Data Provenance and Reliability in Sensor Networks

Emery Boose[1], Aaron Ellison[1], Leon J. Osterweil[2],
Lori A. Clarke[2], Rodion Podorozhny, Alexander Wise[2],
Julian Hadley[1], David Foster[1]

April 2007

[1] Harvard Forest, Harvard University, Petersham, MA
[2] University of Massachusetts Amherst, Amherst, MA
[3] Texas State University, San Marcos, TX

Introduction

Sensor networks that provide high resolution spatial and temporal measurements are revolutionizing environmental research and will soon support real-time environmental modeling and forecasting (Porter et al. 2005). The reliability of the resulting datasets (including both original measurements and derivative products) can be enhanced in a number of ways. For example, the physical network can be designed to minimize the number of missing or questionable values through inclusion of redundant sensors (so that measured values can be compared) or complementary sensors (so that measured values can be compared to modeled values). The data processing system can be designed to take advantage of such measurements and to support real-time quality control, modeling, and gap filling, as well as critical post-processing tasks such as correction for sensor drift and substitution for missing values.

More sophisticated methods are required to ensure that the processes used to create these datasets are in fact reproducible and scientifically sound. In our view this is a general and critical problem in science today that is compounded by ready access to powerful computer tools and networks and is especially acute for complex systems such as sensor networks. The problem can be addressed by (1) developing conceptual methods for defining processes using formal representations with sufficient accuracy and detail to support analysis and execution, and (2) developing cyberinfrastructure (CI) tools based on these definitions that scientists can actually use to help ensure that their analyses are sound and reproducible.

In our work we have developed a formal process definition (which we call an **analytic web**; Ellison et al. 2006) that consists of a series of coordinated graphs originally developed for use in software engineering (Ghezzi et al. 2003). We have also developed a prototype tool (SciWalker) that implements some of the features of this definition (Osterweil et al. 2005). The analytic web provides dataset provenance in the form of a complete audit trail of all artifacts (e.g., datasets, parameters, code, models) used or created in a particular execution of a process. It also supports dataset reliability through inclusion of detailed process metadata that precisely describes all sub-processes and that can be rigorously analyzed for various errors, including logical, statistical, and propagation of measurement errors. The essential features of the analytic web are illustrated below in the context of a sensor network to measure ecosystem water flux (for more details, see: Boose et al. 2007).

A Water Flux Sensor Network

At the Harvard Forest in central Massachusetts we are designing a sensor network that will integrate ongoing meteorological, hydrological, eddy flux, and tree physiological measurements to study the movement of water through a forest ecosystem. Simultaneous measurements in adjoining small watersheds will enable us to study variations in water flux caused by differences in topography, soils, vegetation, land use, and natural disturbance history. Frequent sampling will enable us to study water flux dynamics at a range of temporal scales, from minutes (e.g., evapotranspiration response to light) to days

(e.g., ground water response to precipitation and snow melt) to years (e.g., ecosystem response to climate, reforestation, land use, and natural disturbance). The data processing system will be designed to provide the best possible data and metadata online for real-time modeling and forecasting, with provision for updates and improvements as they become available.

For this discussion we focus on a simplified system consisting of a single watershed with no belowground inputs or outputs and three key measurements (the actual system will include multiple watersheds and a wider array of measurements). Here the water balance can be described as follows:

$$P - ET - Q = dS$$

where P = precipitation, ET = evapotranspiration, Q = stream discharge, and dS = change in ecosystem water storage (including ground water, soil moisture, surface water, snow pack, and vegetation). The terms in this equation may represent instantaneous rates or amounts integrated over a fixed time interval. Measurements will be made at a meteorological station, an eddy flux tower, and a stream gage. The meteorological station will provide two independent measures of precipitation (*P1, P2*). Evapotranspiration (*ET*) will be measured directly at the eddy flux tower and simultaneously modeled using linear regression against recently measured photosynthetically active radiation (*PAR*). Surface discharge (*Q*) will be measured directly at the stream gage and simultaneously modeled using a simple linear reservoir model and recently measured values of *P* and *Q*. Modeled values of *ET* and *Q* will be substituted for measured values if the latter are missing, out of range, or otherwise unsuitable (e.g., eddy flux measurements are not reliable under certain wind conditions; Ellison et al. 2006).

