Towards Design Support for Provenance Awareness: A Classification of Provenance Questions

Paraskevi Zerva
Department of Informatics
King’s College London
London, United Kingdom
paraskevi.zerva@kcl.ac.uk

Steffen Zschaler
Department of Informatics
King’s College London
London, United Kingdom
szschaler@acm.org

Simon Miles
Department of Informatics
King’s College London
London, United Kingdom
simon.miles@kcl.ac.uk

ABSTRACT
As the complexity of online services increases, there is a corresponding need for service-oriented systems to provide support for answering questions about how they have processed and produced data. This need is particularly evident in compositions of services, where audits of each individual service’s use do not provide a connected picture of the composition’s processing history. The provenance awareness of a system is its ability to answer questions about the history of its processing, through recording provenance data during execution. As the size and usage of a system increases, so can the size of the provenance data recorded, leading to increased demands on storage and decreased performance of the service. However, the exact impact of provenance recording depends on what details of the service execution are being documented. Our goal is to make provenance awareness accessible as an explicit non-functional property (NFP) in composite service specifications, as is common for performance or reliability properties. This would enable composite service designers to analyse the properties of their services to see whether they meet users’ requirements based on the provenance questions they could ask about the service’s outputs and the dependent performance, storage and other properties. We present a preliminary approach towards this end, focusing on the step of categorising potential provenance questions according to their effect on other NFPs, so that the provenance awareness of a service can be specified as the categories of questions it can answer.

Keywords
Formal specification, non-functional properties, provenance awareness, provenance question categories, service oriented architecture (SOA), web service composition

1. INTRODUCTION
Service-oriented computing (SOC) [18] comprises both a software design philosophy and a system architecture, the service-oriented architecture (SOA), in which a collection of loosely coupled services communicate with each other through published and discoverable interfaces and message-exchanging protocols. SOA aims to provide services to be consumed either by end-user applications or by other services (clients) or business processes distributed in a network. Those services are well-defined, self-contained, and platform-independent computational entities that perform certain business functions [17]. They can be dynamically located, invoked and combined, so that new services can be dynamically created by composing locally and/or remotely available services. The latter are discovered through service brokers that publish service descriptions along with additional information such as the services’ reliability, trustworthiness, quality of the service, service level agreements, etc. [25].

Service composition is important when no single service is sufficient to satisfy all the required functionality. Service-oriented systems aim to satisfy a set of functional and non-functional requirements (NFRs). In general, functionality is the key for selecting the appropriate candidates for service composition. However, in the case where existing services have the same functionality but different non-functional properties (NFPs), such as performance, reliability and security, those should be considered in order to best satisfy the overall NFRs. Composition according to NFRs is not a trivial task as there are dependencies between NFPs and properties of the execution environment that influence them, e.g., workload, physical resources. Formal modelling of those properties is crucial for enabling precise specification and analysis at design time. A relatively recent concern in service non-functional requirements has been to provide answers to the users about the documented history of the service’s processing and what process led to the data produced during this process, referred to as provenance. Our goal is to make provenance awareness accessible as an explicit NFP to enable designers to explore trade-offs between provenance awareness and performance and storage requirements.

The provenance of some data is the process that led to that data being produced [14]. Provenance awareness of a service-oriented system refers to its ability to answer different classes of questions about the documented history of a service’s processing. Our intention is to analyse at design-time in what ways a service-oriented system is provenance aware. Services may be compositions of other services, the latter being referred to as individual services, and the questions may relate to how the composition has occurred. The provenance questions that a user may ask are classified into provenance question categories, where one category corresponds to particular kinds of information being recorded in
the provenance, and so particular sets of provenance questions being answerable. Recording and storing provenance data has a potentially huge impact on the performance and storage demands of a service [21, 3], so distinguishing different categories by the kind of data recorded, helps us analyse the impact of provenance recording on a given service or composition’s other NFPs.

