



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 870811



Social cohesion, Participation, and Inclusion
through Cultural Engagement

D4.6: Provenance and Process analysis layer: Supporting use cases

(v1.0)

| Deliverable information | |
|---------------------------------|---|
| WP | WP4 |
| Deliverable dissemination level | PU Public |
| Deliverable type | R Document, report |
| Lead beneficiary | OU |
| Contributors | UH |
| Date | 28/04/2023 |
| Authors | Enrico Daga, Paul Mulholland, and Jason Carvalho (OU) |
| Document status | Final |
| Document version | v1.0 |

Disclaimer: The communication reflects only the author's view and the Research Executive Agency is not responsible for any use that may be made of the information it contains

PAGE INTENTIONALLY BLANK

Project Information

Project Start Date: 1st May 2020

Project Duration: 36 months

Project Website: <https://spice-h2020.eu>

Project Contacts

Project Coordinator

Silvio Peroni

ALMA MATER STUDIORUM -
UNIVERSITÀ DI BOLOGNA
Department of Classical Philology
and Italian Studies – FICLIT

E-mail: silvio.peroni@unibo.it

Project Scientific Coordinator

Aldo Gangemi

Institute for Cognitive Sciences
and Technologies of the Italian
National Research Council

E-mail: aldo.gangemi@cnr.it

Project Manager

Adriana Dascultu

ALMA MATER STUDIORUM -
UNIVERSITÀ DI BOLOGNA
Executive Support Services

E-mail: adriana.dascultu@unibo.it

SPICE Consortium

| No. | Short name | Institution name | Country |
|-----|------------|---|----------------|
| 1 | UNIBO | ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA | Italy |
| 2 | AALTO | AALTO KORKEAKOULUSAATIO SR | Finland |
| 3 | DMH | DESIGNMUSEON SAATIO - STIFTELSEN FOR DESIGN- MUSEET SR | Finland |
| 4 | AAU | AALBORG UNIVERSITET | Denmark |
| 5 | OU | THE OPEN UNIVERSITY | United Kingdom |
| 6 | IMMA | IRISH MUSEUM OF MODERN ART COMPANY | Ireland |
| 7 | GVAM | GVAM GUIAS INTERACTIVAS SL | Spain |
| 8 | PG | PADAONE GAMES SL | Spain |
| 9 | UCM | UNIVERSIDAD COMPLUTENSE DE MADRID | Spain |
| 10 | UNITO | UNIVERSITA DEGLI STUDI DI TORINO | Italy |
| 11 | FTM | FONDAZIONE TORINO MUSEI | Italy |
| 12 | MAIZE | MAIZE SRL (previously CELI SRL) | Italy |
| 13 | UH | UNIVERSITY OF HAIFA | Israel |
| 14 | CNR | CONSIGLIO NAZIONALE DELLE RICERCHE | Italy |

Executive Summary

SPICE is an EU H-2020 project dedicated to research on novel methods for citizen curation of cultural heritage through an ecosystem of tools co-designed by an interdisciplinary team of researchers, technologists, museum curators engagement experts, and user communities. This technical report D4.6 presents the results of Task 4 of Work Package 4: “Provenance and process analysis layer“. This is a final report that complements and integrates D4.5, which discussed the requirements and envisioned implementation of provenance support for the Linked Data Hub to support the analysis of citizen curation processes. In this deliverable, we report on how the linked data layer of the project supports provenance and process analysis. We report on the Data Journey ontology that we implemented for supporting the analysis of data science processes and show its applicability to citizen curation pipelines, considering the case study of scripting citizen curation with the Deep Viewpoints system, developed in the context of the IMMA case study (WP7). The SPICE Linked Data Hub supports the recording of changes on the data, and these can be used to generate a Data Journey representation of user processes at the data level. We describe how such a representation can be generated, and discuss the potential of data journeys for analysing citizen curation applications, for the benefit of data managers and application developers.