The data processing system will contain three subsystems: (1) A **real-time** subsystem will collect, analyze, and document data from the three measurement stations. This subsystem will retrieve measurements every 30 minutes, perform range checking, calculate best values from redundant sensors (*P1*, *P2*), create and apply models (*ET* and *Q*), choose between measured and modeled values (*ET* and *Q*), and calculate the change in water storage (*dS*) for each 30-minute period. (2) A **post-processing** subsystem will update models using before and after measurements. Real-time models are based on preceding measurements but experience has shown that such models can often be improved by using both preceding and subsequent measurements. This ongoing activity will occur daily, 30 days after the original measurement. (3) An **alternate** measurement subsystem will permit the system operator to substitute new values for original measurements, including corrections for sensor drift and replacement of missing values with data from other sites. Because measured values are changed through this activity, all models and derivative values within 30 days of the new measurement will also be updated to capture any possible "ripple effects."

Online data and metadata products will include: (1) all original and alternate 30-minute measurements with appropriate quality control metadata (e.g., value missing or out of range), (2) all original and updated daily models with unique identifiers plus date and time of creation, (3) an ordered dataset containing current best estimates of *P*, *ET*, *Q*, and

dS for each 30-minute period, updated regularly by the real-time and post-processing subsystems and as needed by the alternate measurement subsystem, and accompanied by metadata identifying source (e.g., real-time measurement, alternate measurement, modeled value), reliability (e.g., good, estimated, questionable), and history (e.g., processing time-stamp, models used).

An Analytic Web

An analytic web is a **scientific process definition**; i.e., it describes a process with sufficient accuracy and detail to support execution and analysis of the process and (if desired) re-execution to reproduce the original results. An analytic web consists of a **process-derivation graph** (PDG) that precisely describes the process and all sub-processes, and one or more **dataset-derivation graph**s (DDG) that provide a complete history of a single execution trace through the process. An optional third graph, the **data-flow graph** (DFG), offers a useful and intuitive overview of a process in the form of a standard flow chart. The DFG serves as the basis for most current workflow tools. However, its semantics are generally too weak to unambiguously represent execution pathways or the details of process coordination (including iteration parameters and conditions, concurrency, and exception management) without excessive annotation that can lead to severe loss of clarity (Boose et al. 2007).

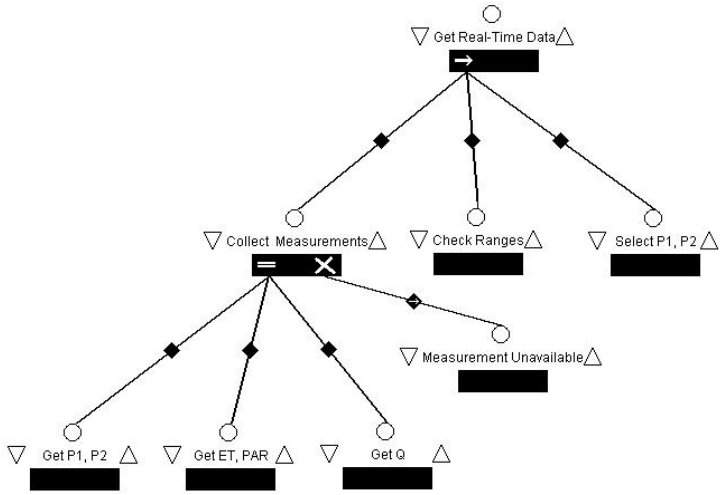

Fig. 1. A simplified PDG (in Little-JIL) for the **Get Real-Time Data** process.