This paper presents initial work towards providing explicit provenance question categories for service-oriented systems. We form those categories by analysing the kinds of provenance data that need to be recorded, collected and stored to answer realistic questions.

The remainder of this paper is structured as follows. Related work on formal specification of NFPs for web services and on provenance in service-oriented systems is presented in Section 2. Section 3 presents a motivating example that sets the scope of our work and that we use as a reference to present different categories of user questions requiring provenance data to be answered. Those questions can be classified into different provenance question categories that we call facets, presented in Section 4. Next, Section 5 discusses the impact of provenance collection, recording and storage mechanisms for different question categories. It also presents a high-level analysis of the trade-off of provenance awareness in terms of those requirements and the impact on other service NFPs. We conclude with our future perspectives in Section 6.

2. RELATED WORK

This section discusses related work in the area of non-functional properties and provenance in service-oriented systems.

NFPs take the form of quantitative quality attributes specified as constraints over quality values, as measured by specified metrics. For example, services have associated NFPs such as performance (measured by metrics such as response time and throughput), reliability and security. The precise specification of NFPs is important, so that designers can ensure that their services will meet non-functional requirements at run-time. Modelling languages for expressing NFPs of individual services, and accompanying approaches to predict NFPs of composite services, have been developed. At design time, the purpose of formal modelling and prediction of NFPs is to compare different design alternatives and to analyse whether those alternatives meet the NFRs. Where services are composed, the service composition’s NFPs will depend on those of the individual services composed as well as on details of the composition.

Several standards have been developed that provide concepts for modelling services, business processes [1] and choreographies [9] but the focus is on the functional properties of services and their compositions. WSOL [22] and WSLA [10] are semi-formal languages that allow the specification of a number of NFP+s, including performance, dependability, security and cost. SLang [11] is similar, but offers more formality. However, neither WSLA/WSOL nor SLang deal with NFPs of composed services. The model-driven approach of Gallotti et al. [2, 5] supports model-based analysis of service compositions, with a focus on the assessment of performance and reliability NFPs, by automatically transforming a service composition design model into an analysis model, which then feeds a probabilistic model checker for quality prediction. Filieri et al. [4] present a novel approach to reliability modelling and analysis of a component-based system that allows dealing with multiple failure modes and studying the error propagation among components.

While there are theories and models for some NFPs (most notably performance and reliability as shown in some examples above), a large number of other properties are less well studied. An insufficiently explored area in the NFP specification is that of provenance awareness. Notably, none of the languages described above provide a specification of provenance awareness for composed services.

The importance of provenance data for SOA has been increasingly recognised and a number of approaches have been developed to provide a provenance framework for them. Tsai et al. [23, 24] analyse the unique characteristics of data provenance in a SOA system and provide a framework for classification and collection of provenance data. However, this work focuses on security, reliability and integrity of data as it is routed through a SOA system rather than considering the NFPs of services and their dependencies in composite service scenarios. Michlmayr et al. [12] present an approach in which service runtime events are captured, but it is again focused on security issues such as data integrity and access control mechanisms. Rajbhandari et al. [20] propose an approach for capturing and recording provenance in SOAs, including a scalability analysis of the effects of increases in provenance data, in order to evaluate the performance of their recording mechanism. They extend this preliminary work [19] with an analysis tool that makes use of provenance data to assist in evaluating the trustworthiness of workflow execution results. Muniwimy-Reddy et al. [15, 16] define desirable properties for distributed provenance storage systems and design alternatives for storing data and provenance on cloud-based web service platforms (e.g., Amazon’s Web Services platform (AWS)). They mainly propose the design and implementation of three protocols for maintaining provenance and data on the cloud, evaluating each protocol with respect to the properties they have established. Yet, their focus is mainly on properties in their design that can guarantee the availability and the scalability of the cloud, while they evaluate their approach by making a comparison of the cost and performance of the proposed three provenance storage protocols. In our work, we are interested in considering the trade-off of performance and storage demands with provenance awareness as an explicit NFP for service-oriented systems realised through composite services.

