface meteorological data. Data quality flagging is used to denote suspect or erroneous data instead of value replacement. The data quality process starts by flagging data during the conversion process from sensor data into the internal DAC format. Data is checked for inferred information, uncertain platform position/movement information, and data already flagged as questionable. After the incoming data are converted into the standard internal format, another QC procedure is applied. The QC procedure includes: verifying the existence of time, latitude, and longitude data for every record, flagging non-sequential or duplicate times, flagging data greater than four standard deviations from a climatology, flagging data that are not within a realistic range of values, flagging platform positions and speeds that are unrealistic, flagging positions where oceanographic platforms moves over land, and flagging inaccurate sensor measurements and relationships. Once the data passes the preprocessor flags, the data time series plots are analyzed and flagged. The subsequent QC checks include: flagging for discontinuity of data, flagging of interesting features in data, flagging erroneous data as “DO NOT USE”, flagging suspect data as “USE WITH CAUTION”, flagging for known instrument malfunction, and flagging for pronounced spikes in data.

Center for Collaborative Adaptive Sensing of the Atmosphere (CASA)

The CASA [44] is a National Science Foundation Engineering Research Center creating a new type of weather observation system featuring networks of low-power, low-cost radars that adaptively and collaboratively collect high-resolution data in the lowest 3 km of the atmosphere. In order to manage diverse user preferences and address resource conflicts, CASA uses user rules to specify in what manner and how often different kind of weather phenomena should be scanned by radars and users weights to establish the relative priorities of different user groups (e.g., emergency managers, and researchers) in case of resource conflict. In user rules, a rule trigger determines whether a rule is
activated based on a detected weather feature or an interval of time. The sector selection, elevation, and number of radars variables define how each radar should scan, and sample interval designates the periodicity of the rule.

**System of Control Oriented Oceanographic Parameters (SCOOP)**

The SCOOP system [45] was designed for the Centre National de Donnes Oceanographiques Francais. It is a validation assistance expert system. The SCOOP reads data ASCII files, and exports files in the same format, after adding data quality flags for each numeric value controlled. Initially, the SCOOP verifies the format of the file. Then, an automatic header check is performed. The automatic header check verifies: no double entries within a preset space-time radius occur, the date, the ship velocity between two consecutive stations and floor depth. Another automatic parameter check is performed once the header has been checked. The automatic parameter check ensures: at least two parameters are used, the abscissa (pressure or time) is monotonically increasing, parameters are within a certain bound, the profile is not constant (sensor jammed or faulty), peaks are detected, measurements match pre-existing climatological statistics for the zone, different measurements time series are met.

**U.S Air Force on the Easter Range Doppler Radar Wind Profiler**

The U.S Air Force operates a network of five Doppler radar wind profilers (DRWP) [41] in the Kennedy Space Center area. DRWP wind estimates can be affected by several factors such as traffic or trees. The Applied Meteorology Unit developed quality control routines for the data collected by the DRWP network. The automated QC process has three classes of components: DRWP criteria, single-gate atmospheric criteria, and multiple-gate atmospheric criteria. In DRWP criteria are based strictly on the design of the instrument and its signal processing algorithms. The criteria do not take the value of the wind estimate into consideration. The single-gate atmospheric criteria use the value of an individual wind estimate to determine if it is erroneous. No consideration is given to adjacent observations in space or time. Multiple-gate atmospheric criteria compare individual time estimates with spatially and temporally estimate. The success of multiple-gate criteria depends on the quality of the surrounding data; Thus, the DRWP and single-gate atmospheric criteria algorithms must be used first to eliminate erroneous data.
The Oklahoma Mesonetwork (Mesonet)