Document History

| Version | Release date | Summary of changes | Author(s) - Institution |
|----------------|---------------------|--|---|
| v0.1 | 01/02/2023 | Prepared template | Enrico Daga (OU) |
| v0.2 | 20/02/2023 | Outline and initial structure | Enrico Daga (OU) |
| v0.3 | 01/03/2023 | Introduction and partial related work | Enrico Daga (OU) |
| v0.4 | 31/03/2023 | Completed draft | Enrico Daga and Paul Mulholland and Jason Carvalho (OU) |
| v0.5 | 26/04/2023 | Feedback from reviewers | Tsvi Kuflik (UH), Belén Díaz Agudo (UCM) |
| v0.6 | 27/04/2023 | Integration of feedback from reviewers | Enrico Daga (OU) |
| v1.0 | 28/04/2023 | Submission to EU | Coordinator |

Table of contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 1 |
| 2 | Background and related work | 2 |
| 3 | Deep Viewpoints | 4 |
| 4 | The SPICE Linked Data Hub | 9 |
| 5 | The Data Journey Ontology (DJO) | 11 |
| 5.1 | Definition | 11 |
| 5.2 | Ontology description | 12 |
| 6 | Data journeys for citizen curation | 17 |
| 6.1 | Data journeys for citizen curation applications | 17 |
| 6.2 | Generation of data journeys | 18 |
| 6.3 | An example data journey: a script on Ilcruthach by Kevin Mooney | 21 |
| 7 | Discussion and conclusions | 27 |

1 Introduction

SPICE is an EU H-2020 project dedicated to research on novel methods for citizen curation of cultural heritage through an ecosystem of tools co-designed by an interdisciplinary team of researchers, technologists, museum curators and engagement experts, and user communities. In the SPICE project, we are researching on a linked non-open data management platform that publishes the content of museum archives and supports the development of sophisticated citizen curation applications, including the collection of user-generated content to be integrated within the archives of memory institutions.

This technical report D4.6 presents the results of Task 4 of Work Package 4: “Provenance and Process analysis layer”. This deliverable follows and integrates D4.5 - Provenance and Process analysis layer: Requirements analysis. There, we discussed how the concept of *Data Journey* is fundamental to provide process analysis capabilities to citizen curation applications.

In this line of research we observe citizen curation from the perspective of data science. The notion of data journey has been discussed in the data studies literature. Specifically, Leonelli [1] defined it as the “movement of data from their production site to many other sites in which they are processed, mobilised and re-purposed.”. The work in data studies emphasises the difficulty of understanding data journeys empirically because of a *multitude of perspectives*. Our definition has the objective of being consistent with the one of Leonelli but also to relate with the literature from Web semantics, specifically, data provenance. A Data Journey is a multi-layered, semantic representation of a data processing activity, linked to the digital assets involved (code, components, data). Hence, we introduce a layered semantics perspective to the definition of data journeys for describing citizen curation applications in terms of user activities (changes on data objects), datanodes (versions of objects at certain points in time), and assets (evolving data objects).

The focus of this deliverable is the implementation of a data journeys knowledge graph on top of the SPICE Linked Data Hub. We report on the Data Journey ontology that we implemented for supporting the analysis of data science workflows and show its applicability to citizen curation pipelines, derived from a case study of the SPICE project: the Deep Viewpoints system. The SPICE Linked Data Hub supports the recording of changes on the data (Activity Log), and these can be used to generate a Data Journey representation of user processes. In this deliverable, we describe how such representation can be generated, and discuss the potential of data journeys for analysing citizen curation applications, for the benefit of data managers and application developers. Specifically, we show the potential of the SPICE LDH Activity Log by building a Knowledge Graph following the Data Journeys ontology with the Deep Viewpoints application data.

The rest of the deliverable is structured as follows. The next chapter is dedicated to background and related work (Chapter 2). We review relevant literature on provenance and process analysis. Next, we introduce the SPICE Linked Data Hub, specifically, its activity log functionality. Chapter 3 describes the Deep Viewpoints system, on which we base our case study. Chapter 4 introduces the SPICE Linked Data Hub. The main contribution of this work is presented in Chapter 5, where we introduce the Data Journeys Ontology (DJO) that has the aim of describing data science workflows through abstraction. Chapter 6 shows how we generate a knowledge graph from LDH activity logs with the support of the SPARQL Anything system, developed in the SPICE project. Finally, we discuss the potential of such a representation for a deeper understanding of citizen curation applications (Chapter 7).

2 Background and related work

An overview of relevant related work was given in the previous deliverable D4.5 [2]. Here, we touch upon essential elements to support the reader in understanding the content of the coming chapters.

The result of the recent "Stakeholders' survey on a European collaborative cloud for cultural heritage" [3] reports on a shared interest in the development of collaborative infrastructures for cultural heritage organisations. Regarding tools support, the most popular option refers to systems for creating, sharing, and re-using interactive digital content. However, citizen curation applications will generate a large amount of data through very different types of applications. Therefore, it is important to develop methods that help data analysts to gather sufficient information to support the understanding of user activities and, therefore, support museum practitioners and other domain experts in analysing the processes behind citizen curation. In this deliverable, we argue that the journey a citizen curation object goes through, its lineage or provenance, is a powerful unit of analysis for making sense of citizen curation applications. To help the reader, we briefly survey related work in provenance and process analysis.

