available at www.sciencedirect.com

www.elsevier.com/locate/ecolinf

Ensuring reliable datasets for environmental models and forecasts

Emery R. Boose[a,*], *Aaron M. Ellison*[a], *Leon J. Osterweil*[b], *Lori A. Clarke*[b], *Rodion Podorozhny*[c], *Julian L. Hadley*[a], *Alexander Wise*[b], *David R. Foster*[a]

[a]*Harvard University, United States*
[b]*University of Massachusetts, United States*
[c]*Texas State University, United States*

ARTICLE INFO

Article history:
Received 31 October 2006
Received in revised form
10 July 2007
Accepted 30 July 2007

Keywords:
Analytic web
Little-JIL
Metadata
Process
Sensor network
Water flux

ABSTRACT

At the dawn of the 21st century, environmental scientists are collecting more data more rapidly than at any time in the past. Nowhere is this change more evident than in the advent of sensor networks able to collect and process (in real time) simultaneous measurements over broad areas and at high sampling rates. At the same time there has been great progress in the development of standards, methods, and tools for data analysis and synthesis, including a new standard for descriptive metadata for ecological datasets (Ecological Metadata Language) and new workflow tools that help scientists to assemble datasets and to diagram, record, and execute analyses. However these developments (important as they are) are not yet sufficient to guarantee the reliability of datasets created by a **scientific process** — the complex activity that scientists carry out in order to create a dataset. We define a dataset to be **reliable** when the scientific process used to create it is (1) reproducible and (2) analyzable for potential defects. To address this problem we propose the use of an **analytic web**, a formal representation of a scientific process that consists of three coordinated graphs (a data-flow graph, a dataset-derivation graph, and a process-derivation graph) originally developed for use in software engineering. An analytic web meets the two key requirements for ensuring dataset reliability: (1) a complete audit trail of all artifacts (e.g., datasets, code, models) used or created in the execution of the scientific process that created the dataset, and (2) detailed process metadata that precisely describe all sub-processes of the scientific process. Construction of such metadata requires the semantic features of a high-level process definition language.

In this paper we illustrate the use of an analytic web to represent the scientific process of constructing estimates of ecosystem water flux from data gathered by a complex, real-time multi-sensor network. We use Little-JIL, a high-level process definition language, to precisely and accurately capture the analytical processes involved. We believe that incorporation of this approach into existing tools and evolving metadata specifications (such as EML) will yield significant benefits to science. These benefits include: complete and accurate representations of scientific processes; support for rigorous evaluation of such processes for logical and statistical errors and for propagation of measurement error; and assurance of dataset reliability for developing sound models and forecasts of environmental change.

* *Corresponding author*.
E-mail address: boose@fas.harvard.edu (E.R. Boose).

1574-9541/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.ecoinf.2007.07.006

1. Introduction

At the dawn of the 21st century, more environmental scientists are collecting more data at more rapid rates than at any time in the past. The pace of environmental change is also accelerating. Policy analysts and decision-makers require robust and reliable models of ongoing ecological dynamics and forecasts of environmental change (Clark et al., 2001). Uncertainty in these forecasts will decline as more data are accumulated and models are updated and run again (Ellison, 1996). Economic markets are evolving that will take advantage of these forecasts to set prices for scarce natural resources, emissions levels, and restoration efforts (Cacho et al., 2003). The lone ecologist studying obscure organisms in pristine sites who analyzes simple datasets and publishes results in technical journals read by few is quickly being replaced by teams of investigators who are studying “real-world” environmental problems, analyzing terabytes of data in (near) real time, and communicating their results to vastly broader audiences, all with the aid of new information technologies.

Great progress has been made in recent years in the development of standards, methods, and tools for data analysis and synthesis (Jones et al., 2006). Descriptive metadata that provide essential information about the contents of individual or aggregated datasets (Michener et al., 1997) can now be represented using a community standard, Ecological Metadata Language (EML).[1] New scientific workflow tools help scientists to assemble datasets and to diagram, record, and execute analyses. Perhaps the most notable of such tools for ecologists is the open-source Kepler,[2] which is able to utilize EML directly (Altintas et al., 2004; Ludäscher et al., 2005). There is also growing interest in developing permanent archives of environmental models to facilitate verification of past results and development of future models (Thornton et al., 2005).

Nevertheless these important developments are not sufficient to ensure the reliability of datasets created through a *scientific process* — the complex activity that scientists carry out in order to create a dataset. We define a dataset to be *reliable* when the scientific process used to create it (both the overall process and the specific execution trace) is (1) reproducible and (2) analyzable for potential defects. By reproducible, we mean that the reported results could be replicated exactly by an independent entity (Schwab et al., 2000; NRC, 2003). By analyzable, we mean that the process could be rigorously evaluated for potential defects such as undesirable outcomes or logical and statistical errors (Dwyer et al., 2004).

EML does not have a formal structure for accurately capturing the processes used by scientists to analyze and synthesize data, update existing models, or create new models. Workflow tools such as Kepler are typically based on data-flow graphs, representations that are unable to capture all of the essential details of many current and increasingly complex scientific processes (see below). Further, the archiving of models, even with extensive documentation and versioning, does not tell us exactly how they were used in a particular application.

The goal of dataset reliability can be realized by recording and archiving two sets of information: (1) a complete audit trail of all *artifacts* (e.g., datasets, code, statistical or simulation models) used or created in the execution of the scientific process that created the dataset, and (2) detailed *process metadata* that completely and precisely describe all sub-processes of the scientific process. We believe that accomplishing this goal will require advances on several fronts. First, we need conceptual methods for defining processes using formal representations with sufficient accuracy and detail to support analysis and execution (the focus of this paper). Second, we need community standards for expressing these definitions in a persistent and platform-independent form such as XML. Third, we need cyberinfrastructure tools based on these definitions that scientists can actually use to help ensure that their analyses are sound and reproducible. These advances would provide critical benefits to science, including complete and accurate records of scientific processes; support for rigorous evaluation of such processes for logical and statistical errors and for propagation of measurement errors; and assurance of dataset reliability for developing sound models and forecasts of environmental change.

