Smart Checklists for Human-Intensive Medical Systems

George S. Avrunin, Lori A. Clarke, Leon J. Osterweil
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA USA
Email: {avrunin,clarke,ljo}@cs.umass.edu

Julian M. Goldman
Massachusetts General Hospital
Boston, MA USA
Email: jmgoldman@partners.org

Tracy Rausch
DocBox, Inc.
Waltham, MA USA
Email: tracy@docbox.com

***Abstract*—Human-intensive cyber-physical systems involve software applications and hardware devices, but also depend upon the expertise of human participants to achieve their goal. In this paper. we describe a project we have started to improve the effectiveness of such systems by providing Smart Checklists to support and guide human participants in carrying out their tasks, including their interactions with the devices and software applications.**

***Keywords*-Human-intensive systems, Checklists, Process Guidance, Process Modeling, Process Analysis**

I. Introduction

This paper describes an approach for increasing the effectiveness of human-intensive cyber-physical systems, systems involving people, devices, and software applications in which the participation and expertise of humans play a central role in achieving success. Such systems are extremely difficult to understand, develop, and maintain, as they add the complexity and variability of human participation to the already complex interactions among the hardware devices and software applications. To address these difficulties, we are building on our previous work on formalizing and analyzing medical processes [1] to explore how the use of smart, context-aware, dynamic checklists can help to make humans more effective and improve outcomes in human-intensive health care systems. The approaches and technologies we plan to explore and develop will also be applicable to other human-intensive cyber-physical systems such as air traffic control and safety-critical plant management.

A central feature of our approach is the use of precisely defined, carefully analyzed process models to support real-time guidance in the form of checklists that change and adapt as process execution events modify the contexts in which humans perform their tasks. The use of checklists to support human participation in processes in domains such as aviation and space has been well established for decades. More recently, checklists have also been used to support human participation in medical processes (e.g., [2], [3]). Usage in the medical domain, however, has highlighted a variety of shortcomings (e.g., [4], [5]) that we will address in this project.

We believe that *Smart Checklists* can address these shortcomings and be key contributors to a vision of much more effective human participation in human-intensive cyber-physical systems in general, and in health care systems in particular. We are exploring this hypothesis by investigating, developing, and evaluating process monitoring and analysis technologies and an architectural framework for integrating them to support the implementation of a Smart Checklist System. This system will communicate process monitoring and guidance information to human performers while requiring performers to take few, if any, additional steps. The system will draw upon both analyses of the process model and process execution monitoring to communicate to process performers historical context information and future execution projections to help them make better-informed decisions. The system will use context information to adjust lists of pending actions dynamically to reflect current circumstances, eliminating unnecessary steps and highlighting approaching deadlines. Importantly, Smart Checklist guidance will not assume normative sequential process execution, which is rarely what happens in real-world human-intensive cyber-physical systems. The system will use process models that incorporate exception management and concurrency specifications to guide process performers in identifying and responding to exceptional or hectic circumstances, which have been shown to be major sources of errors and confusion. Smart Checklists will also be useful when execution deviates from the specified process. Although following well-established procedures can reduce errors (e.g., [6]), circumstances sometimes warrant deviating even from previously established exception handling procedures. In such situations, the Smart Checklist System will continue to gather context information, but no longer (aggressively) interject guidance until and unless the human performer accepts a proposed plan for rejoining the specified procedure. Behind the scenes, however, analyses would continue to use process model and context information to anticipate and warn of impending hazard violations.

Smart Checklists will also help assure that deadlines are met. For example, in a hospital, blood must be transfused shortly after it leaves the blood bank. Such a real-time constraint will be specified in our process model, and our analyzers will use it to statically determine potential timing constraint violations and to optimize on-line timing moni-

978-1-4673-2266-9/12/$31.00 ©2012 IEEE

toring. Hopefully, warnings will be provided early enough to assure that deadlines are met.

Our Smart Checklist system will also automatically derive [7] a safety envelope [8] to be dynamically monitored when it is not possible to prove definitely that activities are compatible with process safety requirements. Such monitoring will complement monitoring for deviation from the specified process, property monitoring, and real-time monitoring, providing added assurance that process executions are adhering to their requirements. Moreover, the information gathered from all these types of monitoring will also be used to create retrospective profiles of process executions, perhaps leading to new insights for process improvement.