Process-derivation graph. Our implementation of the PDG uses **Little-JIL**, a visual language for the coordination of agents (Wise et al. 2000, Wise 2006). Its distinguishing features include the use of scoping to clarify required inputs, facilities for specifying parallel processing, mechanisms for defining the handling of exceptional conditions, and the clarity with which iteration can be specified and controlled. A process is defined in Little-JIL using hierarchically decomposed steps, where each step represents a task to be done by an assigned agent. Each step has a name and a set of badges that represent such features as control flow among its sub-steps, its interface (a specification of its input and output datasets), prerequisite and post-requisite conditions, and the exceptions that it handles. A step with no sub-steps is called a leaf step and represents an activity to be

performed by an agent without any guidance from the process. An agent may be a human or it may be a computational tool that is executed with the designated input when the leaf step is encountered.

The PDG takes the form of an inverted tree in which process steps are decomposed into their sub-steps. It is a prospective structure of types that indicates the possible ways in which a process may be executed. As such it supports testing and validation of all possible execution paths (Dwyer et al. 2004) and analyses that can identify defects that could lead to invalid sequences of statistical processes (Oates and Jensen 1999). Such testing is particularly important in complex systems where manual examination of all possible paths is not feasible.

The simplified PDG in Fig. 1 shows the step decomposition for the Get Real-Time Data process of the water flux system. This process collects real-time measurements, performs range checking, and selects among the two precipitation measurements. Note that the left arrow in the Get Real-Time Data step bar indicates that its sub-steps are to be performed in sequential order (from left to right) while the equal sign in the Get Measurements step bar indicates that its sub-steps may be performed concurrently. The X in the Get Measurements step bar points to an exception handler (Measurement Unavailable) that contains a specification of what to do if a real-time measurement is not available (e.g., because of a communications timeout). The PDG also indicates how processing is to resume after completion of the exception handler. The complete PDG for the water flux system is considerably larger and contains significantly more annotations, but even this simplified fragment illustrates the clarity and flexibility of this approach.

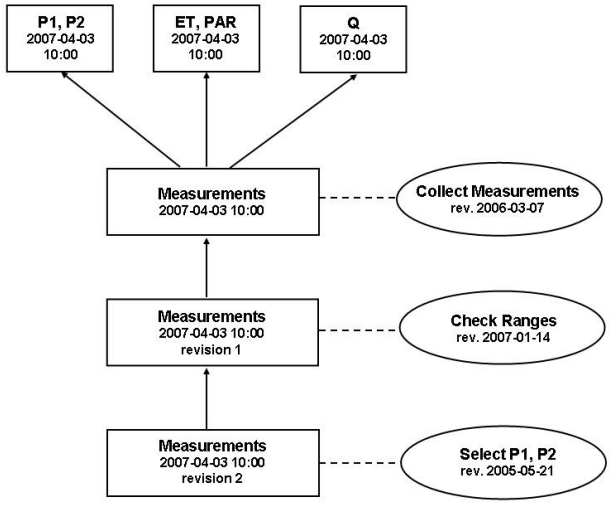

Fig. 2. A DDG corresponding to one execution of the PDG in Fig. 1.

Dataset-derivation graph. The DDG documents the specific datasets used or created in the execution of a process as well as the specific process steps whose execution (represented as a path through the PDG) resulted in the creation of each of these datasets. As such the DDG is a retrospective structure of instances. It provides a complete history

of a single execution trace but no information about other traces. Note that a separate DDG is created every time the process is executed.

The DDG in Fig. 2 shows the results of a single execution of the Get Real-Time Data process. In the figure, dataset instances are depicted as rectangles and process instances as ovals. Arrows connect a dataset instance to the dataset instance(s) from which it was derived. Process instances used to create a particular dataset are indicated by dotted lines. Note that the DDG takes the form of a genealogical tree in which the complete history of any dataset instance is contained in the portion of the tree above it. Though the current example is simple, a DDG for the complete water flux system would be considerably greater in size and complexity.