In our work to date we have developed a software prototype (SciWalker) that creates a complete audit trail for the execution of a scientific process (Osterweil et al., 2005), and we have applied SciWalker to the analysis of archival carbon flux data derived from eddy-covariance towers (Ellison et al., 2006). Rudiments of this relatively simple scientific process were easily captured with the traditional data-flow graph widely used in workflow tools such as Kepler. However, new challenges arise when there is a need to create audit trails and to construct process metadata for more complex systems such as the real-time analysis and modeling of data delivered continuously by sensor networks. In this paper we illustrate these challenges in the context of a real-time sensor network for measuring ecosystem water flux, and we use the Little-JIL language (Wise et al., 2000) to precisely and accurately capture the analytical processes involved. We believe this approach can be integrated into existing tools and evolving metadata specifications (such as EML) and that the need to do this will prove to be essential for emerging environmental observatories such as the National Ecological Observatory Network (NEON).[3]

2. A water flux sensor network

Water is essential to life, and the movement of water through ecosystems has long been studied by scientists. In the past, for practical reasons, such studies have been limited to specific ecosystem components (e.g., stream discharge) or to relatively long time frames (e.g., annual water budgets). In recent years, new technologies and methods have made it possible to measure elements of the water cycle that previously were difficult to measure directly, including evapotranspiration (e.g.,

1 http://knb.ecoinformatics.org/software/eml/.
2 http://www.kepler-project.org/.
3 http://www.neoninc.org/.

using eddy-covariance measurements; Wilson et al., 2000) and sap flow in trees (e.g., using measurements of thermal conductivity; Wilson et al., 2001; Phillips et al., 2004). Current advances in sensor network technology promise a paradigm shift in environmental research (Atkins et al., 2003; Estrin et al., 2003) in which the ability to conduct simultaneous measurements over broad areas and at high sampling rates will open up entirely new areas for investigation (Porter et al., 2005). Moreover the ability to collect and process this information in (near) real time holds considerable promise for real-time environmental modeling and forecasting, as well as for demonstration and education (Cayan et al., 2003). However it also presents significant challenges for how to manage and document the analysis of streaming data in real time.

In this section we describe a planned sensor network for measuring real-time ecosystem water flux at the Harvard Forest Long-Term Ecological Research (LTER) site in Petersham, Massachusetts, USA (Fig. 1). This system will integrate ongoing meteorological, hydrological, eddy flux, and tree physiological measurements. 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. High-frequency sampling will enable us to study water flux dynamics at a wide range of temporal scales, from minutes (e.g., response of evapotranspiration to light) to days (e.g., groundwater response to precipitation and snow melt) to years (e.g., ecosystem response to climate, reforestation, land use, and natural disturbance). Data will be collected via a field-based wireless network, processed using our analytical web tools (see below), and immediately posted to the Internet for real-time use by the scientific community and the public.

Here we illustrate the data requirements for a simplified system consisting of a single watershed and three key measurements (the actual system will encompass multiple watersheds and a wider array of measurements). The water balance for a single watershed can be described as follows:

$$P - ET - Q = dS \qquad (1)$$

where P=precipitation, ET=evapotranspiration, Q=stream discharge, and dS=change in ecosystem water storage (including groundwater, 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 (the equation as formulated ignores belowground inputs and outputs).

Measurements will be made at a meteorological station, an eddy flux tower, and a stream gauge. Because accurate and complete measurements of precipitation (P) are critical, two independent rain gauges will be used at the meteorological station, and rules established for what to do if the two measurements of precipitation ($P1$, $P2$) do not agree. For example, a difference of more than a specified amount would

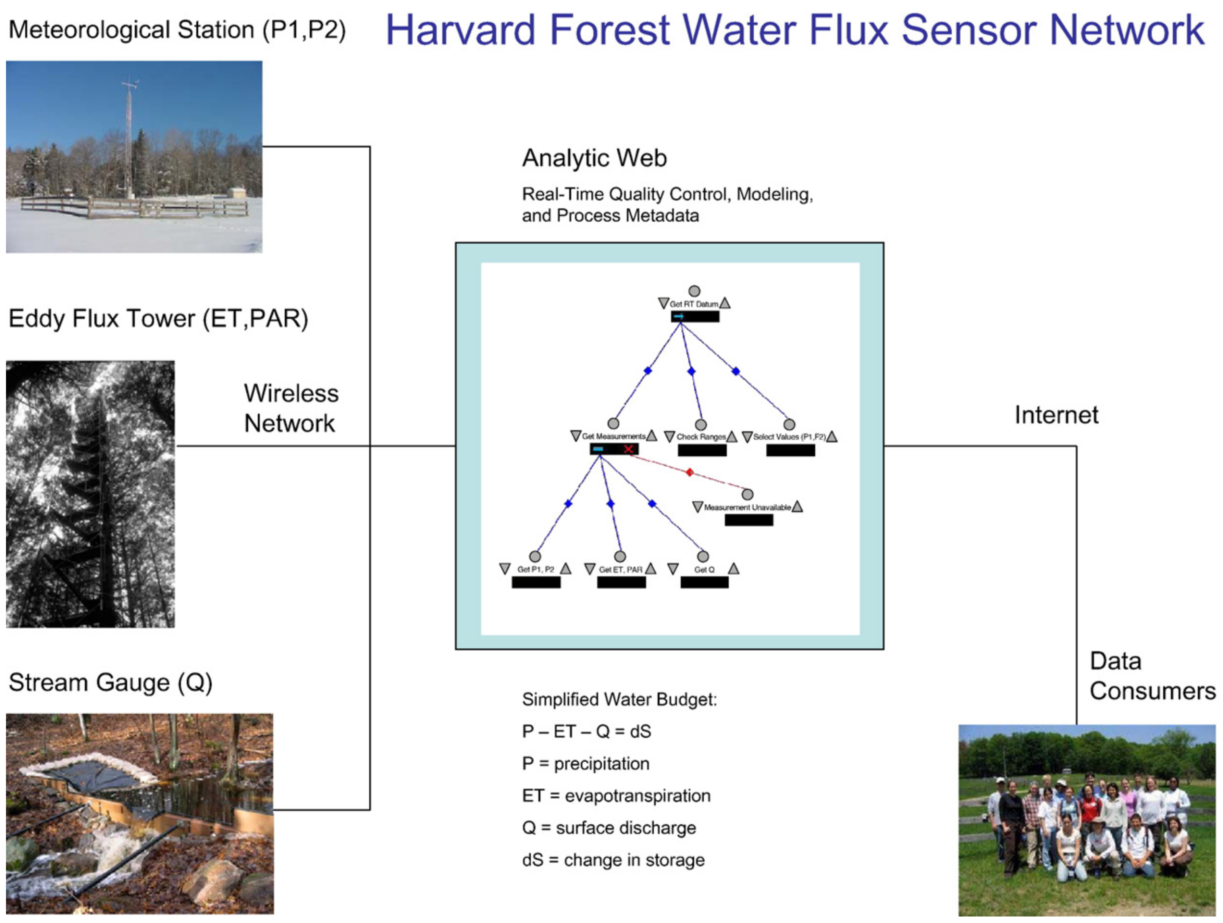

Fig. 1 – A conceptual diagram of the planned water flux sensor network at the Harvard Forest.

