

Provenance Collection in Reservoir Management Workflow Environments

Fan Sun, Jing Zhao
Department of Computer Science
University of Southern California
Los Angeles, CA 90089
Email: {fansun, zhaoj}@usc.edu

Karthik Gomadam, Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California
Los Angeles, CA 90089
Email: {gomadam, prasanna}@usc.edu

Abstract—There has been a recent push towards applying information technology principles, such as workflows, to bring greater efficiency to reservoir management tasks. These workflows are data intensive in nature, and the data is derived from heterogenous data sources. This has placed an emphasis on the quality and reliability of data that is used in reservoir engineering applications. Data provenance is metadata that pertains to the history of the data and can be used to assess data quality. In this paper, we present an approach for collecting provenance information from application logs in the domain of reservoir engineering. In doing so, we address challenges due to: 1) the lack of a workflow orchestration framework in reservoir engineering and 2) the inability of many reservoir engineering applications to collect provenance information. We present an approach that uses the workflow instances detection algorithm and the Open Provenance Model (OPM) for capturing provenance information from the logs.

Keywords—provenance; workflow; reservoir engineering;

I. INTRODUCTION

Reservoir management (also called subsurface management) predicts the behavior of oil and natural gas within subsurface rock formations, using elements of geology and petroleum engineering. There has been a recent push towards applying information technology principles, such as workflows, to bring greater efficiency to reservoir management tasks [1], [2]. Tasks such as model upscaling, reservoir forecasting, and water injection optimization are now realized by integrating different applications using workflows. These workflows are data intensive in nature (as exemplified in Section II), and the data is derived from heterogenous data sources, such as reservoir simulation models, equipment test data, cumulative production history, and performance curves [3]. This has placed an emphasis on the quality and the reliability of the data that is used in reservoir management workflows.

Data provenance is metadata that pertains to the history of the data and can be used to assess data quality. It can be used to estimate data quality and data reliability based on the source data and transformations, and provide proof statements on data derivation. Provenance information can also be used to trace the audit trail of data, determine the resource usage, and detect errors in data generation. The use of provenance information in determining the data quality

has been demonstrated in domains such as e-Science and GIS [4], [5], [6], [7], [8], where data intensive workflows involving heterogenous data sources are a norm. Given the similarity between workflows in these domains and reservoir management, one can borrow from the approaches for assessing data quality, to apply to reservoir management.

The ability to trace audit trails and estimate the quality and reliability of data could address major challenges in the design, development and maintenance of data intensive workflows. To realize this ability, one must address two important challenges.

- **Lack of a workflow orchestration framework:** In the e-Science domain, workflow orchestration engines such as Taverna [9] can be used to collect provenance information. The lack of a such framework to integrate the different applications, makes it hard to collect and integrate provenance information in reservoir management.
- **Collecting provenance information:** In reservoir management, engineers use software applications for reservoir modeling, simulation, optimization and forecasting, many of which lack the ability to collect provenance information.

The main contribution of this paper is an approach to collect provenance information from log files. Log files generated by applications contain detailed information on user activities and data history, and are thus crucial to understanding the provenance. Information is captured in log files as sequential records of predefined user activities, such as names of invoked commands and data items that are used as parameters. This paper addresses the challenge of collecting provenance information from such low level and fine grained data.

Our approach builds upon our earlier work in semantic rich workflow modeling and workflow instance detection [10]. The semantic rich workflow model maps low level log entries onto high level information such as user intentions. Data flow descriptions defined in the workflow model provide the heuristics for provenance collection. Given such workflow models, we apply WIDA [10] to detect workflow instances from the logs. Within the detected workflow in-

stances, provenance information is then captured according to the open provenance model and data flow constraints. Rather than defining provenance information at the level of each application, we define provenance information at the level of a workflow, allowing us to understand the provenance information from a data flow perspective. This is important as the overall data flow context in which a particular object is created or changed can yield significant clues for identifying its provenance information.

The rest of this paper is organized as follows. We present examples that motivate this research in Section II. Section III presents a short sketch of the semantic rich workflow model that is used to model domain-specific tasks. In Section IV, we discuss the algorithms for workflow instance detection and provenance collection. We sketch the system architecture and implementation details in Section V. Section VI surveys related methodologies and systems. We conclude this paper and discuss our future work in Section VII.