To see how Smart Checklists could be applied in the health care domain, consider a patient who is undergoing coronary artery bypass graft (CABG) surgery in the operating room (OR), and who will subsequently be moved to the intensive care unit (ICU). Anticipating this move, ICU personnel (nurses, technicians, respiratory therapists, assistants) will have to prepare appropriate equipment (e.g. multiple intravenous medication infusion pumps, multiple blood pressure monitors, lung ventilator), supplies, and medications. While the patient is undergoing surgery, a decision support system will be accessing the hospital network for context information such as medication infusions, allergies, lung ventilator settings, and atypical medical devices/therapies being used. This information will comprise a key part of the context provided through Smart Checklists to guide ICU staff to prepare medication for the patient, set or recheck auto-set medical device settings, and identify missing steps, data, or devices.

To achieve this vision we are developing:

- *Systems Architectures and Infrastructures* to support collaboration among system components and process performers;
- *Process Monitoring, Deviation Detection and Explanation, and Recovery* capabilities to warn process performers when a process execution has departed from the expected procedures, to help explain the reasons for such departures, to suggest possible recovery actions, and to propose how performers might rejoin a normative process path;
- *Context Awareness* facilities to capture, synthesize, and present relevant process execution state and history information and to project future consequences (e.g., resource utilization contention) of alternative decisions;
- *Profile-Based and Timing-Based Analysis* for analyzing process timing constraints, identifying looming deadlines, and using previously-gathered profile information to suggest early actions to increase the likelihood of meeting these deadlines;
- *Safety Envelopes* to determine if system component behaviors are consistent with overall system and process requirements, and to devise restrictive safety envelopes when they are not.

We are optimistic that systems that implement Smart Checklists will gain acceptance because they will reduce workloads such as process documentation (which will be automatically generated) for process performers, reduce errors, and improve outcomes. For instance, in the CABG scenario described above, the actions of the nurses, technicians, respiratory therapists, and assistants will be automatically recorded, annotated with the times, dosages, and device readings. While medical professionals will review and add insightful comments to this documentation, the overall time to create this documentation will be reduced and its accuracy improved, providing a firmer foundation for evidence-based process improvement.

II. Approach

In earlier work (see [1] for summaries of this work and detailed references), we developed and evaluated technologies to support the continuous improvement of health care processes. Specifically we developed a modeling language, called Little-JIL [9], and a suite of analysis tools for evaluating Little-JIL process models. A recent evaluation undertaken by the D'Amour Cancer Center attributes, in large part, a roughly 70% drop in errors to our work on representing and analyzing chemotherapy processes [6].

We are now building SmartCheck, a prototype system that will build on this earlier work to create and manage smart, context-aware, dynamic checklists. These checklists will include lists of impending tasks to be performed and conditions to be checked. The checklists will also incorporate context information, where our notion of context is quite broad, including the history of the process execution, summaries of past executions, and projections of possible future execution of the current process. Some checklist items will be inferred statically using analyses of process models to determine which steps are needed to ensure that a process execution satisfies stated correctness and safety requirements. But some checklist items will change dynamically as execution proceeds, driven by changes to context caused by actions of other performers, new information generated by analysis components, and other system execution events. The components we are building (e.g. our context generators and analyzers), embedded hardware devices, software applications, and process performers will all be viewed as system components that generate events and both produce and consume context information. Appropriate summaries of these events, combined with historical, prospective and current process execution information, will then be synthesized and presented as context information to human performers.

The scenario presented in Section I illustrates the use of a system like SmartCheck, suggesting that up-to-the-minute patient information and individual OR actions could be communicated as appropriate to OR personnel and to ICU

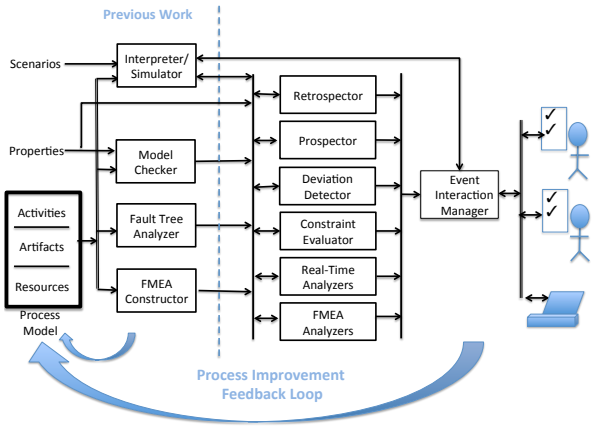

Figure 1. Conceptual Architecture

personnel. As these contexts change the Smart Checklists will be modified in response, to assure that OR and ICU personnel and devices receive suitably modified guidance.