generate a warning of possible sensor failure to the system operator. Evapotranspiration (*ET*) will be measured at the eddy flux tower and modeled using linear regression on recent measured photosynthetically active radiation (*PAR*; actual models will be based on *PAR* and vapor pressure deficit; J. Hadley, unpublished data). Surface discharge (*Q*) will be measured with a stream gauge and modeled using a simple linear reservoir model (Dingman, 2002) in which recent measured values of *P* and *Q* are used to estimate base flow and the fraction of precipitation returned as event flow (E. Boose, unpublished data).

The system for processing data from the sensor network is designed to present the best possible data and metadata online in real time. A key feature is modeling *ET* and *Q* to allow for real-time quality control and to enable interpolation and gap filling when measured values are unreliable or when sensors fail. For example, the modeled value would be substituted for the measured value if the latter were missing, out of range, or otherwise unsuitable (e.g., eddy flux measurements are not reliable under certain wind conditions; see Ellison et al., 2006); in other cases a difference of more than a specified amount would generate a warning to the system operator of possible sensor failure or a new phenomenon not predicted by the model. Note that actual measurements at each station are preprocessed by onsite dataloggers and/or computers, and "measured values" are 30-minute averages.

The data collection and analysis system will contain the following three subsystems: (1) A *real-time* subsystem will collect, analyze, and document data from the meteorological station, eddy flux tower, and stream gauge. This subsystem will retrieve measurements every 30 min, perform range checking, calculate best values from redundant sensors (*P1*, *P2*), create and apply models as described above (for *ET* and *Q*), choose between measured and modeled values (of *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 by necessity are based on preceding measurements, but experience has shown that such models can often be improved by using both preceding and subsequent measurements, especially during periods of rapid ecosystem change (e.g., during spring leaf-out or when soils become saturated by heavy precipitation). This ongoing activity is scheduled to take place 30 days after the original measurement; because new measurements are constantly being accumulated, new post-processing will occur daily. (3) An *alternate* measurement subsystem will permit the system operator to substitute new values for original measurements. For example, measurements may arrive too late for real-time processing, original measurements may require subsequent correction for sensor drift, and missing or questionable measurements may need to be replaced with data from another site. Because measured values (as opposed to modeled values) are changed through this activity, all models and modeled values within 30 days of the new measurement must also be updated to capture any possible "ripple effects".

Online data and metadata products will include the following: (1) All original and alternate 30-minute measurements will be retained, with appropriate quality control metadata (e.g., value missing or out of range). (2) All original and updated daily models will be retained, with unique identifiers and date and time of creation. (3) An ordered dataset will contain 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. Associated metadata will identify data source (e.g., real-time measurement, alternate measurement, modeled value), confidence level (e.g., good, estimated, questionable), and history (e.g., processing time-stamp, models used). Metadata will be generated on the fly using an analytic web.

3. An analytic web

A *scientific process definition* is a formal representation of a scientific process (in the form of structured metadata) that completely and accurately describes the process and is sufficient to support execution of the process and the re-execution required to reproduce the original results. Such a definition is provided by what we call an *analytic web* (Osterweil et al., 2005; Ellison et al., 2006) which consists of three coordinated graphs – a *data-flow graph (DFG)*, a *dataset-derivation graph (DDG)*, and a *process-derivation graph (PDG)* – all originally developed for use in defining and controlling software development projects (Ghezzi et al., 2003). The three graphs and the rigorous analytical information that may be inferred from them are sufficient to render a dataset reliable. In this section we discuss the strengths and limitations of each graph in the context of the water flux sensor network.

3.1. *The data-flow graph*

The most familiar of the three graphs is the data-flow graph (DFG), a representation that also forms the basis for workflow tools such as Kepler. The DFG defines the sequence by which processes (perhaps as simple as the application of a specific tool) are applied to raw and intermediate datasets to create a final dataset. Note that the DFG identifies dataset and process *types*, rather than specific *instances*. For example, the DFG might specify that a process type interpolate via linear regression be applied to a dataset type eddy flux data. The DFG specifies only that a process for interpolating via linear regression must be used; it does not specify which specific tool is used to execute the process. However when the general description of each dataset type in a DFG is associated with (or *bound to*) a specific dataset and the general description of each process type is bound to a specific process, the DFG and this *binding* information describes the sequence of activities to be performed. In a simple scientific process, the DFG and binding information are sometimes sufficiently complete and precise to specify what must be done in order for the process to be executed automatically by a DFG interpreter (e.g., available in SciWalker). In such cases, the DFG interpreter accesses the appropriate input datasets (perhaps via the Internet), sends them to the appropriate processes (and/or tools), initiates execution of those processes (and/or tools) on the datasets, and finally stores and transmits the derived datasets (again perhaps via the Internet).

Though useful and easy to grasp, the DFG has two important limitations: (1) Most DFGs incorporate ambiguities that make it impossible to be sure which, of a number of possible alternatives, is *the* sequence of processing steps that was actually used in the particular execution that led to the creation of a particular output dataset. The dataset-derivation graph (DDG; see next section) of an analytic web captures this information. (2) The DFG is unable to represent the coordination of processes (e.g., parallel processing, multiple interactive loops, exception handling) without excessive annotation. In general, scientific processes that make significant use of such features cannot be fully represented by the DFG or executed by a DFG interpreter without (at the very least) substantial difficulties that can lead to severe loss of clarity. As described below, a process-derivation graph (PDG) can capture this information by drawing on the greater semantic capabilities of a language developed expressly for process definition, such as Little-JIL (Wise et al., 2000).