II. MOTIVATING EXAMPLES

In this section, we discuss two examples that illustrate the need for collecting provenance information in reservoir management.

A. Model Upscaling

Reservoir simulation is widely employed in the petroleum industry for the prediction and management of reservoir performance. A primary input to the simulator is the geological description of the reservoir. This description typically is in the form of a high resolution geo-cellular model, containing petrophysical data such as porosity and permeability. These models are built as geostatistical realizations that are constrained by data of different types and scales. The geo-cellular models are often generated at high levels of resolution because fine scale features can significantly impact reservoir performance. However, this high level detail exceeds the capabilities of standard reservoir simulators and procedures are required to coarsen the resolution to scales more suitable for flow computation. These procedures are commonly referred to as *upscaling* techniques.

Different upscaling procedures are appropriate in different situations. The ideal procedure to use on a particular problem depends on the simulation question being addressed, the production mechanism, and the level of detail that can be accommodated in the coarse model. For a problem involving primary production with only oil being produced, the coarse model should correctly capture the effects of near-well heterogeneity as well as the general large scale flow response of the reservoir. For scenarios involving displacement of oil by water or gas, it may be important to accurately capture the effects of key flow paths between injection and production wells. This may require the use of specialized gridding procedures. Design of upscaling procedures is still commonly done by hand nowadays.

It is important to replicate the important features of the fine scale model in the upscaled model. Some of these features include total injection or production rate, average pressure or saturation throughout the reservoir, and breakthrough times of injected fluids. Others of interest include the degree of coarsening achievable by a given method, the level of robustness of the coarse scale model (i.e., its applicability to models with different global boundary conditions or well locations), and if the method introduces modifications to the form of the governing equations. To obtain this information requires the comparison of the original geological model with the upscaled simulation model, and reviewing the input metrics of the upscaling procedure. Capturing the information related to the original model and the input metrics as a part of the provenance information can automate this comparison.

B. Reservoir Forecasting

Reservoir forecasting refers to predictions of future reservoir performance at all stages of the reservoir life cycle, and under different scenarios by using reservoir simulation techniques. The goal is to visualize the future performance of the oil field under different operating strategies, and generate the production profiles for the economic evaluation of a project.

The core part of a forecasting workflow is a forecasting simulation application. The simulation application usually has three different kinds of input data: the reservoir deliverability and capacity data, the historical production/injection data, and the surface facility constraints data. The reservoir deliverability and capacity data represent the ability of a reservoir to produce oil. The historical production data is usually collected from real production history. The production should be under the constraints of surface facility capacity, which refers to the facility and export system capacities over the life of the reservoir.

Complex processes are involved in creating the input data. For example, the reservoir deliverability and capacity data, including the well data and block data, can be generated using different methods. Further, each process may involve workflows which employ lab tests, seismic simulators, and production simulators etc. According to different accuracy and timeline requirements, reservoir engineers may adopt a particular approach for generating the reservoir deliverability and capacity data. Complete surface facility constraints data should consider factors like development strategies, fluid properties, surface equipment, and even market and transportation conditions. All these workflows will be governed by a forecasting workflow, which employs integrated workflows/applications to generate reservoir production forecasting results and achieve overall system optimization.

Reservoir engineers make decisions based on the forecasting results, e.g., the detail development strategies. After making a set of decisions, reservoir engineers compare and

evaluate these decisions. During this comparison and evaluation procedure, reservoir engineers need the ability to trace back the entire reservoir forecasting workflow. Provenance information must be collected from the reservoir forecasting workflow to help reservoir engineers navigate the complex workflow. The distributed nature of this workflow and the heterogeneity in logging schemes create a need for an integrated approach to capture the provenance information.