The research community has also been interested in methods to make applications provenance aware. The Provenance Aware Service Oriented Architecture (PASOA) project [8, 14] proposes a generic, technology and application independent architecture that meets technical requirements of different use case scenarios. Miles et al. [13] propose a software engineering technique for making applications provenance-aware, by adapting designs to enable their interaction with a provenance middleware layer. In [7] Groth et al. outline the user requirements that arise in designing provenance systems for the web space by focusing on three central dimensions: content, management and use. Similarly to our approach they try to make explicit these dimensions according to the requirements they imply through a number of use case scenarios. Yet, their work do not address how provenance awareness can be expressed as a NFP.

In summary, while there is a large body of work on specifying NFPs of service-based systems, it does not cover provenance awareness. At the same time, while provenance aware-
ness of service-based systems has been discussed in the provenance literature, this has not been done in a way that allows provenance-awareness to be specified as an explicit NFP and analysed in relation to other NFPs of a composed system. With our work, we aim to address this gap.

3. MOTIVATING EXAMPLE

In this section, we introduce a motivating use case scenario. Figure 1 depicts a service composition for booking multiple travel arrangements. The composition is represented as a statechart, where a state corresponds to an abstract task in which a named service operation is invoked, with given input and output messages. In a given instantiation, each task would be performed by a particular service discovered at run-time. The tasks are to book a flight \( (t_1) \), book a hotel \( (t_2) \), book an attraction to visit \( (t_3) \), calculate the driving time from the hotel to the attraction \( (t_4) \), rent a bike to travel between the hotel and attraction \( (t_5) \), rent a car to travel between the hotel and attraction \( (t_6) \), and take payment for the travel package by credit card \( (t_7) \). The composition is expressed in terms of sequential transitions, AND-states to express concurrency, and multiple transitions from a state to represent conditional branching.

A user interacts with a travel planner service, which discovers then invokes individual services, such as a flight booking service or a hotel booking service. We assume that services’ descriptions are published by their providers through a broker, and discovered through that broker. The travel planner uses the service descriptions to bind with and invoke each service.

The user has requirements on a number of NFPs such as performance, reliability, security and provenance awareness. With regards to provenance awareness, the user wants to ask questions related to the provenance of the travel planner results, such as why a bike was booked instead of a car, why a high cost flight was booked, or which bank authenticated the credit card. Provenance data required to answer questions like these must conform to some model, and we adopt the W3C PROV model [6] for our work, which we summarise briefly here. The primary concepts of the model are entities, activities and agents. Activities represent processes that have occurred, such as the execution of a service operation. Entities are digital, physical or conceptual things that existed, such as the messages exchanged between services or part of a service’s state. Activities generated new entities and used existing entities, and one entity may have been derefered from another. An agent denotes something that was responsible for an activity having taken place, such as a service composition’s aggregator or orchestrator, where we say that the agent was associated with an activity it was responsible for. PROV also allows us to express the role played by an entity or agent in an activity, the time at which an entity was generated or used by an activity, the plan that was followed by an activity in execution, and much more.

Figure 2 graphically depicts a snippet of PROV data for a particular run of our travel planner. We show two agents: the travel planner, which works as an aggregator and is associated with all the individual services in the composition, and a third party that manages the authentication of the credit card. The input and output messages (entities) correspond to the data and parameters passed from one service to another during composition plan execution. Messages may either be generated by a service execution as output messages or may be derived from output messages of previous service executions and become input to new services. For clarity, we have not included all provenance information in this illustrative snippet.

To answer particular provenance questions, we would have to record the values of data such as the message contents and service identifiers. Different questions require different kinds and levels of detail, and it depends on what the user will or is likely to require as to what provenance data should be recorded. Our first step, therefore, is to examine the categories of questions that require different provenance data to be recorded.