The system for measuring and modeling water flux illustrates both the utility and the limitations of a DFG (Fig. 2). The overall design of the system is clearly captured in the graph: e.g., the four major loops correspond to the three major subsystems plus a fourth activity to create and apply models, whereas Revised Data represents the current best data to be provided to users. However it would be impossible to determine the derivation history of a particular datum in Revised Data based on the DFG alone. More importantly, the DFG does not specify how the various processes are initiated, controlled, or coordinated. For example: how is model-building coordinated with real-time data processing? When alternate measurements are substituted for original measurements, how is the required updating of neighboring models and modeled values initiated and controlled? What happens if a process (e.g., retrieval of a real-time measurement) is unable to complete successfully? Do the three major subsystems operate independently of one another? Clearly there are a lot of critical questions that cannot be answered easily, if at all, based solely upon information provided by the DFG.

3.2. *The dataset-derivation graph*

The dataset-derivation graph (DDG) documents the specific datasets used and created in the execution of a scientific process as well as the specific process steps whose execution (represented as a path through the DFG) resulted in the creation of each of these datasets. Where the DFG documents dataset types (e.g., eddy flux data), the DDG documents dataset instances (e.g., eddy flux data collected on 1 Oct 2006 at the Harvard Forest). Note that a new DDG, containing a new set of instances, is generated every time the scientific process is executed. The instances represented as DDG nodes may be stored independently with a unique URL (uniform resource locator) or DOI (digital object identifier). The DDG contains the minimal process metadata required to support reproduction

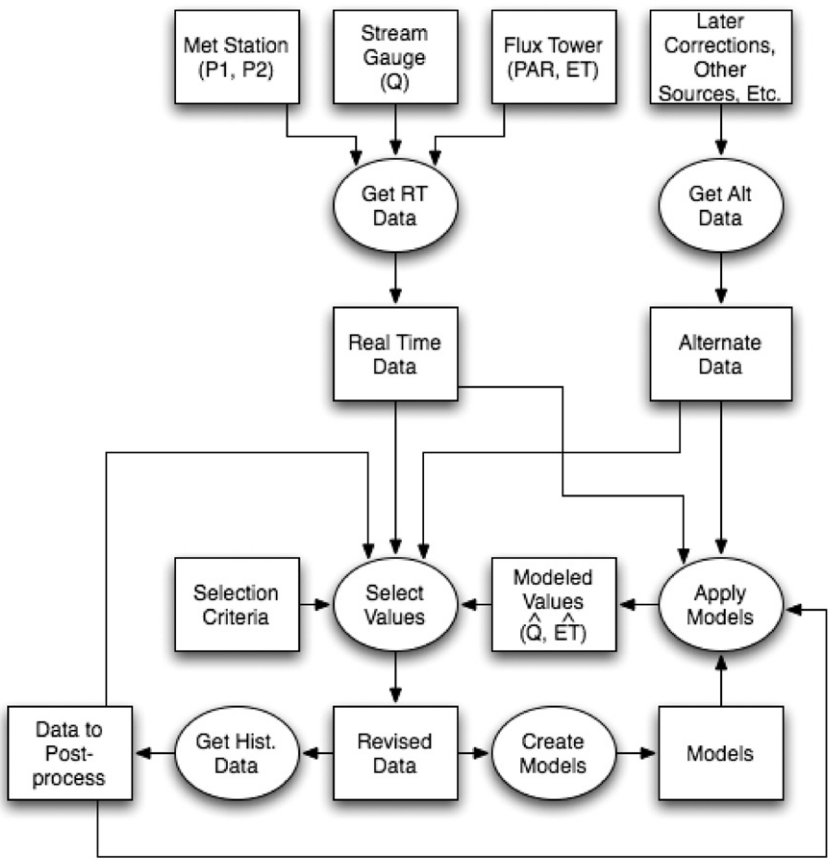

Fig. 2 – A data-flow graph for the water flux system. The icons represent dataset types (boxes) or process types (ovals). The arrows indicate the flow of datasets into and out of processes.

of a particular dataset and can be produced automatically with a DFG interpreter in a tool such as SciWalker.

The DFG and DDG address complementary needs. The DDG provides the specific information needed to reproduce particular datasets or to verify models of interest. The DFG provides the type information needed to replicate the scientific process using different input datasets or different processing tools (as is commonly done in model validation; Horn et al., 1989). The reapplication of a scientific process using alternative dataset or tool bindings is easily accomplished with a tool such as SciWalker. Though the DDG provides complete information about one specific trace through the process, it provides no information about other traces. The testing and validation of all possible traces through the scientific process could be done based on a DFG representation, but a more thorough evaluation requires a more formal structure, such as that provided by a PDG.

The DDG excerpt for the water flux system shows that the size and complexity of the DDG may be considerable (Fig. 3). The figure illustrates the results of two iterations of DFG looping, resulting in the creation of two instances of Revised Data. The first instance (1) is a product of the real-time subsystem, while the second instance (2) is a product of the post-process subsystem. Note that the modeled values also have complex derivation histories of their own. Moreover, additional instances of Revised Data could result from the introduction of alternate measurements.

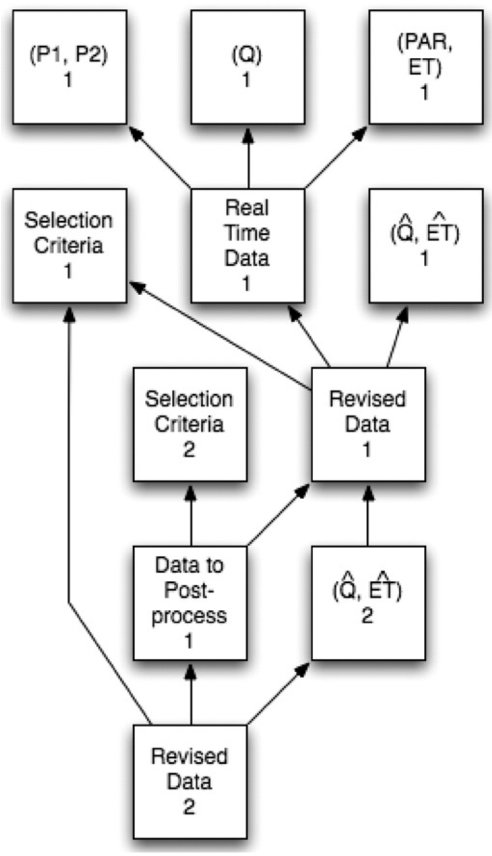

Fig. 3 – An excerpt from a dataset-derivation graph for the water flux system. The icons represent dataset instances. Each arrow indicates the application of a process instance. Arrows connect a dataset instance to the dataset instance(s) from which it was derived.