III. SEMANTIC RICH WORKFLOW MODEL

To capture provenance information within the context of workflows, we first formalize activities in reservoir management using a workflow model. There are two challenges in modeling the domain-specific workflows. First, workflows in reservoir management are usually complex and distributed. Since various legacy software applications are widely used in reservoir management, a workflow usually involves multiple legacy applications. Also, a workflow may involve multiple user roles, such as geologists, geoscientists, and reservoir engineers. The workflow model needs to be able to model the collaboration of multiple applications and multiple agents in the workflows. Second, due to the heterogeneity of legacy applications, there are different representations and granularity of domain elements. For example, the representation of *blocks* differs from ECLIPSE [11] to Petex IPM [12]. The heterogeneity in terms of representation and granularity leads to different log schemas. The workflow model should be flexible enough to integrate the legacy logs.

In this section, we design a semantic rich workflow model to model the workflows in reservoir management domain. The semantic rich workflow model is defined in a hierarchical structure, as shown in Figure 1. It consists of building blocks in three different levels: *workflows*, *intentions* and *realizations*. A workflow is composed of a list of intentions in a sequential pattern. Each intention semantically describes “what to do” as one step in a workflow. Each realization describes “how to implement” an intention in concrete steps within a specific application. An intention may have one or more realizations.

A. The Hierarchical Workflow Structure

The *workflow level* provides an overview of the workflow model. As the root node of a workflow model, the workflow node shows the entire workflow as a single process, and gives no clues as to its internal organization. Instead, it shows the interaction between the workflow and external environment which act as data sources and data sinks. The workflow level also describes metadata about the entire workflow, including the name, author, description, version, etc. Metadata provides semantic information about workflows, which makes them both processable by machines and understandable by humans.

The workflow model is expanded in the *intention level*. We define an *intentions* as one logical component of a

workflow, which conceptually describes one step of a workflow, while keeping the implementation details transparent. An intention is essentially an abstraction of application-specific actions in an application-independent logical step. Each workflow contains a list of intentions that compose the workflow in a sequential pattern. We also define internal data objects of the workflow as application-independent data items in the intention level, and shows the flow of data between the various components of the workflow.

The *realization level* provides implementation details for intentions. A realization describes how an intention is achieved within a legacy application. It consists of a list of *actions* in a specific application, which describes how the corresponding intention is achieved step-by-step. The application-independent data items in the intention level are also mapped to the application-specific data instances in the realization level. In our workflow model, each intention may have one or more possible realizations, while a realization maps to exactly one intention. For instance, suppose we have an intention to “run a reservoir forecasting simulation”. The realization could run either a numerical simulation in ECLIPSE [11], or a material balance simulation such as MBAL in IPM [12]. Both types of realization implement this intention.

The semantic rich workflow model tackles the challenge of heterogeneity of legacy logs. The one-to-many mapping from one intention to multiple realizations allows a conceptual operation to be implemented in different legacy applications.

B. Data Mapping and Data Flow Constraints

The semantic rich workflow model also contains information of data flows. Data flow describes how the data streams flow into and out of processes, thus contains derivation history of data objects. Due to the heterogeneity of legacy logs, the data flow is described in the intention level. Specifically, we specify the inputs and outputs of each intention in the workflow model, and map the corresponding data objects from possible realizations to the intention. A *data mapping* from realization level to intention level is defined as a pair:

$$(realiz_a.data_b, int_c.input/output_d),$$

which denotes that data object b in realization a is the input (or output) d of intention c . Data objects from different legacy applications may be represented in different format. The heterogeneous data objects are linked to the process information by following the data mappings.

We also provide *data flow constraints* to specify derivation relationships of data objects among intentions in the semantic rich workflow model. A data flow constraint describes the relationship between an intention and a data object (shown as the dashed arrows in Figure 1). Formally, a data flow constraint is defined as a pair:

$$(int_a.input/output_b, data_k),$$

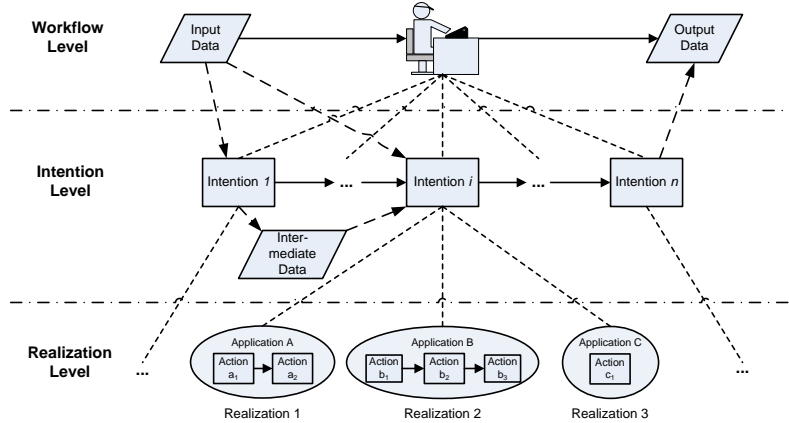


Figure 1. Semantic Rich Workflow Model

which denotes that data object k is the input (or output) b of intention a . The data object could be either one of the inputs/outputs of the entire workflow model, or an intermediate data object generated by another intention. Data flow constraints maintains the consistency of data flow in the semantic rich workflow model. In our work, it is also used to capture provenance between intentions.

IV. PROVENANCE COLLECTION

In this section, we discuss our approach for collecting provenance information from legacy logs. We apply the workflow instance detection algorithm (WIDA) [10] to detect workflow instances, and annotate provenance information within the context of detected instances. The collected provenance information is represented using Open Provenance Model (OPM).

A. The Open Provenance Model

The Open Provenance Model (OPM) [13] is a model for provenance that has been widely used in e-Science. The provenance of data objects are represented by an annotated causality graph, which is a directed acyclic graph, enriched with annotations capturing further information pertaining to execution. Three types of nodes are specified in OPM provenance graphs:

- **Artifacts** represent immutable pieces of states, which may have physical embodiments in physical objects, or digital representations in computer system.
- **Processes** represent actions performed on or caused by artifacts, and resulting in new artifacts.
- **Agents** represent contextual entities acting as catalysts of processes, enabling, facilitating, controlling, or affecting their executions.

B. Workflow Instance Detection

Workflow instances are execution records of workflows. Workflow instances contain runtime information of the

corresponding workflows, e.g. which application has been chosen to implement each intention, the generation and consumption of data objects, the runtime parameters of each process, etc. The goal of workflow instance detection is to identify all instances of a given set of workflows from the log files.

According to the definition of semantic rich workflow model in Section III, the hierarchical structure contains two layers of sequential patterns. The top layer defines a workflow as a sequence of intentions in *sequential workflow pattern* [14]. In the lower layer, each realization consists of a sequence of application specific actions, called a *realization pattern*.

The workflow instance detection algorithm (WIDA) uses Aho-Corasick [15] keyword trees for string matching in both workflow and realization layers. Figure 2 illustrates an overview of the WIDA algorithm. This is realized in the realization and workflow layers in the following manner.

- In the **realization layer**, each log entry is treated as an input symbol. Using each realization pattern as a keyword, we construct the realization pattern trees as keyword trees. The role of the realization pattern trees is searching for a match between the sequence of log entries and the realization patterns. If such a match is found, an instance of the corresponding intention is generated and sent to the workflow layer.
- In the **workflow layer**, a workflow pattern tree is constructed as a keyword tree according to the workflow pattern that is used. The workflow pattern tree detects if a sequence of the populated instances from the realization layer matches any workflow patterns. If a match is found, a workflow instance is generated and populated in the workflow repository.

C. Provenance Annotation

After workflow instance detection, provenance information is annotated within the context of workflow instances.

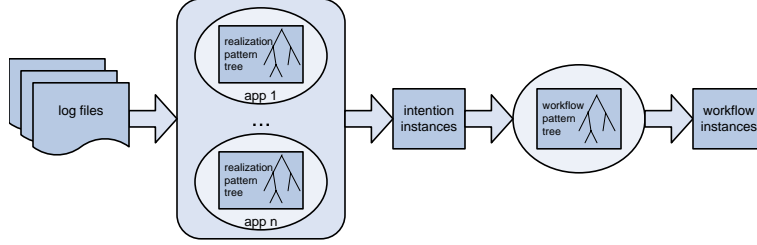


Figure 2. WIDA Overview

According to OPM, we annotate three types of entities in the provenance graph:

- *Agents*: the user information in the workflow instance is captured as agents;
- *Processes*: each intention in the workflow instance is annotated as a process;
- *Artifacts*: each input or output data object of an intention is annotated as an artifact.