The Oklahoma Mesonetwork [65] was developed through a partnership between the University of Oklahoma and Oklahoma State University. Mesonet is a permanent mesoscale weather observation network. The Oklahoma Mesonet operates 115 stations on a continuous basis. Thirteen atmospheric and sub-surface variables are recorded every 5 minutes at each site, producing 288 observations of each parameter per station per day. Because of this continuous observation cycle, there is a need to ensure the quality of data coming from over 2500 instruments. For each datum, the Mesonet automatically generates a QA (Quality Assurance) flag that indicates the quality of the observation. The automated quality assurance system is composed of four components: laboratory calibration and testing, on-site intercomparison, automated QA, and manual QA. To perform the automated QA, five tests are performed: range, step, persistence, spatial, and like-instrument. The range test determines if an observation lies within a predetermined range. The allowable ranges are based on sensor specifications and annual climate extremes in Oklahoma. The step test uses sequential observations to determine which data represent unrealistic “jumps” during the observation time interval for each parameter. The persistence test analyzes data on a calendar day basis to determine if any parameter underwent little or no variation. The test uses either a standard deviation threshold or a predetermined “delta” threshold to determine the variation. The spatial test performs a Barnes objective analysis form each parameter at each observation time. For each analysis, an expected value is calculated based on data from surrounding stations. The expected value is compared with the actual observation. The difference between the two observations is used to determine the quality of the sample. The like-instrument test compares measurements from duplicate sensors; if the sensor values differ by more than a predefined threshold, the data are flagged. A decision-making algorithm compiles the results of the five tests, and logically determines a final flag for each datum. By using this approach, the results from the spatial and like-instrument tests can be used to determine malfunctioning equipment.

European Sea Level Service- Research Infrastructure (ESEAS-RI)

The ESEAS-RI [38] provides standardized quality control and access to a considerable fraction of the European tide gauge data set. The gauges data are used to provide tidal, storm surge and mean sea
level information. Errors in sea level data and related parameters can be measurement errors (e.g., transposition of numbers in manual recording or recording of observations in the wrong column), or could arise from electronic noise in measurements (e.g., problems in the communication, sensor calibration). Errors can also be systematic errors (e.g., errors arose from changes in observational practices, changes on the instrumentation, and changes in the environment surrounding the station). To ensure data quality, quality control procedures are performed at two stages, a collection time stage and a delayed mode stage. In the collection time quality control stage, the data quality procedures detect: strange characters, wrong assignment of date and hour, spikes, outliers, gaps, stabilization of the series. In the delayed mode stage, in addition to performing the quality procedures performed at real time, the following quality control and analysis procedures take place: interpolation of short gaps, filtering to hourly values, tidal analysis, computation and inspection of residuals, basic statistics (e.g., highs, lows, and extremes), computation of daily, monthly and annual means, comparison with neighboring tide gauges, comparison with models or predictions. Also, correlations are computed between the data from different stations or sensors and different parameters at the same station (e.g., wind, and atmospheric pressure).

Commonwealth Scientific and Industrial Research Organisation (CSIRO) Marine Laboratories

The CSIRO Marine and Atmospheric Research (CMAR) [66] houses Australia’s leading regional climate change modeling research teams. CMAR research aims to advance Australian climate, marine and earth systems science. The CMAR developed a QC cookbook for XBT (Expandable Bathythermograph) data. XBT are used by oceanographers to measure the temperature of the upper ocean. XBT are deployed in moving vessels, enabling broad scale coverage of the world’s ocean. To ensure the best possible data quality, the data are verified against various sets of quality control checks. The quality control checks are organized in six categories. The header information category is concerned with the correctness of the time, position and probe type identifier fields contained in the header of the XBT record. The recorder category is used to identify malfunctioning in any of the hardware or software components of the XBT system processor. The general profile category includes malfunctions that are routinely observed and that require interpolation, deletion or filtering of the data.
The general profile category includes wire breaks and various types of spikes introduced by outside electrical interference. The inversion or wire stretch category is used to distinguish between temperature inversions and equipment malfunction (wire stretch) that can look very similar. The structure or signal leakage category is used to distinguish between fine scale temperature structures and equipment malfunction (leakage) that can look very similar. Verification with neighboring profiles and previous knowledge of the region is used. Finally, the eddy, front or current temperature offset category is used to distinguish between changes in temperature caused by eddies, oceanic fronts or currents and equipment malfunction (e.g., wire stretch, leakage, or probe defect).