3.3. *The process-derivation graph and Little-JIL*

By incorporating a stronger and broader set of semantic features than the DFG, the process-derivation graph (PDG) provides a complete and precise specification of all possible executions of a scientific process. As noted above, the relatively crude semantics of the DFG lack facilities for clear and concise specification of iteration parameters and conditions, concurrency, and exception management. The DFG may also contain paths whose execution would not produce valid datasets and that are not intended for actual execution; the DFG lacks facilities to prevent the execution of such paths. In addition, the DFG is unable to specify the particular dataset instances to be used as input to a scientific process or to identify the particular dataset instances produced as output. Such information is critical for the large set of environmental models created by a process of iterative reconsideration or reevaluation, where it is imperative to know which datasets are used as input and which are created as output (Peterson et al., 2003). The PDG, as implemented here with the process definition language Little-JIL, provides semantic features that are adequate to address these specification needs. It can also support rigorous analysis of all possible execution paths through the use of tools such as the FLAVERS finite-state verification system (Dwyer et al., 2004). Such analysis can be used, for example, to eliminate defects that could otherwise lead to invalid sequences of statistical processes and unsound scientific results (Oates and Jensen, 1999).

Little-JIL is a visual language for the coordination of agents (Wise et al., 2000; Wise, 2006). Its semantics are precisely defined using finite-state automata, thereby rendering it able to support precise definitions of processes. Among its distinguishing features are its use of scoping to make the use of required dataset inputs clear, its facilities for specifying parallel processing and for defining the handling of exceptional conditions, and the clarity with which iteration can be specified and controlled. Little-JIL follows earlier efforts to develop languages and diagrammatic notations for defining processes, including procedural languages (Sutton and Osterweil, 1997), rules (Ben-Shaul and Kaiser, 1994), functional decomposition (Suzuki and Katayama, 1991), data-flow diagrams (Diamant et al., 1990), and modified Petri Nets (Bandinelli et al., 1993). However none of these earlier efforts provide the clarity and precision needed for the PDG.

A process is defined in Little-JIL using hierarchically decomposed *steps* (Fig. 4), where a step represents a task to be done by an assigned agent. Each step has a name. A set of *badges* represents control flow among its sub-steps, its interface (a specification of its input and output datasets), the exceptions it handles, etc. 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. Other key features of the language are outlined below.

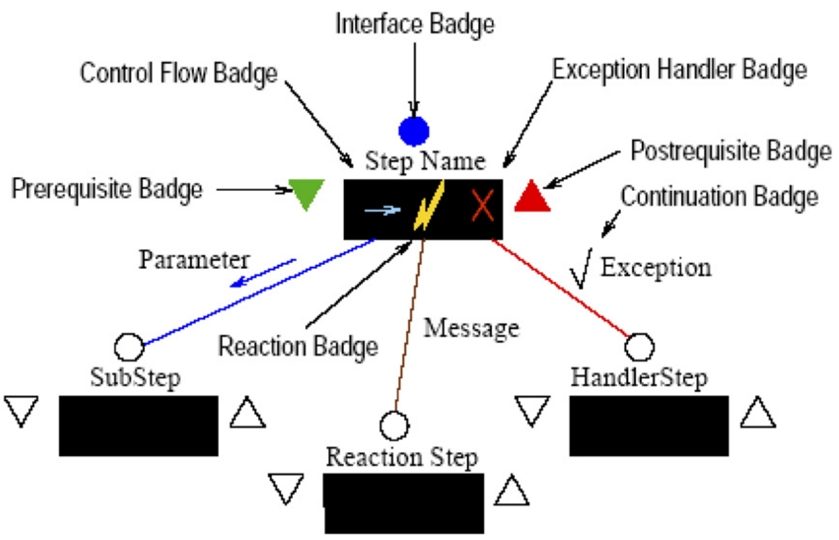

Fig. 4 – A Little-JIL step icon (from Wise et al., 2000).

3.3.1. *Step sequencing*

Every non-leaf step has a sequencing badge (an icon embedded in the left portion of the step bar) which defines the order in which its sub-steps execute. For example, a sequential step (right arrow) indicates that its sub-steps are to be executed sequentially from left to right; the step is completed only after all of its sub-steps have completed. A parallel step (equal sign) indicates that its sub-steps can be executed in any (possibly arbitrarily interleaved) order. It, too, is completed only after all of its sub-steps have completed. A choice step (line through circle) indicates that the agent executing the step is to make a choice among sub-steps, while a try step (right arrow through X) mandates the sequence in which sub-steps are to be tried.

3.3.2. *Artifacts and artifact flows*

Artifacts are entities (e.g., a datum, data, or datasets) that are used or produced by the step. The artifacts used by the step (IN parameters) or produced by the step (OUT parameters) are declared in the step interface (circle atop the step bar). In addition, the flow of artifacts between parent and child steps is indicated by attaching artifact identifications and appropriate arrows to the edges (lines) between parent and child.

3.3.3. *Requisites*

A step optionally can be preceded and/or succeeded by a step that is executed before and/or after (respectively) the execution of the main body of the step. A prerequisite is represented by a downward arrowhead to the left of the step bar, and a post-requisite is represented by an upward arrowhead to the right of the step bar. Requisites enable the checking of a specified condition either as a precondition for step execution or as a post-execution check to assure that the execution has been acceptable. If a requisite fails, an exception is triggered.

3.3.4. *Exception handling*

A step can signal the occurrence of exceptional conditions when there are aspects of the step's execution that fail (e.g., violation of one of the step's requisites). This triggers the execution of a matching exception handler associated with the parent of the step that throws the exception (represented as a step attached to an X on the right of the step bar). Of particular interest and importance is the Little-JIL facility for specifying how execution proceeds after completion of the exception handler; for example, execution may return to the step that triggered the exception or it may continue with the step that handled the exception. Although it is possible for DFGs to represent how exceptions are to be handled and how execution is to resume subsequently, such DFGs quickly become quite complex and impenetrable, especially when exceptions can originate in multiple ways and from multiple process locations. Little-JIL supports these specifications in a particularly clear and intuitive way.

3.3.5. *Scoping*

A parent step and all of its descendants represent a scope, enabling the specification that certain datasets are to be treated as specific to that scope. Little-JIL also supports recursive specifications of a step within its own scope, greatly clarifying the iterative application of a process step to specifically defined datasets.