Figure 1 shows the abstract conceptual architecture of this system. Components on the left were developed in our earlier work and support formalizing and analyzing medical processes, with the smaller arrow at the bottom of the figure indicating that offline static analysis is used to improve the process whose execution is being aided by SmartCheck. The larger arrow at the bottom of the figure indicates that results of previous process executions will continually update summaries to provide increasingly comprehensive historical context information. Components on the right will both generate and react to events, some of which will be picked up automatically from hardware devices and from querying of software systems (such as EHRs). Other events will be originated by process performers and analysis tools executing at runtime. Analysis tools will support each other by exchanging their results in ways indicated below. The outputs of these components will, as shown, be funneled through an Event Interaction Manager that will ensure the right tasks and context information are communicated to the right performer checklists in a clear and timely way.

The key components of our expanded vision of this system are: the **Retrospector**, which will provide information about process execution history; the **Prospector**, which will provide possible future process execution information such as tasks to be performed, resources to be required, and upcoming decisions to be made; the **Deviation Detector**, which will monitor event streams to determine when process execution has deviated from the process model, rank the most likely causes of deviations, suggest what might have been done incorrectly, and indicate possible ways to recover; the **Constraint Evaluator**, which will notify human performers when process execution has, or will imminently, violate a specified constraint; the **Real-Time Analyzers**, which will keep performers apprised of looming deadlines, increasing the urgency of warnings as deadlines approach; the **Profile-Based Analyzers**, which will use accumulated historical data to determine probabilities of unwanted events and suggest ways to avoid them; and the **Event Interaction Manager** (EIM), which will manage the flow of events and information between the analysis components and process performers to assure that the right events and information are delivered to the right performers [10].

III. Research Challenges

In this section we describe a few of the many research challenges we anticipate in developing the SmartCheck prototype.

A. System Architecture and Infrastructure

SmartCheck will support the activities of agents, ranging from doctors and nurses to infusion pumps and medical record systems, who are both providers and consumers of this information. The components of SmartCheck itself will both use and create information needed by other components and by those process agents. Different instances and different types of agents will require and provide different kinds of information at different times. The necessity to create, share, and use large amounts of diverse and complex information about the execution of the process, its history and possible future paths, is a key challenge of human-intensive systems, and is being studied as a key activity of this project, using the example of complex health care processes such as the OR-ICU handoff process described above. We now focus on two particularly central aspects of this research.

Specification of state information requirements: Our Little-JIL process language interpreter manages a lot of current state information, but, as noted in Section II, the context information that SmartCheck will provide will be much more comprehensive. For example, historical information will be an important component of the context information needed especially by decision-makers such as doctors. Analysis of actual process models will help us to understand the needs for these different kinds of information, thereby offering important insights into desirable strategies for meeting these needs. Ultimately this research will suggest desiderata for languages to be used as the basis for specifying the integration of components to implement human-intensive cyber-physical systems.

The SmartCheck Architecture: While Figure 1 presents an abstract view of the system that we are building, it leaves unanswered many questions about how this will be implemented as a concrete system architecture. For example, the various analysis components that we are developing will be both consumers and producers of context information, and the various agents in the system will also be both consumers and producers. Various offline static analyzers will be producers of information used to augment process context, but new process executions will also create data of

value as input for augmenting the historical archive. All of this raises questions about where various kinds of context information will reside, what entities will collect it and how they will maintain it.