**Global Temperature-Salinity Pilot Project (GTsPP)**

The GTsPP [42] handles all temperature and salinity profile data. The data includes observations collected using water samplers, continuous profiling instruments, thermistor chain data and observations acquired using thermostalinographs. Before applying the quality control tests, basic scrutiny is applied to data. Data is applied a format checking procedure which ensures alphanumeric values occur where expected and no illegal characters are present. The quality control tests are grouped in stages. The first stage is concerned with determining that the position, the time, and the identification of a profile are correct. The second stage is concerned with resolving impossible values for variables (e.g., air pressure, current speed, salinity, and water temperature). The next stage examines the consistency of the incoming data with respect to references such as climatologies. The final stage looks at the internal consistency within the data set.

**Coastal Ocean Monitoring Center (COMC)**

The COMC [67] developed and operates a hydrological monitoring network around Taiwan coast. The goal of the hydrological network is to forecast severe sea-state for coastal hazard mitigation and help the government make correct policy for coastal area management. The network consists of nine deep-water buoy stations, one shallow-water pile station, ten coast weather stations and ten tide stations. Data quality control for the COMC is based on both objective criteria and human experience. Automated QC consists of two stages. The first stage is to examine raw time series, which includes the default basic check and data gap interpolation. The second stage examines statistical parameters derive
from the raw time series data, which include range rationality checks, variation continuity checks and physical correlation checks. Long-term QA includes: monthly data comparison with nearby sites, annual calibration of corresponding parameters, and sensors calibration.

**Atmospheric Radiation Measurement (ARM) Climate Research Facility (ACRF)**

The ARCF [43] is a national user facility for the study of global climate change. Research at this facility includes the study of alterations in climate, land productivity, oceans and other water resources, atmospheric chemistry, and ecological systems that may alter the capacity of the Earth to sustain life. The data quality office inspects and reviews approximately five thousand fields on a daily to weekly basis. Instruments are regularly checked for malfunctioning. If observed instrument values fall outside an expected range, the applicable data base field is marked as an observed problem. Currently, the hourly processing of automated quality control information is limited to checking for violations of simple range and maximum rate of change (delta) limits. A new system will statistically review historical data to improve quality control limits and to allow better detection of trends in data (e.g., calibration drifts). The quality control limits for data inspection purposes include time-varying and site-specific quality control limits.

**AmeriFlux**

The AmeriFlux [55] network provides continuous observations of ecosystem level exchanges of CO₂, water, energy and momentum spanning diurnal, synoptic, seasonal, and inter-annual time scales and is currently composed of sites from North America, Central America, and South America. AmeriFlux is part of a "network of regional networks" (FLUXNET) which coordinates regional and global analysis of observations from micrometeorological tower sites. In the Ameriflux network, Eddy covariance and profile data are typically screened for validity and removed when either Sonic Anemometer-Thermoheter (SAT) or fast response Infrared Gas Analyzer (IRGA) data flags indicate problem data, SAT or IRGA signals were out-of-range, rain occurred, or, 30-min data collection periods were incomplete. Other data quality diagnoses include: variance filters (signals are outside a preset, physically meaningful variance threshold, using Gaussian distributions), stability parameter, or some indication of non-stationary data (time series for wind statistics in steady state). Filling or averaging
routines are used to provide gap filling for data over long interval of times (i.e. mean diurnal variation, non-linear regressions, and look-up tables based on meteorological and seasonal conditions.
Appendix E: Eddy Covariance Experiment Detailed Results

Appendix E presents the detailed results associated with the Eddy Covariance Experiment conducted to demonstrate the use of data properties to detect anomalies in scientific sensor data.