The PDG for the water flux system illustrates the power of this approach (Fig. 5). The requisite root process (Process Data) has three parallel sub-steps that represent the three subsystems (real-time, post-processing, alternate; Fig. 5a). Each of these steps has a series of sequential sub-steps that retrieve and process a datum; the Process Alt Datum step in the alternate measurement subsystem has an additional iterative sub-step (Process Next Datum) to ensure that models and modeled values from the appropriate temporal window are duly updated. The real-time subsystem (Process RT Datum) has two sequential sub-steps that are expanded in the next two figures (Fig. 5b and c). The Get RT Datum step retrieves real-time data from the three measurement stations, performs range checking, and selects among the two precipitation measurements. The Process Datum step retrieves the

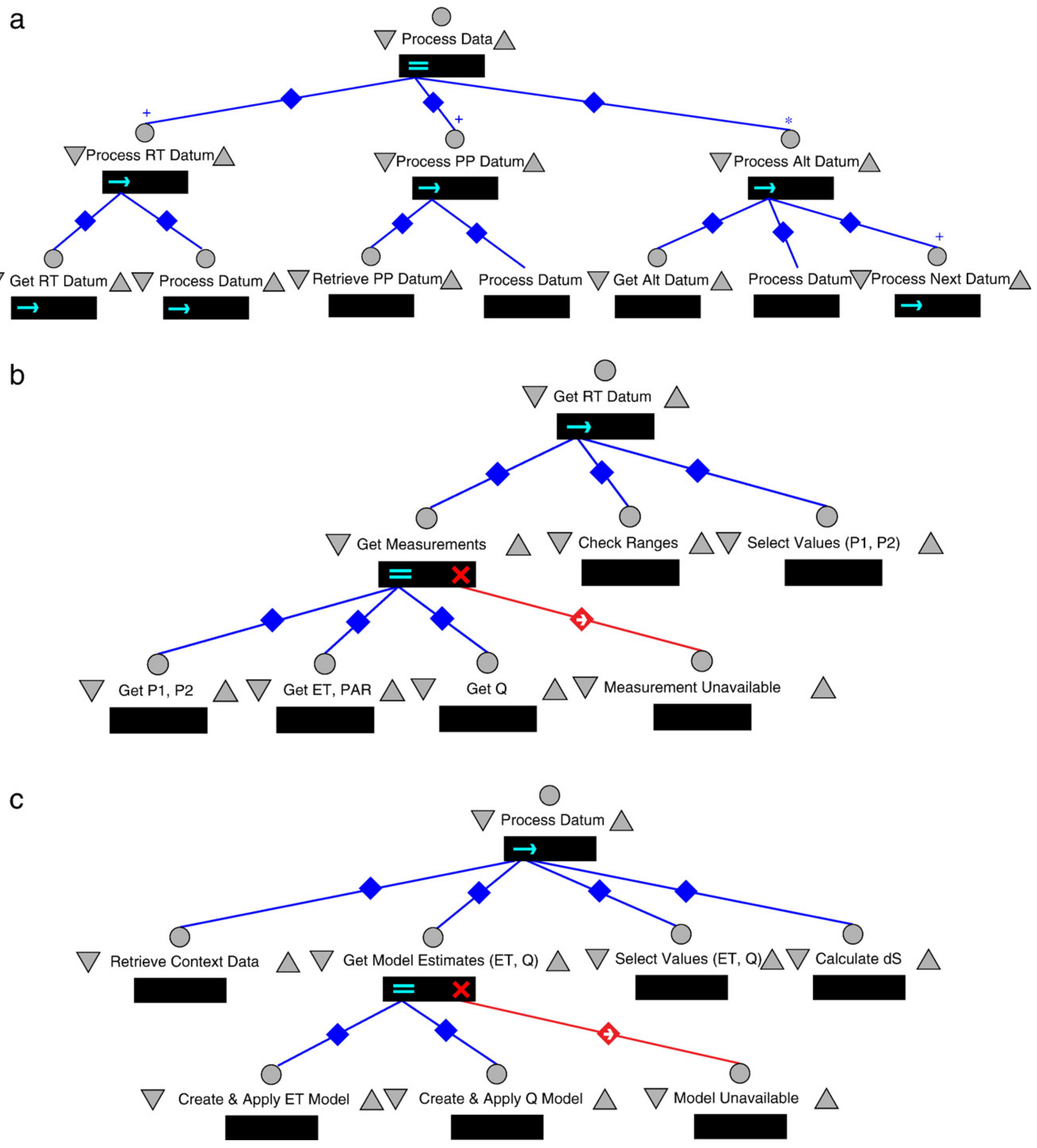

Fig. 5 – Simplified excerpts from the process-derivation graph (in Little-JIL) for the water flux system, showing (a) the root process and major sub-steps, (b) the process for real-time data collection and quality control, and (c) the process for creation and application of models and calculation of the water budget.

appropriate context data (a 30-day or 60-day window for real-time and post-processing, respectively), gets the model estimates, selects among measured and modeled values, and calculates the change in water storage (*dS*). Creation and application of the two models (for *ET* and *Q*) are parallel processes. Note that the Process Datum step is utilized by each of the three major subsystems but only has to be defined once.

It is important that the water flux system be robust and able to react properly to a wide range of contingencies. The PDG provides an effective and efficient means for ensuring robustness through the use of exceptions and exception handlers. For example, in Fig. 5b an exception handler (Measurement Unavailable) provides instructions for what to do if a real-time measurement is not available (e.g., because of a communications timeout), whereas in Fig. 5c an exception handler (Model Unavailable) provides instructions for what to do if a model cannot be generated (e.g., because of a significant gap in the data required to construct the model). In both cases the PDG makes the way in which processing is to be resumed clear.

The detail captured in the PDG also allows us to apply rigorous testing and analysis tools. Bounding the number of iterations to consider, it is possible to determine the set of distinct paths that could be executed based on a PDG and then to select test cases that would exercise each of these. If there is parallelism in the PDG, then the number of paths can be extremely large. In such situations, it is useful to apply static analysis techniques, such as the FLAVERS tool mentioned above, to verify that expected sequences of events always (or never) occur. For example, it might be essential that corrective action be taken whenever the precipitation measured by the two rain gauges varies by more than a specified amount. If FLAVERS analysis reveals there is a path where this is not the case, this path would be displayed so that the process description could be corrected. As processes become large, and especially when parallelism and exception handling are included, it is particularly important to apply such techniques to validate that the process definition adheres to required behaviors.