4. QUESTION CATEGORIZATION

As expressed earlier, our ultimate aim is to specify provenance-awareness, i.e. the extent to which a (composite) service can answer provenance questions regarding past executions, as a non-functional property (NFP). This will allow the service’s designers and potential clients to compare the properties of the service with their requirements. As well as considering whether the service will be able to answer provenance questions they will want to put, they will also care about other non-functional properties, such as performance and storage costs, affected by large-scale provenance capture.

For each different service or composition, the provenance questions will be different. Therefore, a general model of provenance-awareness as a NFP cannot be based on specific questions, but generalised categories of questions. To meet our objectives, these categories should be such that we can distinguish the effects of a service supporting that category on other NFPs (performance, storage) of the service.

Different kinds of data, of different sizes and occurring with different frequencies, are required to answer different provenance questions. It is the capture of these kinds of data that can affect the other NFPs of a service. Therefore, a category of provenance questions should be defined by the kinds of data it requires to be captured and stored.
Many realistic provenance questions require a combination of kinds of data. Therefore, instead of dividing all questions into distinct categories, we define a number of facets, each corresponding to some kind of captured data, and then show how a given provenance question exhibits a combination of facets, meaning that it requires those facets’ corresponding data to be captured. The facets act as orthogonal axes for the space of provenance questions.

In this section, we first define nine facets for provenance questions, and then discuss questions exhibiting a combination of facets. We illustrate them using example questions from the motivating scenario of the previous section.

### 4.1 Facets for Provenance Awareness

We identify nine facets for categorising provenance questions below, with accompanying example questions that could be asked of our motivating composite service. In this preliminary work, each facet is defined intuitively by the kinds of data that are required to be captured and stored during a service’s execution. This is not a precise definition and does not include the dependencies between data within a provenance graph, which are also relevant for answering the questions. In our future work, we expect to define the facets in terms of ‘templates’ for provenance graphs, that express how the data are interrelated in a way abstracted from particular service instances.

1. **Service and provider identity** Requires: DNS host names or IP addresses of the service providers; service names or URIs of selected services; IP addresses of the broker. Example questions: (1.1) Who was the provider of the service used for flight booking? (1.2) What service broker discovered the flight booking service?

2. **Data flow** Requires: Input and output messages exchanged between services, parameters passed in message exchanges, an identifier for every service (service name or URI), which messages relate to every service. Example questions: (2.1) What were the input parameters passed in the driving time calculation service (Min)? (2.2) What was the output of the driving time calculation service (Mout)? (2.3) Were any data inserted as changes to the initial user preferences for the flight booking service (e.g., 1st class instead of 2nd class seat, ticket for under-25-year olds)?

3. **Resources & physical deployment** Requires: Information on availability of different types of resources (e.g., during individual / composite service execution), IP address as an identifier for every resource. Example questions: (3.1) What resources (memory, hard disks, CPUs) were used for the attraction search service concurrently to the flight and hotel booking services? (3.2) What resources were available during the execution of the composite service?

4. **Time** Requires: Information on composite (or individual) service execution timestamps: timestamps for start / completion, for service invocation / discovery, identifier for the service (service name or URI). Example questions: (4.1) When did the composite service travel planner execution start / end? (4.2) When was the car rental invoked / discovered? (4.3) How long did it take to complete the authentication process of the credit card?

5. **Routes not followed** Requires: Information on which branches of the workflow have been taken, on the actual composition execution plan and on additional branches that have not been followed including the relevant services, their policies and their provided interfaces. It is important to clarify that this facet requires more information than simply data flow (facet 2) or design information (facet 9): Recording data flow only may never record a potential route, if it is never followed by any service execution. Conversely, recording design information only will provide information about all potential routes, but will not tell us which ones have actually been followed or not followed in a particular service execution. Example questions: (5.1) Are there branches in the composite service that have not been explored (e.g., for the bike rental service)? (5.2) What are the alternative routes for authenticating and confirming the travel package
booking by using Paypal service instead of simply sending the credit card information?

