Available online at www.sciencedirect.com
ScienceDirect
Web Semantics: Science, Services and Agents on the World Wide Web 5 (2007) 28–38
JOURNAL OF Web Semantics
www.elsevier.com/locate/websem

Provenance-based validation of e-science experiments

Simon Miles *, Sylvia C. Wong, Weijian Fang, Paul Groth, Klaus-Peter Zauner, Luc Moreau

School of Electronics and Computer Science, University of Southampton, Highfield, Southampton SO17 1BJ, United Kingdom

Received 15 June 2006; received in revised form 30 October 2006; accepted 30 November 2006
Available online 4 January 2007

Abstract

E-science experiments typically involve many distributed services maintained by different organisations. After an experiment has been executed, it is useful for a scientist to verify that the execution was performed correctly or is compatible with some existing experimental criteria or standards, not necessarily anticipated prior to execution. Scientists may also want to review and verify experiments performed by their colleagues. There are no existing frameworks for validating such experiments in today's e-science systems. Users therefore have to rely on error checking performed by the services, or adopt other ad hoc methods. This paper introduces a platform-independent framework for validating workflow executions. The validation relies on reasoning over the documented *provenance* of experiment results and *semantic descriptions* of services advertised in a registry. This validation process ensures experiments are performed correctly, and thus results generated are meaningful. The framework is tested in a bioinformatics application that performs protein compressibility analysis.

Keywords: Provenance; E-science; Process validation; Semantic service description

1. Introduction

Very large scale computations are now becoming routinely used as a methodology to undertake scientific research: success stories abound in many domains, including physics (griphyn.org), bioinformatics (mygrid.org.uk), engineering (geodise.org) and geographical sciences (earthsystemgrid.org). These large scale computations, which underpin a scientific process usually referred to as *e-science*, are ideal candidates for use of Grid technology [8].

E-science experiments are typically formed by invoking multiple services, whose compositions are modelled as workflows [9]. Thus, experimental results are obtained by executing workflows. As part of the scientific process, it is important for scientists to be able to verify the correctness of their own experiments, or to review the correctness of their peers' work. Validation ensures results generated from experiments are meaningful.

Traditionally, program validation has been carried out in two complementary manners. On the one hand, *static verification* analyses program code or a workflow before it is executed and establishes that the program/workflow satisfies some properties. These verifications are extensively researched by the programming language community. Examples include type inference, escape analysis and model checking. They typically depend on the semantics of the programming language being analysed. On the other hand, static verification is complemented by *run-time* checking, which is carried out when the program executes, and verifies that data values satisfy constraints, expressed by either types or assertions.

Such validation methods suffer from limitations when workflows are executed in dynamic open environments. First, programs (or workflows) may not be expressed in languages that analysis tools operate on, or may not be directly available because they are exposed as services, hereby preventing static analysis. Second, in general, in open environments, we cannot make the assumption that services always check that their inputs or outputs match their interface specifications (if available at all); furthermore, such interfaces may be under-specified (for instance, many bioinformatics services tend to process and return strings encoding specific biological sequence data); as a

* Corresponding author. Tel.: +44 23 8059 8309.
E-mail addresses: sm@ecs.soton.ac.uk (S. Miles), sw2@ecs.soton.ac.uk (S.C. Wong), wf@ecs.soton.ac.uk (W. Fang), pg03r@ecs.soton.ac.uk (P. Groth), kpz@ecs.soton.ac.uk (K.-P. Zauner), l.moreau@ecs.soton.ac.uk (L. Moreau).

1570-8268/$ – see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2006.11.003

result, no guarantee exists that specific, domain-level types will be checked dynamically.

A new, more specific limitation comes from the evolving conduct of e-science. Studies of user practice have shown that rapid development cycles are being adopted by e-scientists, in which workflows are frequently modified and tuned and scientific models are evolved accordingly. As a result, it is important for scientists to be able to verify that previous experimental results are compatible with recent criteria, models and requirements. Since these models did not necessarily exist at experiment design or execution time, it is a necessity to perform such validation *after* the experiment has been completed.