The complete PDG for the water flux system is considerably larger (nearly all of the leaf steps shown in Fig. 5 would be decomposed into sub-steps) and contains significantly more annotations (e.g., artifact and requisite details). Nevertheless, even this abbreviated version illustrates how the questions raised above (in connection with the DFG for the water flux system) can be answered through the use of specification details made available in the PDG.

3.4. *Process metadata*

The three graphs of an analytic web provide complementary views of a scientific process. The DFG is not sufficient (either by itself or with the DDG) to ensure dataset reliability in all but the simplest cases. Nevertheless we have retained it as a component of the analytic web because of its familiarity to a broader audience and because it provides a useful high-level view of a scientific process. Just as software designers often start with a high-level design to gain some understanding of a problem and then subsequently develop a detailed design, so we expect that scientists may first outline a scientific process using a DFG and then elaborate the details of that process using a PDG. Analysis techniques could even be developed to determine if the PDG is consistent with the corresponding DFG and to report any discrepancies that were found in order to help detect misunderstandings in either representation.

The DDG, by tracking what did happen in a particular execution of a scientific process, contains the instance information needed to reproduce that particular trace through the process and to evaluate it for soundness. The PDG, on the other hand, contains the type information needed to reconstruct the overall process and to understand the choices made in a particular execution. Moreover, by specifying everything that could happen during execution, the PDG supports evaluation of all possible traces (and thus of the entire process) for soundness.

Dataset reliability is ensured by either (1) the PDG plus the DDG, because the DDG provides a complete derivation history and the PDG provides all the information needed to reconstruct and analyze the processes utilized; or (2) the PDG plus the input dataset instances, because together they can be used to re-derive all intermediate and final datasets (in effect, to recreate the DDG). The first strategy provides a complete record of all artifacts but usually requires more storage space, while the second strategy saves space but requires more processing time (and access to the original tools and computing environment).

The same tradeoff of space versus time can be seen in a common and related problem: how to ensure access to earlier versions of a dataset so that earlier analyses can be verified. One solution, which works well for traditional ecological studies, is to permanently archive snapshot versions of the dataset as it evolves over time, using nonproprietary formats and unambiguous identifiers (Jones et al., 2006). However this approach is problematic for streaming data and virtually untenable for systems like the water flux system where the data not only are streaming but also are subject to constant revision. In this case it would be far more efficient to archive the PDG and all input data instances (real-time measurements, alternate measurements, selection criteria, etc.), since together these can be used to recreate on demand the current best dataset at any point in time.

Two complementary approaches have been identified for establishing dataset provenance (Braun et al., 2006). In a “disclosed-provenance system” there is a defined process that describes how data items and datasets are generated. In an “observed-provenance system” data items and datasets are logged and monitored as they are created. The analytic web combines the strengths of both approaches: the PDG provides an *a priori* process definition, while the DDG is created by recording what happens as the process is executed. We believe that constructing the DDG directly from execution of a sufficiently articulate process definition is inherently simpler than reconstructing what happened through queries of a structured database of artifacts (as proposed, e.g., in Bowers et al., 2006).

The DDG and PDG play complementary roles in supporting the critical goals of reproducibility and repeatability in science. A scientific study is reproducible if one is able to get the same results from the same input data; it is repeatable if one is able to get comparable results after repeating the experiment with new measurements (Cassey and Blackburn, 2006). The DDG supports reproducibility by ensuring that the original data analysis can be replicated exactly. The PDG supports repeatability by ensuring that the same kind of analysis can be applied to new input data.

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. Selecting the appropriate level of granularity in the PDG is also a challenge and practical guidelines may be required to guide users. While the PDG supports any level of granularity and greater detail tends to improve dataset reliability, excessive detail is counterproductive. Nevertheless we are optimistic that standards can be developed for the XML representation of the three graphs that comprise an analytic web and for unique identifiers for dataset and process instances, and that software tools can be developed that will greatly simplify the task of generating, managing, and interpreting process metadata.

4. Discussion

In the long run we expect that scientific process definition tools will be universally accepted as a basic requirement for scientific data analysis. Eventually such tools will be unobtrusive and will be able to record all relevant details as the scientist works. In the short run we predict that scientists will increasingly choose to use such tools as a means to create, test, execute, and document their analyses. Though there has been great progress in tool development in recent years, we think that current efforts could be strengthened by the inclusion of concepts and methods from analytic webs. In particular we believe that enhancing existing tools to include DDG and PDG capabilities would provide the following critical advantages:

(1) Creation of a PDG requires scientists to think carefully about how they perform a given data analysis. It is our experience that such analyses are often carried out on an intuitive level, without clear articulation (even in the mind of the scientist).
(2) The PDG allows a scientist to create representations that are flexible and scalable. Individual process steps can be kept relatively simple and the overall design can be quickly modified by changing the PDG.
(3) The PDG supports execution of a scientific process and re-execution using different input data and/or different tools, often leading to a significant savings in time and effort.
(4) The PDG provides a useful and efficient framework for structuring and generating the DDG.
(5) Scientific processes represented by PDGs can be rigorously evaluated for undesirable outcomes, logical and statistical errors, and propagation of measurement errors. This is a significant advantage, especially for complex systems.
(6) The PDG provides an alternative to retaining versioned datasets, which may be impractical for streaming data and for other situations where datasets are subject to frequent change.
(7) The combination of the PDG and the DDG (or the PDG and the input dataset instances) provides a basis for evaluating dataset reliability regardless of the complexity of the scientific process.

We believe these advantages will prove to be increasingly important in the future, especially for real-time sensor networks in emerging environmental observatories such as NEON. There are significant practical problems that remain to be addressed, including (1) developing XML (or similar) standards for representing the graphs, (2) devising efficient methods for storing and managing process metadata, (3) developing guidelines for optimal granularity, (4) creating scientific process definition tools that are easy to use and reliable, (5) training scientists in the analytical thinking required to create a PDG, and (6) promoting the use of such tools and their benefits to the scientific community. But the potential benefits to science are great.