We are building upon existing technologies, namely the Janus message-passing system, developed at the University of Massachusetts, and the DocBox technology, developed by DocBox, Inc. and based upon years of collaborative work by the Medical Device Plug-and-Play (MD PnP) Interoperability program. Janus translates agent activities into events that are recognizable as Little-JIL process events (and vice versa), using an event translation table that will be dynamically modifiable so that the information exchanged will vary over time as needed. Janus has been used primarily for supporting communication with human agents, but in SmartCheck it will also communicate with devices and hospital software systems through the DocBox platform. DocBox's platform utilizes the Object Management Group's Data Distribution Service specification as a messaging service. This, paired with a JBoss Drools multi-event processing engine, creates a link between the process performers, devices at the patient's bedside, and the hospital network that we will use to communicate process performers' actions to the EIM, which will then pass them on to the SmartCheck components via Janus and the event translation specification. The DocBox platform is compliant with the ASTM F2761-2009 Device Integrated Clinical Environment Standard for safely engineering medical device integration for patient-centric care and the platform's nomenclature and information model is based on the IEEE 11073-10102 standard for point-of-care medical device communication and SNOMED CT. Initially, we will use the specification of the DocBox platform as an interface for SmartCheck's communication with the clinical environment, but, eventually, SmartCheck may run as an application on the DocBox platform. Our experience with these key components will help define requirements for the information that devices will need to provide to support implementation of the Smart Checklist vision.

B. Process Monitoring, Deviation Detection and Explanation, and Recovery

SmartCheck will attempt to determine whether a sequence of events coming from a process execution is consistent with at least one trace through a process model and, if not, attempt to explain why and attempt to suggest remedial action. Essentially, SmartCheck will compare the incoming sequence of events with *all* possible executions described by the process model, and determine whether this sequence occurs on a prefix of any possible execution. SmartCheck will do this by defining the sequence of events to be a property, and then using model checking to determine whether any execution satisfying the process model also satisfies the property. If not, SmartCheck will report a deviation and will also provide indications of how and why the deviation might have occurred and will attempt to suggest how to recover from the deviation. But finding good explanations is not straightforward

For example, suppose a process model specifies two possible step sequences, a, b, c, d, g, h, and a, c, d, e, f, h, and the event sequence a, c, d, g has been observed. Either step b has been omitted from the first path, or step g has been incorrectly performed after step d in the second, but we cannot definitively determine which is the case. Making matters worse, even detection of a deviation (such as the omission of b) can require observing an arbitrarily large number of steps after the deviation has occurred in cases where the process paths are sufficiently similar.

We are exploring the use of string matching to address this problem. We select a set of *edit operations* (e.g. insertion and deletion) to modify a sequence and assign a cost to each operation. We then use a search procedure (e.g., a standard dynamic programming algorithm) to identify low-cost sequences of edit operations that convert event sequences satisfying the process model into the observed sequence. Each sequence of edit operations then provides a possible *explanation* of how the observed execution deviated from the process model. The costs of these explanations are used to rank them and suggest which to show to the process performers. We are also exploring whether more accurate explanations might be derived from treating complicated edits as single operations with lower cost than equivalent sequences of insertions and deletions (e.g., [11], [12]).

We will also explore providing performers with guidance about corrective actions that they might take. Although there are many difficulties in doing this well, the safety literature (e.g., [11]) makes it clear that very serious problems often result from misguided attempts to recover from erroneous situations.

In health care in particular, we do not expect to find easy algorithms for guiding performers back to safe conditions after a deviation has occurred. But we are asking domain experts to identify deviations that are especially common and to construct appropriate repertoires of recovery actions for them. We are also asking domain experts to select the constraints they consider especially critical during recovery from deviation. These constraints are being chosen from those defining the safety envelope, those used in real-time analysis, and those used in model checking. The Constraint Evaluator component shown in Figure 1 will monitor actions taken during recovery to check that violations of these constraints are not imminent.

C. Context Awareness

SmartCheck will provide process performers with information they can use to build confidence in the appropriateness of requests to perform a tasks. This information is likely to be of special value in scenarios like the OR-ICU

handoff, where some process performers enter the process with little knowledge of the history of the particular process execution. The answers to queries will be derived from process execution context information that SmartCheck will gather and maintain.

A process defined in a language such as Little-JIL is the synthesis of information about activities, resources, and artifacts. The SmartCheck Prototype will maintain these, but we are developing new technologies to gather additional kinds of information such as what steps, performed by which entities, used which inputs in what ways to derive which outputs.