Total Experiment Results

Table E1. Eddy covariance total experiment results.

<table>
<thead>
<tr>
<th>Experiment Totals</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Loading Time (milliseconds)</td>
<td>4204611</td>
</tr>
<tr>
<td>Total Verification Time (milliseconds)</td>
<td>68808631</td>
</tr>
<tr>
<td>Total Process Time (milliseconds)</td>
<td>73013242</td>
</tr>
<tr>
<td>Total Evaluation Calls</td>
<td>219800854</td>
</tr>
<tr>
<td>Total Anomalies Found</td>
<td>50857351</td>
</tr>
<tr>
<td>Total Verification Effectiveness</td>
<td>0.231379224</td>
</tr>
</tbody>
</table>

![Experiment Totals](image)

Figure E1. Total distribution of experimental loading time and distribution time.
Figure E2. Ratio of total data evaluations to total anomalies found.

Total Experiment Results by Data Property Types

Table E2. Total number of anomalies as identified by data properties.

<table>
<thead>
<tr>
<th>Data Property</th>
<th># of Anomalies</th>
<th>Season</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&lt;0.476</td>
<td>2058199</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&gt;0.463</td>
<td>3618041</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp!=0.0</td>
<td>2</td>
<td>Winter</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that atm_press&lt;50.034</td>
<td>244194</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that 50.0&lt;atm_press</td>
<td>3746330</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that</td>
<td>atm_press-press</td>
<td>&lt;2.0</td>
<td>5855612</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that t_hmp&lt;10.2</td>
<td>997837</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that 9&lt;t_hmp</td>
<td>4409387</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that</td>
<td>t_hmp-temp</td>
<td>&lt;1.5</td>
<td>262343</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that H2O&lt;4.63</td>
<td>1340770</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that 4.48&lt;H2O</td>
<td>4230945</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>Condition</td>
<td>Count</td>
<td>Season</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>--------</td>
<td>--------</td>
<td>---------------</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that UZ&lt;0.75</td>
<td>119278</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that -1.3&lt;UZ</td>
<td>14869</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that CO2&lt;625</td>
<td>4541037</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that 600&lt;CO2</td>
<td>24042</td>
<td>Winter</td>
<td>Datum</td>
</tr>
<tr>
<td>For all agc dataset values, it is always the case that agc &lt;70</td>
<td>162408</td>
<td>Winter</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==NAN</td>
<td>0</td>
<td>Winter</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==-99999</td>
<td>0</td>
<td>Winter</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all Ts dataset values, it is always the case that, if Ts&gt;10.2, then agc&lt;70</td>
<td>0</td>
<td>Winter</td>
<td>Data Dependent Instrument</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&lt;2.0</td>
<td>1051248</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&gt;1.1</td>
<td>605904</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp!=0.0</td>
<td>1</td>
<td>Summer</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that atm_press&lt;85.94</td>
<td>3772761</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that 85.89&lt;atm_press</td>
<td>2123522</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that</td>
<td>atm_press-press</td>
<td>&lt;2.0</td>
<td>22</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that t_hmp&lt;35</td>
<td>128625</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that 25&lt;t_hmp</td>
<td>3318309</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that</td>
<td>t_hmp-temp</td>
<td>&lt;1.5</td>
<td>2639963</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that H2O&lt;9.04</td>
<td>5277555</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that 8.57&lt;H2O</td>
<td>1078739</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that UZ&lt;2.0</td>
<td>13426</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that -2.0&lt;UZ</td>
<td>17894</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that CO2&lt;580</td>
<td>3986130</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that 480&lt;CO2</td>
<td>55689</td>
<td>Summer</td>
<td>Datum</td>
</tr>
<tr>
<td>For all agc dataset values, it is always the case that agc &lt;70</td>
<td>348003</td>
<td>Summer</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==NAN</td>
<td>0</td>
<td>Summer</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==-99999</td>
<td>0</td>
<td>Summer</td>
<td>Instrument</td>
</tr>
<tr>
<td>For all Ts dataset values, it is always the case that, if Ts&gt;35, then agc&lt;70</td>
<td>0</td>
<td>Summer</td>
<td>Data Dependent Instrument</td>
</tr>
</tbody>
</table>
Table E3. Total number of anomalies as identified by data categories.