Discussion

In theory, the analytic web can document dataset provenance, ensure dataset reproducibility, and enhance dataset reliability even for systems of great complexity. The PDG is flexible, scalable, and largely self-documenting. Individual steps can be kept simple and the overall system design can be modified by changing the PDG. The PDG also provides an alternative to retaining versioned datasets (which may be impractical for streaming data or other data subject to frequent revision), since the combination of the PDG and all input dataset instances is sufficient to recreate all intermediate and final datasets (in effect, to recreate the DDG).

In practice, it remains to be seen whether the process metadata associated with an analytic web can be standardized and reduced in size and complexity to the point where it will gain acceptance from the scientific community. We are optimistic that standards and software tools can be developed that will greatly simplify the task of generating, managing, and interpreting such metadata. We believe this approach can be integrated into existing tools and evolving metadata specifications such as Ecological Metadata Language (EML), and that the need to do so will prove to be essential for emerging environmental observatories such as the National Ecological Observatory Network (NEON).

Acknowledgements

This work was supported by NSF grant CCR-0205575 and is a contribution from the Harvard Forest Long-Term Ecological Research (LTER) program.

References

Boose, E. R., A. M. Ellison, L. J. Osterweil, R. Podorozhny, L. Clarke, A. Wise, J. L. Hadley, and D. R. Foster. 2007. Ensuring Reliable Datasets for Environmental Models and Forecasts. In M. Jones (ed.), Novel Concepts of Ecological Data Management. Proceedings of 5th International Conference on Ecological Informatics. Elsevier (in review).

Dwyer, M.B., Clarke, L.A., Cobleigh, J.M. and Naumovich, G., 2004. Flow analysis for verifying properties of concurrent software systems. Association for Computing Machinery - Transactions on Software Engineering and Methodology 13, 359-430.

Ellison, A.M., Osterweil, L.J., Hadley, J.L., Wise, A., Boose, E.R., Clarke, L., Foster, D.R., Hanson, A., Jensen, D., Kuzeja, P., Riseman, E. and Schultz, H., 2006. Analytic webs support the synthesis of ecological datasets. Ecology 87, 1345-1358.

Ghezzi, C., Jazayeri, M. and Mandrioli, D., 2003. Fundamentals of software engineering. Pearson Education, Inc., Upper Saddle River, New Jersey.

Oates, T. and Jensen, D., 1999. Toward a theoretical understanding of why and when decision tree pruning algorithms fail. Pp. 372-378 in Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI99), Orlando, Florida.

Osterweil, L.J., Wise, A., Clarke, L., Ellison, A.M., Hadley, J.L., Boose, E.R. and Foster, D.R., 2005. Process technology to facilitate the conduct of science. In: M. Li, B. Boehm, and L.J. Osterweil (Editors), Lecture Notes in Computer Science - SPW 2005. Springer-Verlag, Berlin, Germany, pp. 403-415.

Porter, J., Arzberger, P., Braun, H., Bryant, P., Gage, S., Hansen, T., Hanson, P., Lin, C., Lin, F., Kratz, T., Michener, W., Shapiro, S. and Williams, T., 2005. Wireless sensor networks for ecology. BioScience 55, 561-572.

Wise, A., 2006. Little-JIL 1.5 language report. LASER, University of Massachusetts, Amherst, Massachusetts (http://laser.cs.umass.edu/techreports/06-51.pdf).

Wise, A., Cass, A.G., Lerner, B.S., McCall, E.K., Osterweil, L.J. and Sutton, S.M., Jr., 2000. Using Little-JIL to coordinate agents in software engineering. Proceedings of the Automated Software Engineering Conference (ASE 2000), Grenoble, France (http://laser.cs.umass.edu/techreports/00-45.pdf).