The broad notion of data journeys has been discussed in the data studies literature. In particular, with the recently edited volume by Leonelli and Tempini, which brings together different case studies from plant phenomics to climate data processing and studies them through the lens of data journeys [1]. Fundamentally, they argue that the journey a dataset goes through, its lineage or provenance, is a powerful unit of analysis for explaining it.

The need to understand the provenance of data has been well documented in data management [4] and web [5] literature, which has investigated approaches for representing, extracting, querying, and analysing provenance information. Indeed, the importance of understanding provenance for web information led to the W3C Prov standard for provenance interchange [6] as well as the recent Coalition for Content Provenance and Authenticity¹. We refer to the two surveys cited above for more information about provenance systems. Our work, in particular, builds upon these existing representations in order to provide a *multi-layered* view of a data journey allowing different levels of abstraction to sit alongside one another. Specifically, we build on the notion of *datanode* as specified in the Datanode Ontology [7], developed to express complex data pipelines to reason upon the propagation of licences and terms and conditions in distributed applications [8].

The need to tie data to the workflow that generated it has been recognised in the scientific workflow community [9]. An essential contribution of this work is that, for workflow analytics and reusability, different granularity levels of workflow representations and associated provenance (e.g. high-level tasks in the domain vs command-line tool parameters) should be captured [10, 11]. These representations can then be bundled together with the corresponding data assets and other documentation, creating a research object [12] that can be published using web standards [13, 14]. However, most applications (data science processes but also end-user systems like in the case of citizen curation) are not expressed with those formalisms. Instead, data journeys are complementary to these existing definitions and are designed to be extracted from existing sources, be they code, UIs, or event traces.

In [15], we analysed the landscape of data management applications in cultural heritage from the point of view of citizen curation, and found little mention of provenance (with the exception of ResearchSpace) and no mention of process analysis. ResearchSpace [16] is an open source platform for enabling researchers to create, link, share and search data by using Semantic Web languages and technologies. It provides an assertion and argumentation model for tracing multiple perspectives on historical facts; enables a multilevel visual representation of resources; allows the creation of data and narratives in the form of knowledge graphs; captures provenance of data; expresses researchers' views as graphs connecting narratives, data, processes, and arguments. ResearchSpace builds on the knowledge graph platform, enabling customisation and extensibility of the interaction with the graph database through the use of Semantic Web standards and expressive ontologies for schema modeling based on CIDOC-CRM. However, the notion of provenance implemented by ResearchSpace relates to metadata associated with

¹<https://c2pa.org>

digital assets (provenance of a digital image) rather than providing insight on the *production* of the digital content, in our case, the citizen contributed content.

We conclude that the notion of computational provenance, intended as the ability to monitor and report changes (user operations) on citizen curated objects is original to the SPICE project. To the best of our knowledge, this is the first time that such a notion is applied to citizen curated content in the context of cultural heritage applications.

More insight on the state of affairs of cultural heritage systems in relation to citizen curation can be found in the survey article published by the SPICE project team on the ACM Journal of Computing and Cultural Heritage [15].

3 Deep Viewpoints

Deep Viewpoints [17] is a web app developed using the Angular framework that supports two forms of citizen participation: interpretation and mediation. Interpretation involves citizens developing their own understanding of, and response to, artworks. The interpretation process is guided by scripts comprising statements, artworks and prompts (e.g. questions) that help the museum visitor to develop and share their own viewpoint. Mediation involves authoring new scripts that guide the process of interpreting one or more artworks. Crucially, in Deep Viewpoints, scripts are not only authored by museum professionals but also by citizens to offer fellow visitors to the museum a new way of looking at and thinking about artworks. In particular, the mediation process has been carried out by a number of community groups traditionally under-served by the cultural sector in order to bring new citizen voices into the curatorial process.


The Deep Viewpoints app can be operated in three user modes: anonymous, logged in and admin. Anonymous users can view and respond to scripts. Their script responses go into a moderation queue that can be handled by either the author of the script to which they responded, or admin. Logged in users can respond to scripts, create scripts, moderate their own scripts, select artworks from the IMMA collection and add and edit new themes that can be associated with the scripts. In addition, admin can view, modify, or delete any contributed content as well as manage user accounts.