We also annotate relationships between the entities in the provenance graph. Examples of annotated relationships include:

- *Execute*: each process is annotated as “executed” by the agent;
- *Trigger*: each process is annotated as “triggered” by the previous process;
- *Consume/generate*: each input/output data is annotated as “consumed”/“generated” by the corresponding process.

The provenance collection process is illustrated in Algorithm 1.

V. IMPLEMENTATIONS

The system employs client/server architecture in a distributed environment. The clients are responsible for parsing the log files and uploading the collected entries to the provenance collection server. The provenance collection server merges parsed log entries, detects workflow instances and annotates provenance information from logs. Figure 3 shows the system architecture.

We classify the output of provenance collection into three categories:

- *Workflow instances*: detected workflow instances are represented in XML format and are stored in a XML database,
- *Provenance information*: the collected provenance information is represented as a graph and is stored in a XML database,
- *Annotated logs*: logs with annotations of provenance information are stored in a relational database.

VI. RELATED WORK

Data provenance has been widely used in the e-Science domain for data analysis. Different aspects of provenance,

Algorithm 1 The Provenance Collection Process

Require: $WI = \{wi\}$: a set of workflow instances,

Ensure: $G = \{E, R\}$: the provenance graph in OPM.

```

1: for each workflow instance  $wi$  in  $WI$  do
2:   add agent node  $A$  to  $G$ ,
3:   for each intention instance  $intention_i$  in  $wi$  do
4:     add process node  $P_i$  to  $G$ ,
5:     add edge “executed” from  $A$  to  $P_i$  in  $G$ ,
6:     add edge “triggered” from  $P_{i-1}$  to  $P_i$  in  $G$ ,
7:     for each input data  $input_j$  of  $intention_i$  do
8:       add artifact node  $T_j$  to  $G$ ,
9:       add edge “consumed” from  $P_i$  to  $T_j$  in  $G$ ,
10:    end for
11:    for each output data  $output_k$  of  $intention_i$  do
12:      add artifact node  $T_k$  to  $G$ ,
13:      add edge “generated” from  $P_i$  to  $T_k$  in  $G$ ,
14:    end for
15:  end for
16:  for each data flow constraint  $dfcs$  in  $wi$  do
17:    if  $dfcs$  is satisfied then
18:      combine the nodes that is specified in  $dfcs$ 
19:    end if
20:  end for
21: end for
22: output  $G$ .
```

including provenance collection, modeling, representation, storing, and application, have been addressed. Simmhan et al. [16] present a survey of projects on data provenance in e-Science, including myGrid [7], Chimera [8], CMCS [5], ESSW [4],

In the myGrid project [7], workflows are defined by using XSculfi language and are executed using the Taverna workflow engine [9]. The provenance information is automatically logged during the workflow execution. Chimera [8] uses Virtual Data Language (VDL) to define its workflow, and uses Virtual Data Catalog (VDC) to represent the provenance data. In both projects capturing provenance data is tightly coupled with the workflow execution environment. Provenance information is automatically captured during the workflow execution. The research presented in this paper ad-

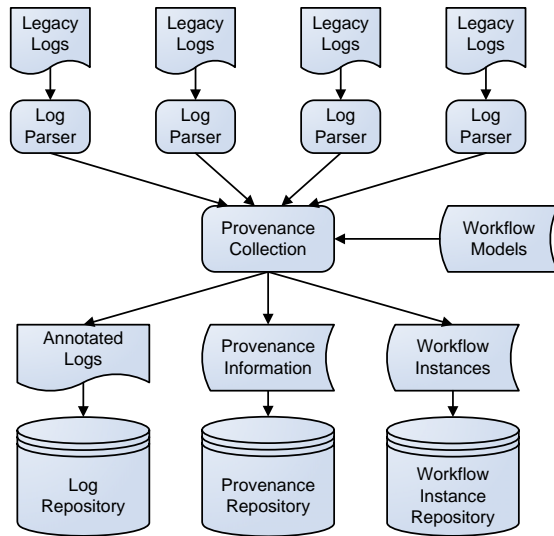


Figure 3. System Architecture

addresses the challenges in collecting provenance information in an environment that neither has an orchestration platform nor has established techniques for collecting provenance information.