6. Past history Requires: Information to be stored for more than one past invocations of individual and composite services; different questions will differ according to the specific information to be stored (as covered by other facets) and the number of past invocations / the length of time for which to store the information. Example questions: (6.1) How often has the service of the chosen flight booking service been accessed / invoked the last 2 days? (6.2) What was the failure rate in the time frame the travel planner service is running the last 8 months?

7. NFPs and QoS Requires: Different provenance data to be recorded for every particular NFP of every service that takes part in the composition. Information on SLAs between the different services e.g., \((t_4)\) and \((t_5)\). Example questions for individual services: (7.1) What was the response time for \((t_4)\)? (7.2) What was the processing time for \((t_5)\)? (7.3) What was the network latency for \((t_7)\)? (7.4) What was the execution cost for the hotel booking service \((t_9)\)? (7.5) What was the reputation for service \((t_6)\)? (7.6) Was all credit-card information encrypted securely throughout processing? Example questions for composite services: (7.7) What was the average response time / processing time / overall execution time for the travel planner service? (7.8) What was the total execution time for the travel planner service? (7.9) What was the execution cost for the travel planner service?

8. Actors Requires: Information about different actors that own / manage / monitor services, an identifier for them (e.g., IP address), their association to services or data that they manage. Example questions: (8.1) Who managed (e.g., particular bank) the authentication of the credit card number? (8.2) Was the authentication of the credit card made by a trusted third party? (8.3) Which third party takes control of the monitoring for QoS (e.g., response time, availability)?

9. Design information Requires: Service interfaces (required and provided); dependencies in the composition of services; information on protocols or message format used between services. Example questions: (9.1) Which task invokes the service instantiating \((t_6)\)? (9.2) Which services preceded \((t_7)\)? (9.3) Did all the messages exchanged between the different services of the composite service use the same format / protocol?

4.2 Multi-Facet Provenance Questions

Many real questions will exhibit multiple of the facets defined in the previous section. In this section, we present some examples illustrated using our motivating case study.

Example questions: Was the reputation the reason that the flight booking service was selected? Was a prior violation of the SLA for reliability the reason that the attraction search service of service provider Expedia was not selected? Facets exhibited: Facets 2, 6 and 7. Provenance data required: Values passed directly between services about service’s reputation (facet 2) during service invocation; past history data of user’s feedback about the service (facet 6); SLAs of \((t_7)\) with other services (facet 7); an identifier for the service (service name or URI, facet 1).

Example questions: At what time did the flight booking service fail to be completed or invoked? At which timestamp was service \((t_7)\) not available? Facets exhibited: Facets 4, 6 and 7. Provenance data required: Information on failures of completion / invocation (facet 6) or latency (facet 7) for individual services that take part in the composition, timestamps for those critical events (facet 4); information on link-up state / availability (facet 7) of the service and timestamps for those states (facet 4).

In this section, we have made a classification of provenance questions for service-oriented systems considering individual and composite services. We have defined facets of provenance questions according to the different data required to be recorded and stored in order to answer those questions. Realistic questions may exhibit a composition of multiple facets. In every case, we have to consider the provisions we have to make in designing a system with the necessary recording and storing mechanisms in order to capture the necessary provenance data. Next, we discuss on those provisions and we try to analyse the requirements on collection and storage.

5. DISCUSSION

In the previous section, we presented a provenance question categorization by defining facets. The distinction between facets has an impact on other NFPs. For example, the question “Which flight booking services were available at the time of performing the service discovery at broker L?” requires data of several facets, such as the availability of the service (facet 7), the timestamp of discovery (facet 4), the service name or URI to identify the particular broker service location (facet 1). As questions require more kinds of data, the more that provenance collection mechanisms need to record and store, and so the higher the impact on storage demands and performance of the service.