The *provenance* of a piece of data denotes the process by which it is produced. Provenance-aware applications are applications that record documentation of their execution so that the provenance of the data they produce can be obtained and reasoned over. We have studied a range of e-science application domains and established that they have a range of requirements for provenance-awareness [15]. In the former requirements study, many examples of experiment validation were discovered and in varying domains. For example, in a distributed particle physics experiment, there was a requirement to verify that those library versions used to analyse the experiment data were not ones known to contain bugs. In partially lab-based biology and chemistry experiments, requirements for validation included checking that health and safety rules had been followed by experiments in the past month. In a more general, computer-science centred example [25], processes can be validated to ensure, for fault tolerance, that multiple services assumed to be independent did not actually depend on the same, possibly faulty, service. We refer the reader to the survey for the full range of such use cases.

In this paper, our thesis is that provenance-based validation of experiments allows us to verify their validity after experiments have been conducted. Specifically, our contributions are: (a) a provenance-based architecture to undertake validation of experiments; (b) the use of semantic reasoning in undertaking validation of experiments; (c) an implementation of the architecture and its deployment in a bioinformatics application in order to support a set of use cases. Our experimentation with the system shows that our approach is tractable and performs efficiently.

The structure of the paper is as follows. Section 2 describes some use cases we have identified that require experiment validation. Section 3 briefly discusses current approaches to e-science experiment validation and explains why it is necessary to perform validation after an experiment was executed. Section 4 introduces the proposed framework for validation of workflow execution. Section 5 then describes how the architecture can be applied to the use cases introduced in Section 2. In Section 6, we discuss how semantic reasoning is essential in properly establishing the validity of experiments. Section 7 then presents results from an implementation of the validation framework with an e-science application (specifically, the protein compressibility analysis experiment). The paper finishes with discussion in Section 8 and conclusions in Section 9.

2. Use cases

The motivation for this work comes from real problems found by scientists in their day-to-day work. Therefore, in this section, we introduce a number of use cases in the bioinformatics domain where it is necessary to perform some form of validation of experiments *after* they have been completed. As identified in the use cases below, while service-based validation can only be performed at run-time, it is sometimes necessary to validate an experiment after it has been executed. Third parties, such as reviewers and other scientists, may want to verify that the results obtained were computed correctly according to some criteria. These criteria may not be known when the experiment was designed, because criteria evolve as science progresses. Thus, it is important that previously computed results can be verified according to revised sets of criteria.

Use Case 1 *((Interaction validity, interface level)). A biologist, B, performs an experiment on a protein sequence. One stage of this experiment involves generating a pre-specified number of permutations of that sequence. Later, another biologist, R, judges the experiment results and considers them to be suspicious. R determines that the number of permutations specified was an invalid value, e.g. it was negative.*

In this example, we consider that the service provider could have specified a restriction for the number of permutations to non-negative integers in the service schema, since the parameter only makes sense for non-negative integers. However, this does not guarantee that the service will validate the data against the schema at run-time. In general, whether validation is carried out at run-time is service specific.

In Use Case 1, *B* could have entered a negative value for the number of permutations. In this case, the value is incorrect because it does not conform to the restrictions and requirements as specified by the interface document of the service. By validating the experiment using its provenance, *R* can determine that *B* entered an invalid value for the number of permutations, and thus the results generated by the experiment were not meaningful.

Use Case 2 *((Interaction validity, domain-level)). A bioinformatician, B, downloads a file containing sequence data from a remote database. B then processes the sequence using an analysis service. Later, a reviewer, R, suspects that the sequence may have been a nucleotide sequence but processed by a service that can only analyse meaningfully amino acid sequences. R determines whether this was the case.*

Nucleotides and amino acids are two separate classes of biological sequences, but the symbols used in the syntax of nucleotides are a subset of those used for amino acids. Therefore, it is not always possible to detect which type of sequence is used by superficially examining the data. The service used in Use Case 2 could require an amino acid sequence as its input. If a nucleotide sequence was accidentally used rather than an amino acid sequence, the problem would not be detected at run-time, and the experiment results would not be meaningful.

Given that many bioinformatics services operate on strings, the biological interpretation of a piece of data is information