<table>
<thead>
<tr>
<th>Data Category</th>
<th>Experiment Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datum</td>
<td>46774731</td>
</tr>
<tr>
<td>Instrument</td>
<td>510414</td>
</tr>
<tr>
<td>Data Relationship</td>
<td>8757940</td>
</tr>
<tr>
<td>Data Dependent Instrument</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure E3. Ratio of total anomalies by data property category.
Figure E4. Data property contribution to detection of data anomalies.
Total Experiment Results by Season-Winter

**Table E4.** Results of using SDVE to identify anomalies in EC winter sensor data.

<table>
<thead>
<tr>
<th>Winter Season</th>
<th>20100209</th>
<th>20100210</th>
<th>20100211</th>
<th>20100212</th>
<th>20100213</th>
<th>20100214</th>
<th>20100215</th>
<th>20100216</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total loading time (milliseconds)</td>
<td>120895</td>
<td>157627</td>
<td>164950</td>
<td>308636</td>
<td>316694</td>
<td>337651</td>
<td>356894</td>
<td>153982</td>
<td>1917329</td>
</tr>
<tr>
<td>Total verification time (milliseconds)</td>
<td>1966827</td>
<td>1959808</td>
<td>2090168</td>
<td>5652990</td>
<td>6088721</td>
<td>7568848</td>
<td>6041330</td>
<td>3202133</td>
<td>34570825</td>
</tr>
<tr>
<td>Total process time (milliseconds)</td>
<td>2087722</td>
<td>2117435</td>
<td>2255118</td>
<td>5961626</td>
<td>6405415</td>
<td>7906499</td>
<td>6398224</td>
<td>3356115</td>
<td>36488154</td>
</tr>
<tr>
<td>Total calls</td>
<td>6369671</td>
<td>7524143</td>
<td>7524143</td>
<td>16416312</td>
<td>16416312</td>
<td>16416312</td>
<td>16416312</td>
<td>7759243</td>
<td>94842448</td>
</tr>
<tr>
<td>Total violations</td>
<td>1569575</td>
<td>2028065</td>
<td>1878597</td>
<td>4622648</td>
<td>4540343</td>
<td>4571695</td>
<td>4772018</td>
<td>2456619</td>
<td>26439560</td>
</tr>
<tr>
<td>Total verification effectiveness</td>
<td>0.246414</td>
<td>0.269541</td>
<td>0.249676</td>
<td>0.281589</td>
<td>0.276575</td>
<td>0.278485</td>
<td>0.290688</td>
<td>0.316605</td>
<td>0.276197</td>
</tr>
</tbody>
</table>

**Figure E5.** Distribution of experimental loading time and distribution time for the winter season.

**Figure E6.** Daily ratio of total loading times to verification times for the winter season.
Table E5. Number of anomalies as identified by data properties for winter season.