VII. CONCLUSIONS AND FUTURE WORK

Reservoir engineering is a data intensive domain, with data quality being a significant metric in determining the overall outcome of domain operations. In this paper, we discussed an approach for capturing provenance information from log files. In doing so, we address the shortcomings of the reservoir engineering environment created due to the lack of an orchestration platform. This is one of the earliest attempts to define a systematic approach for provenance collection in the domain of reservoir engineering. Our continuing research in this area, now aims to address the problem of creating a unified provenance model. This model will describe the basic domain attributes for describing and analyzing provenance information. Further, having such a model would allow us to annotate the provenance data and employ reasoning techniques to draw better conclusions.

ACKNOWLEDGMENTS

This research was funded by CiSoft (Center for Interactive Smart Oilfield Technologies), a Center of Research Excellence and Academic Training and a joint venture between the University of Southern California and Chevron. We are grateful to the management of CiSoft and Chevron for permission to present this work. We also thank Dr. Amol Bakshi of Chevron Corporation for his valuable time and inputs.

REFERENCES

- [1] G. D. Al-Qahtani and et al., "Complex well modeling workflow enabling full field optimization and forward decisions," in *Latin American and Caribbean Petroleum Engineering Conference*, Cartagena de Indias, Colombia, June 2009.
- [2] B. J. Crockett, "The fully integrated workflow: The integration of business processes with people and digital technologies," in *SPE Digital Energy Conference and Exhibition*, Houston, Texas, USA, April 2009.
- [3] R. Soma, A. Bakshi, and V. K. Prasanna, "An architecture of a workflow system for integrate asset management in the smart oil field domain," in *1st IEEE International Workshop on Scientific Workflows (SWF)*.
- [4] R. Bose and J. Frew, "Composing lineage metadata with xml for custom satellite-derived data products," in *16th International Conference on Scientific and Statistical Database Management (SSDBM)*, Greece, June 2004.
- [5] J. Myers, C. Pancerella, C. Lansing, K. Schuchardt, and B. Didier, "Multi-scale science, supporting emerging practice with semantically derived provenance," in *International Semantic Web Conference (ISWC)*, Sanibel Island, Florida, USA, 2003.
- [6] J. Widom, "Trio: A system for integrated management of data, accuracy, and lineage," in *The Biennial Conference on Innovative Data System Research (CIDR)*, Asilomar, California, USA, January 2005.
- [7] J. Zhao, C. A. Goble, R. Stevens, and S. Bechhofer, "Semantically linking and browsing provenance logs for e-science," in *Proceedings of International Conference on Semantics of a Networked World (ICSNW)*, 2004.
- [8] Y. Zhao, M. Wilde, and I. Foster, "Applying the virtual data provenance model," in *1st International Provenance and Annotation Workshop (IPAW)*, Chicago, USA, May 2006.
- [9] K. Belhajjame, K. Wolstencroft, O. Corcho, T. Oinn, F. Tanoh, A. William, and C. Goble, "Metadata management in the taverna workflow system," in *Proc. of CCGRID*, 2008, pp. 651–656.
- [10] F. Sun, V. K. Prasanna, A. Bakshi, and L. Pianelo, "Workflow instance detection: Toward a knowledge capture methodology for smart oilfields," in *Proc. of IRI*, 2008, pp. 363–369.
- [11] "Schlumberger ECLIPSE," <http://www.slb.com/>.
- [12] "Petex IPM," <http://www.petex.com/products/>.
- [13] "OPM," <http://twiki.ipaw.info/bin/view/Challenge/OPM/>.
- [14] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distributed and Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.
- [15] A. V. Aho and M. J. Corasick, "Efficient string matching: An aid to bibliographic search," *Communications of the ACM*, vol. 18, no. 6, pp. 333–340, June 1975.
- [16] Y. L. Simmhan, B. Plale, and D. Gannon, "A survey of data provenance in e-science," *SIGMOD Record*, vol. 34, no. 3, pp. 31–36, September 2005.