Provenance data need to be recorded and stored in order for future provenance questions to be answered. In Table 1 we show a ranking of the impacts of the different facets. At this stage in our work, the ranking is intuitive rather than experimentally verified. The Past history data facet requires storage of provenance data for potentially long time intervals, possibly the lifetime of the service, so could require any amount of storage. Similarly, monitoring software required to assess NFPs can capture the operation of services at a fine detail. Therefore, both of these facets are said to have “very high” impact. Messages between services can be arbitrarily large, even if they only occur at well defined points, so the impact is “high”. Timestamps, choices between routes, and the identities of actors in the system are all kinds of information that may be present frequently during a service’s execution, but are each small data to record. The resources used by services will not normally change during execution, meaning recording this has lower impact. Finally, some data, such as interface specifications, will be present after execution without recording, though copies may need to be kept to answer future provenance questions.

Our ultimate aim is to formally specify the provenance-awareness of services as a NFP in such a way that it can trade off with other NFPs on which it has an impact, such as performance and storage. This means specifying the facets that the service supports in a precise but generalised way. We expect to do this by specifying templates for the provenance that will be recorded. Composite services are of particular interest, because the properties of a composition is a function of the properties of its parts, and this applies to provenance-awareness as much as to other NFPs. Additionally, the composition’s design clearly relates to the form of
<table>
<thead>
<tr>
<th>Facets</th>
<th>Provenance collection required</th>
<th>Storage and performance impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Past history data</td>
<td>Recording mechanisms to capture failure or success rates or keeping logs of information for service reputation for a long time interval or life of the service</td>
<td>very high</td>
</tr>
<tr>
<td>NFPs and QoS</td>
<td>Recording mechanisms for different NFPs for individual and composite services (e.g., performance monitors, link-up monitors for availability)</td>
<td>very high</td>
</tr>
<tr>
<td>Data flow</td>
<td>Recording mechanisms to continuously capture the parameters and data passed in the input and output messages (during message exchanges) between different services of the composition plan execution</td>
<td>high</td>
</tr>
<tr>
<td>Time</td>
<td>Service and message timestamps annotations, use of clocks to record timestamps for service discovery, service execution, service invocation and critical events</td>
<td>medium</td>
</tr>
<tr>
<td>Service and provider identity</td>
<td>Recording mechanisms to capture information on the identification of service providers and services (IP address and URIs)</td>
<td>medium</td>
</tr>
<tr>
<td>Routes not followed</td>
<td>Recording mechanisms to capture branches of the workflow being taken during the composition execution plan</td>
<td>medium</td>
</tr>
<tr>
<td>Actors</td>
<td>Recording mechanisms to capture information on the identification of third parties and their association with services they manage or interact with</td>
<td>medium</td>
</tr>
<tr>
<td>Resources &amp; physical deployment</td>
<td>Recording mechanisms to capture the availability of resources and resource usage for the services during the composition execution plan</td>
<td>low</td>
</tr>
<tr>
<td>Design information</td>
<td>No need for provenance recording mechanisms</td>
<td>storage of application data, no impact on performance</td>
</tr>
</tbody>
</table>

Table 1: Facets ranking according to impact of provenance recording on storage and performance

provenance that will be recorded, and so to the templates used for specifying provenance-awareness.

6. CONCLUSION

This paper discussed general requirements of provenance awareness for service-oriented systems realised through individual and composite services. We focus on the trade-off of provenance awareness in terms of storage and data collection and how those affect other NFPs such as performance. In order to solve this problem we have taken initial steps to analyse the requirements of answering different types of provenance questions for composite services. To address those requirements this paper proposed a comprehensive facet classification of provenance questions based on the needs in provenance data that have to be collected and stored. Finally, we motivated our facet classification on a travel planner composite service scenario.

There are two strands of future work that we plan to follow: First, we will explore the formal specification of provenance-question categories. We will investigate the use of templated PROV specifications for this purpose and will aim to establish a way of checking whether the specific data required to answer a specific provenance question is captured by a particular provenance template. Secondly, we will aim to underpin our discussion from Sect. 5 with actual performance-analysis results for specific use-case scenarios.

7. REFERENCES