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Winter Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&lt;0.476</td>
<td>2058199</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&gt;0.463</td>
<td>3618041</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp!=0.0</td>
<td>2</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that atm_press&lt;50.034</td>
<td>244194</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that 50.0&lt;atm_press</td>
<td>3746330</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that</td>
<td>atm_press-press</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that t_hmp&lt;10.2</td>
<td>997837</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that 9&lt;t_hmp</td>
<td>4409387</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that</td>
<td>t_hmp-temp</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that H2O&lt;4.63</td>
<td>1340770</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that 4.48&lt;H2O</td>
<td>4230945</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that UZ&lt;0.75</td>
<td>119278</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that -1.3&lt;UZ</td>
<td>14869</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that CO2&lt;625</td>
<td>4541037</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that 600&lt;CO2</td>
<td>24042</td>
</tr>
<tr>
<td>Fol all agc dataset values, it is always the case that agc&lt;70</td>
<td>162408</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==NAN</td>
<td>0</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==-99999</td>
<td>0</td>
</tr>
<tr>
<td>For all Ts dataset values, it is always the case that, If Ts&gt;10.2, then agc&lt;70</td>
<td>0</td>
</tr>
</tbody>
</table>

Table E6. Number of anomalies as identified by data categories for the winter season.

<table>
<thead>
<tr>
<th>Data Category</th>
<th>Season Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datum</td>
<td>25344929</td>
</tr>
<tr>
<td>Instrument</td>
<td>162410</td>
</tr>
<tr>
<td>Data Relationship</td>
<td>6117955</td>
</tr>
<tr>
<td>Data Dependent Instrument</td>
<td>0</td>
</tr>
</tbody>
</table>
Total Experiment Results by Season-Summer

Table E7. Results of using SDVE to identify anomalies in EC Summer sensor data.

<table>
<thead>
<tr>
<th>Summer Season</th>
<th>20100706</th>
<th>20100707</th>
<th>20100708</th>
<th>20100709</th>
<th>20100710</th>
<th>20100711</th>
<th>20100712</th>
<th>20100713</th>
<th>Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total loading time (milliseconds)</td>
<td>181682</td>
<td>282567</td>
<td>334412</td>
<td>255184</td>
<td>293460</td>
<td>380877</td>
<td>267346</td>
<td>291754</td>
<td>2287282</td>
</tr>
<tr>
<td>Total verification time (milliseconds)</td>
<td>2466096</td>
<td>4105263</td>
<td>437963</td>
<td>3879315</td>
<td>4105684</td>
<td>7071414</td>
<td>4632904</td>
<td>3632167</td>
<td>34237806</td>
</tr>
<tr>
<td>Total process time (milliseconds)</td>
<td>2647778</td>
<td>4387830</td>
<td>4682375</td>
<td>4134999</td>
<td>4399144</td>
<td>7452291</td>
<td>4897250</td>
<td>3923921</td>
<td>36525088</td>
</tr>
<tr>
<td>Total calls</td>
<td>10072646</td>
<td>16416312</td>
<td>16416312</td>
<td>16387888</td>
<td>16416312</td>
<td>16416312</td>
<td>16416312</td>
<td>16416312</td>
<td>124958406</td>
</tr>
<tr>
<td>Total violations</td>
<td>1407735</td>
<td>2889519</td>
<td>3040061</td>
<td>2884073</td>
<td>2929931</td>
<td>4908289</td>
<td>3505712</td>
<td>2852471</td>
<td>24417791</td>
</tr>
<tr>
<td>Total verification effectiveness</td>
<td>0.139758</td>
<td>0.176015</td>
<td>0.185185</td>
<td>0.175988</td>
<td>0.178477</td>
<td>0.298989</td>
<td>0.213551</td>
<td>0.173758</td>
<td>0.19271512</td>
</tr>
</tbody>
</table>

Figure E7. Distribution of experimental loading time and distribution time for the summer season.

Figure E8. Daily ratio of total loading times to verification times for the summer season.
**Table E8.** Number of anomalies as identified by data properties for the summer season.