Acknowledgments

This work was supported by NSF grant CCR-0205575 and is a contribution from the Harvard Forest Long-Term Ecological Research (LTER) program. We thank the editor and two anonymous reviewers for critical comments that greatly improved the clarity of the manuscript.

REFERENCES

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludäscher, B., Mock, S., 2004. Kepler: an extensible system for design and execution of scientific workflows. 16th International Conference on Scientific and Statistical Database Management, 21–23 June 2004, Santorini Island, Greece.

Atkins, D.E., Droegemeier, K.K., Feldman, S., Garcia-Molina, H., Klein, M.L., Messerschmitt, D.G., Messina, P., Ostriker, J.P., Wright, M.H., 2003. Revolutionizing science and engineering through cyberinfrastructure: report of the National Science Foundation blue-ribbon advisory panel on cyberinfrastructure. (http://www.nsf.gov/cise/sci/reports/atkins.pdf).

Bandinelli, S., Fuggetta, A., Ghezzi, C., 1993. Process model evolution in the SPADE Environment. IEEE Transactions on Software Engineering 19 (12).

Ben-Shaul, I.Z., Kaiser, G., 1994. A paradigm for decentralized process modeling and its realization in the Oz environment. 16th Intl. Conference on Software Engineering, pp. 179–188.

Bowers, S., McPhillips, T., Ludascher, B., Cohen, S., Davidson, S., 2006. A model for user-oriented data provenance in pipelined scientific workflows. International Provenance and Annotation Workshop, IPAW 2006. Springer, Chicago, IL., pp. 133–147.

Braun, U., Garfinkel, S., Holland, D., Muniswamy-Reddy, K., Seltzer, M., 2006. Issues in automatic provenance collection. International Provenance and Annotation Workshop, IPAW 2006. Springer, Chicago, IL., pp. 171–183.

Cacho, O.J., Hean, R.L., Wise, R.M., 2003. Carbon-accounting methods and reforestation incentives. Australian Journal of Agricultural and Resource Economics 47, 153–179.

Cassey, P., Blackburn, T.M., 2006. Reproducibility and repeatability in ecology. BioScience 56, 958–959.

Cayan, D., VanScoy, M., Dettinger, M., Helly, J., 2003. The wireless watershed in Santa Margarita Ecological Reserve. Southwest Hydrology 2, 18–19.

Clark, J.S., Carpenter, S.R., Barber, M., Collins, S., Dobson, A., Foley, J.A., Lodge, D.M., Pascual, M., Pielke Jr., R., Pizer, W., Pringle, C., Reid, W.V., Rose, K.A., Sala, O., Schlesinger, W.H., Wall, D.H., Wear, D., 2001. Ecological forecasts: an emerging imperative. Science 293, 657–660.

Diamant, J., Davidson, H., Thunquest, G., 1990. Process modeling in HP SoftBench. 6th International Software Process Workshop. IEEE Computer Society, pp. 83–85.

Dingman, S.L., 2002. Physical hydrology, 2nd ed. Prentice Hall, New Jersey.

Dwyer, M.B., Clarke, L.A., Cobleigh, J.M., 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., 1996. An introduction to Bayesian inference for ecological research and environmental decision-making. Ecological Applications 6, 1036–1046.

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., Schultz, H., 2006. Analytic webs support the synthesis of ecological datasets. Ecology 87, 1345–1358.

Estrin, D., Michener, W., Bonito, G., 2003. Environmental cyberinfrastructure needs for distributed sensor networks: a report from a National Science Foundation sponsored workshop, 12–14 August 2003. Scripts Institute of Oceanography.

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

Horn, H.S., Shugart, H.H., Urban, D.L., 1989. Simulators as models of forest dynamics. In: Roughgarden, J., May, R.M., Levin, S.A. (Eds.), Perspectives in Ecological Theory. Princeton University Press, New Jersey, pp. 256–267.

Jones, M.B., Schildhauer, M.P., Reichman, O.J., Bowers, S., 2006. The new bioinformatics: integrating ecological data from the gene to the biosphere. Annual Review of Ecology, Evolution, and Systematics, 37, pp. 519–544.

Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger-Frank, E., Jones, M., Lee, E., Tao, J., Zhao, Y., 2005. Scientific workflow management and the Kepler system. Concurrency and Computation: Practice and Experience 18, 1039–1065.

Michener, W.K., Brunt, J.W., Helly, J.J., Kirchner, T.B., Stafford, S.G., 1997. Nongeospatial metadata for the ecological sciences. Ecological Applications 7, 330–342.

NRC (National Research Council), 2003. Sharing publication-related data and materials: responsibilities of authorship in the life sciences. National Academy Press, Washington, D.C.

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

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

Peterson, G.D., Carpenter, S.R., Brock, W.A., 2003. Uncertainty and the management of multistate ecosystems: an apparently rational route to collapse. Ecology 84, 1403–1411.

Phillips, N.G., Oren, R., Licata, J., Linder, S., 2004. Time series diagnosis of tree hydraulic characteristics. Tree Physiology 24, 879–890.

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

Schwab, M., Karrenback, N., Claerbout, J., 2000. Making scientific computations reproducible. Computing in Science and Engineering 2, 61–67.

Sutton Jr., S.M., Osterweil, L.J., 1997. The Design of a Next Generation Process Language, Fifth ACM SIGSOFT Symp. on the Foundations of SW Eng. Springer-Verlag, Zurich, Switzerland, pp. 142–158.

Suzuki, M., Katayama, T., 1991. Meta-operations in the process model HFSP for the dynamics and flexibility of software processes. First International Conference on the Software Process. IEEE Computer Society Press, Redondo Beach, CA, pp. 202–217.

Thornton, P.E., Cook, R.B., Braswell, B.H., Law, B.E., Post, W.M., Shugart, H.H., Rhyne, B.T., Hook, L.A., 2005. Archiving numerical models of biogeochemical dynamics. Eos 86, 431.

Wilson, K.B., Hanson, P.J., Baldocchi, D.D., 2000. Factors controlling evaporation and energy partitioning beneath a deciduous forest over an annual cycle. Agricultural and Forest Meteorology 102, 83–103.

Wilson, K.B., Hanson, P.J., Mulholland, P.J., Baldocchi, D.D., Wullschleger, S.E., 2001. A comparison of methods for determining evapotranspiration and its components: sap-flow, soil water budget, eddy covariance and catchment water balance. Agricultural and Forest Meteorology 106, 153–168.

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., Sutton Jr., S.M., 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).