All data created in the Deep Viewpoints app is stored in the Linked Data Hub for cross-reference with the IMMA collection. This enables the data to be queried in order to explore relationships between the collection and the citizen-contributed content, for example finding scripts that contain an artwork by a particular artist, or responses to a question within a script about a certain artwork. The Linked Data Hub supports the handling of licences and user policies between apps such as Deep Viewpoints app and datasets such as the IMMA collection on which the app relies. Deep Viewpoints can also make use of server-side content monitoring provided through the Linked Data Hub such as the detection of hate speech and personal information in citizen contributions.

As introduced in Deliverable D6.7 [18], Deep Viewpoints has a script authoring interface for use by both museum staff and visitors. Prior to script authoring, the logged in user searches the IMMA collection to create their own personal selection of artworks for use in the scripts, as shown in Figure 3.1. The selected artworks can then be used to author a script.


From the script page, the user can create a new script, open a script, or remove it (Figure 3.2). They can also view the script as it will appear to others and view any responses it has received. Once a script is opened, the authoring process is organised into four tabs: Artworks, Stages, Themes and Title. From the Artworks tab, the user can add any of their selected artworks to the script (see Figure 3.3). From the Stages tab, the user can construct their script as a sequence of stages (see Figure 3.4). Each stage can contain none, some or all of the artworks associated with the script. A statement stage provides information or a perspective and does not request any input from the follower of the script. A question stage asks a single question, for example about one or more artworks, to which the script follower can provide a response. A help text can also optionally be associated with the question which the script follower can choose to reveal for further assistance. A multi-question stage asks a set of questions about the same artworks. The author of the script can decide whether to present the questions in a predefined sequence or to be shuffled in a random order. The multi-question stage was proposed to parallel the onsite visitor experience in which a tour guide may ask a series of questions at a specific point in the tour. A multi-question stage has an optional help text that can provide additional guidance on how to answer the questions (e.g. look slowly at the artwork from left to right) or who the questions are for (e.g. these questions are for anyone who has emigrated to another country). A story stage provides a story stem or story opener and instructions on how to continue the story. This stage was inspired by the story completion method, a projective technique in counselling psychology for eliciting perceptions of the topic introduced in the story stem.

The themes tab is used to associate themes with the script, as shown in Figure 3.5. The title tab, shown in Figure 3.6,




Carrier
Kevin Mooney
2021

Delete



Blighters
Kevin Mooney
2018-21

Delete



Once Upon a time 1
Patricia Hurl
1986

Add

Search IMMA collection:

Madonna ×

- Madonna** and Child (after Masaccio), Patricia Hurl, 1984-85
- Madonna** and Child with Onlookers, David Godbold, 1992
- Madonna** Irlanda, Micheal Farrell, 1978

Figure 3.1: Selecting artworks from the IMMA collection.

is used to determine how the script will be summarised on the Deep Viewpoints homepage: its title, description and accompanying artwork.

The four-stage script resulting from the above authoring process is shown in Figures 3.7a - 3.7d. This script was authored by members of a youth group visiting IMMA and it constitutes our reference use case in this deliverable.

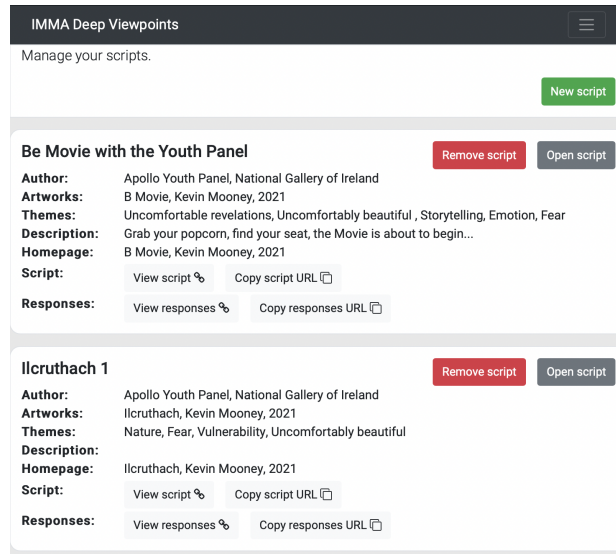


Figure 3.2: Script authoring page.