Initially SmartCheck will use Data Derivation Graphs (DDGs) [13], [14] to manage such process information. A DDG is a DAG-like structure that records the steps performed, what entity performed them, and (usually pointers to) the artifacts used as inputs and outputs. The history of how an artifact was developed is provided as a trace through the DDG, and complements the current values of artifacts. Query languages and summarizers are being developed to extract needed information and represent the results. Some summarization may be done proactively. Controlling the size of a DDG will be a challenge, as DDGs become very large very quickly, especially if they record fine-scale details of the execution of long-running processes. We are studying such questions as which details are worth capturing and recording, and whether automated or semi-automated techniques might be used to change the kinds of details recorded. We will also address the problem of expediting the extraction of information that is to be presented, as well as historical information and summaries of that information.

SmartCheck will provide information about expected future paths through the process, including information about the consequences of decisions that a process performer may be about to make. The Prospector will use algorithms from model checking to trace paths through the Little-JIL activity diagram forward from the step(s) currently being executed. While tracing paths forward, the Prospector will accumulate information about resource utilization levels, potential resource contention, estimated elapsed time, and projections of what other process performers will be doing at what future times. The Prospector will rely upon appropriate estimates of time, cost, resource requirements, and accumulated historical records of past process executions.

D. Profile-based and Timing-based Analysis

We will use process execution monitoring results to accumulate historical information summarizing multiple process executions. This historical information will include, for example, the probabilities of various events occurring in different process contexts. Such information will be used to refine prospective analyses as well as the process models themselves. We use such information, for example, to try to determine when a particular check is wasteful because the circumstances under which it detects errors rarely arise or to suggest when inserting an additional check at a particular point should be expected to catch a large number of errors before they cause harm. Doing this effectively requires knowing the probabilities of all the ways the process could arrive at that point. We are exploring probabilistic model checking techniques for deriving this information.

Real-time constraints are clearly an important part of many health care processes. For instance, in surgery in which the patient is placed on a ventilator, it is sometimes necessary to stop the ventilator before taking an x-ray. It is, of course, crucial that the ventilator be turned back on quickly, and patients have died when exceptional events diverted the attention of the medical staff who then failed to restart the ventilator in time. Soft real-time constraints are also common, as in nursing checks on patient condition or laboratory tests to adjust medication dosages.

Little-JIL currently includes a primitive timer construct, but this is cumbersome to use at best and may not be powerful enough to express some of the important kinds of timing constraints needed for human-intensive systems. We are investigating additional specification techniques for adding real-time information to process models and ways to use this information to efficiently implement appropriate warnings of impending deadlines. We are also exploring static analysis approaches, such as real-time model checking, to detect potential timing vulnerabilities and guide process performers' decisions about meeting upcoming deadlines.

IV. Evaluation

The smart SmartCheck prototype will be used to explore several important research questions such as:

1) Does the architecture of our prototype system adequately support communication and interaction among the prototype components and with the agents (including humans, devices, and software systems) executing the process?
2) How well do our technologies assess and represent current, past, and future context and accumulated historical data? How well do they respond to queries about process context from process performers?
3) What kind of detail in the event stream is required for our system to determine the progress of process execution?
4) How well do our technologies detect process deviations and identify likely causes, and how is this affected by the quality of the event stream?
5) How can profiles of past executions be used to improve process models, monitoring and guidance?

We will explore these questions using simulated event streams gathered from a variety of sources, and will experiment with degrading these event streams by eliminating events, seeding errors, etc., and determining how such changes affect the ability of our tools to derive context

and detect deviations. We will also ask experts on the medical processes to evaluate such things as the quality of the explanations suggested. For instance, we will present these experts with information about an execution of one of the processes and output from SmartCheck at that stage of execution and ask them how they would use the output and whether it is helpful, what other information they would want, what queries they might make for additional context information. The execution scenarios considered will include nominal ones as well as ones in which exceptional conditions have arisen. As the prototype develops, we hope to carry out observations in simulated clinical settings to see whether the guidance offered by SmartCheck can actually improve performance.