<table>
<thead>
<tr>
<th>Data Property</th>
<th>Summer Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&lt;2.0</td>
<td>1051248</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp&gt;1.1</td>
<td>605904</td>
</tr>
<tr>
<td>For all e_hmp dataset values, it is always the case that e_hmp!=0.0</td>
<td>1</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that atm_press&lt;85.94</td>
<td>3772761</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that 85.89&lt;atm_press</td>
<td>2123522</td>
</tr>
<tr>
<td>For all atm_press dataset values, it is always the case that</td>
<td>atm_press-press</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that t_hmp&lt;35</td>
<td>128625</td>
</tr>
<tr>
<td>For all t_hmp dataset values, it is always the case that 25&lt;t_hmp</td>
<td>3318309</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that</td>
<td>t_hmp-temp</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that H2O&lt;9.04</td>
<td>5277555</td>
</tr>
<tr>
<td>For all H2O dataset values, it is always the case that 8.57&lt;H2O</td>
<td>1078739</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that UZ&lt;2.0</td>
<td>13426</td>
</tr>
<tr>
<td>For all UZ dataset values, it is always the case that -2.0&lt;UZ</td>
<td>17894</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that CO2&lt;580</td>
<td>3986130</td>
</tr>
<tr>
<td>For all CO2 dataset values, it is always the case that 480&lt;CO2</td>
<td>55689</td>
</tr>
<tr>
<td>For all agc dataset values, it is always the case that agc&lt;70</td>
<td>348003</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==NAN</td>
<td>0</td>
</tr>
<tr>
<td>For all diag_csat dataset values, it is never the case that diag_csat==99999</td>
<td>0</td>
</tr>
<tr>
<td>For all Ts dataset values, it is always the case that, If Ts&gt;35, then agc&lt;70</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table E9.** Number of anomalies as identified by data categories for the summer season.

<table>
<thead>
<tr>
<th>Data Category</th>
<th>Summer Totals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Datum</td>
<td>21429802</td>
</tr>
<tr>
<td>Instrument</td>
<td>348004</td>
</tr>
<tr>
<td>Data Relationship</td>
<td>2639985</td>
</tr>
<tr>
<td>Data Dependent Instrument</td>
<td>0</td>
</tr>
</tbody>
</table>
Vita

Irbis J. Gallegos was born in 1982 in Cd. Juarez, Mexico, as the oldest of two children to Ramon Gallegos and Rosario Payan. He attended elementary, middle, and high schools in Mexico, with the exception of the last two years of high school that he completed in El Paso, Texas, USA in the spring of 2000. He entered The University of Texas at El Paso (UTEP) in the fall of 2000 and received his Bachelor of Science (Cum Laude) in Computer Science degree in the spring of 2004. In the Fall of 2004, he entered the Ph.D. program in Computer Science under the guidance of Dr. Ann Q. Gates.

While pursuing his Ph.D. degree he worked in the Pan American Center for Earth and Environmental Studies (PACES) center and the Sharing Resources to Advance Research and Education through Cyberinfrastructure (Cyber-ShARE) center of excellence as a research assistant. As part of his interdisciplinary work with UTEP Systems Ecology Lab, he was a research intern in the Barrow Arctic Science Consortium (BASC). In collaboration with his colleagues, he has presented his research at multiple conferences and has seven peer-refereed conference publication. He also served as a teaching assistant for multiple Computer Science courses such as Data Structures and Computer Architecture I/Assembler Language. He also served as a peer leader coordinator for the Computer Science department and as a recruiting advocate for UTEP College of Engineering.

During his time at UTEP, he has been the recipient of numerous honors and awards such as the Alliance for Graduate Education and the Professoriate (AGEP) Fellowship, the National Science Foundation Minority Scholarship, the Patricia and Jonathan Rogers Scholarship, and the Dodson Fellowship. His profile appeared in several websites and publications including the Cyber-ShARE Center of Excellence Annual Report 2007-2008, the Computing Alliance for Hispanic-Serving Institutions (CAHSI) website, the UTEP Graduate School website, and the Winter 2006/Spring 2007 Minority in College issue of the Diversity/Careers in Engineering and Information Technology magazine.

Permanent address: 1345 Michelangelo
El Paso, TX, 79936

This dissertation was typed by Irbis Gallegos.