V. Concluding Remarks

We believe that the proposed approach has the potential to transform health care. As argued earlier, experience with less flexible and less comprehensive approaches, such as checklists and simplistic process models, have resulted in a significant reduction in errors. We believe that our proposed approach addresses many of the adoption issues associated with these other approaches and provides better support for process performers. Although our proposed approach will not be applicable to all medical processes, there are clearly a large number of medical processes where it might prove applicable. Also, although clinical practices vary from hospital to hospital, there appear to be some core processes that are followed nearly universally. Support for these processes, along with specialization that would allow these processes to be tailored for specific locations, would make an important contribution to improving the performance of these processes. In addition, our process-based approach should be applicable to other human-intensive systems that have serious safety concerns.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Awards CCF-0820198, CCF-0905530 and IIS-0705772. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] G. S. Avrunin, L. A. Clarke, L. J. Osterweil, S. C. Christov, B. Chen, E. A. Henneman, P. L. Henneman, L. Cassells, and W. Mertens, "Experience modeling and analyzing medical processes: UMass/Baystate medical safety project overview," in *1st ACM International Health Informatics Symposium*, Arlington, VA, Nov. 2010, pp. 316–325.

[2] B. M. Hales and P. J. Pronovost, "The checklist: a tool for error management and performance improvement," *J. Crit. Care*, vol. 21, pp. 231–235, 2006.

[3] J. D. Birkmeyer, "Strategies for improving surgical quality—checklists and beyond," *New Engl. J. Med.*, vol. 363, pp. 1963–11 965, 2010.

[4] L. A. Hassell, A. V. Parwani, L. Weiss, M. A. Jones, and J. Ye, "Challenges and opportunities in the adoption of College of American Pathologists checklists in electronic format: perspectives and experience of Reporting Pathology Protocols Project (RPP2) participant laboratories," *Arch. Pathol. Lab. Med.*, vol. 134, no. 8, pp. 1152–1159, 2010.

[5] B. D. Winters, A. P. Gurses, H. Lehmann, J. B. Sexton, C. Rampersad, and P. J. Pronovost, "Clinical review: checklists—translating evidence into practice," *Crit. Care*, vol. 13, no. 6, p. 210, 2009.

[6] W. C. Mertens, S. C. Christov, L. A. Clarke, G. S. Avrunin, L. J. Osterweil, L. J. Cassells, and J. L. Marquard, "Using process elicitation and validation to develop accurate and complete safety-critical medical process descriptions: A case study of chemotherapy ordering and delivery," submitted to *The Joint Commission Journal on Quality and Patient Safety*.

[7] H. M. Conboy, G. S. Avrunin, and L. A. Clarke, "Process-based derivation of requirements for medical devices," in *1st ACM International Health Informatics Symposium*, Arlington, VA, Nov. 2010, pp. 656–665.

[8] A. Yasmeen and E. L. Gunter, "Automated framework for formal operator task analysis," in *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, ser. ISSTA '11. New York, NY, USA: ACM, 2011, pp. 78–88. [Online]. Available: http://doi.acm.org/10.1145/2001420.2001430

[9] A. G. Cass, B. S. Lerner, E. K. McCall, L. J. Osterweil, J. S. M. Sutton, and A. Wise, "Little-JIL/Juliette: A process definition language and interpreter," in *Proceedings of the 22nd International Conference on Software Engineering*, Limerick, Ireland, 2000, pp. 754–757.

[10] T. J. Sliski, M. P. Billmers, L. A. Clarke, and L. J. Osterweil, "An architecture for flexible, evolvable process-driven user guidance environments," in *Joint 8th European Software Engineering Conference (ESEC 2001) and the 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 9)*, Sep. 2001, pp. 33–43.

[11] J. Reason, *Human Error*. Cambridge University Press, 1990.

[12] T. W. van der Schaaf, "Near miss reporting in the chemical process industry: An overview," *Microelectronics and Reliability*, vol. 35, no. 9–10, pp. 1233–1243, 1995.

[13] L. J. Osterweil, L. A. Clarke, A. M. Ellison, E. R. Boose, R. Podorozhny, and A. Wise, "Clear and precise specification of ecological data management processes and dataset provenance," *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 189–195, 2010.

[14] L. J. Osterweil, L. A. Clarke, R. Podorozhny, A. Wise, E. Boose, A. M. Ellison, and J. Hadley, "Experience in using a process language to define scientific workflow and generate dataset provenance," in *CM SIGSOFT 16th International Symposium on Foundations of Software Engineering*, 2008, pp. 319–329.