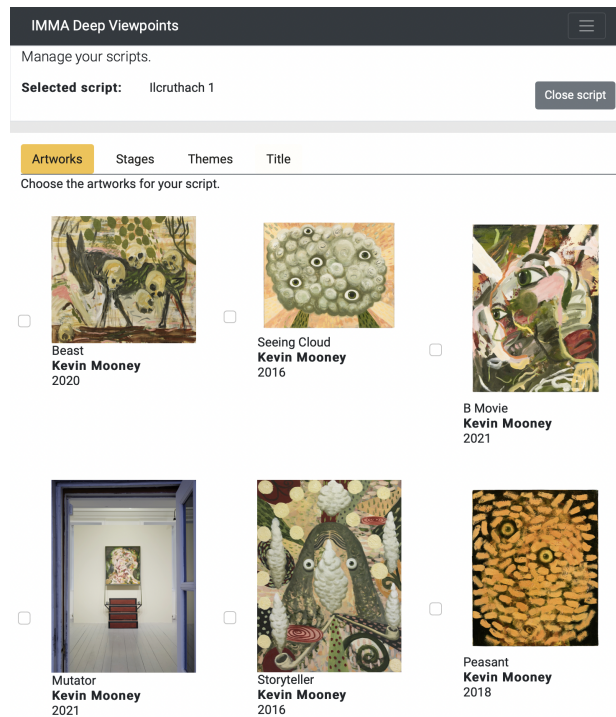


Figure 3.3: Selecting artworks for use in the script.

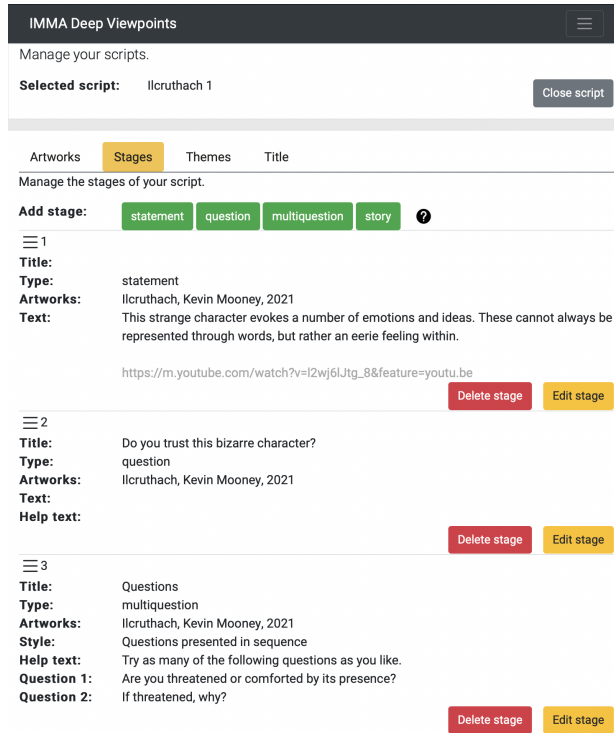


Figure 3.4: Authoring stages of the script.

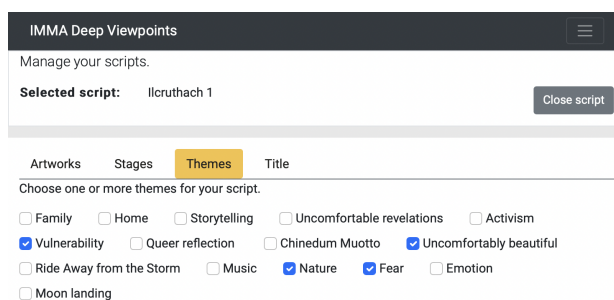


Figure 3.5: Selecting themes of the script.

IMMA Deep Viewpoints

Manage your scripts.

Selected script: Ilcruthach 1 Close script

Artworks Stages Themes **Title**

Decide how your script will be presented in Deep Viewpoints.

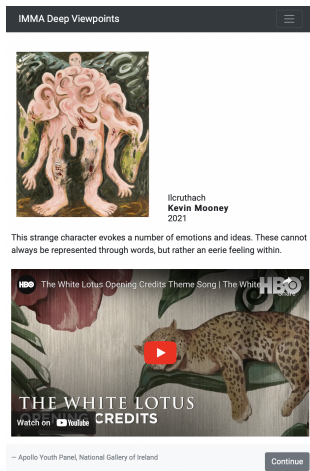
Title:

Description:

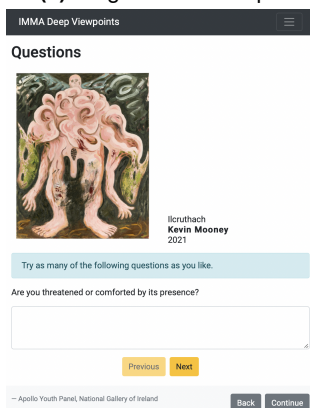
A short description of what the script is about, what will it involve, why people should take part

Homepage artwork: Ilcruthach, Kevin Mooney, 2021

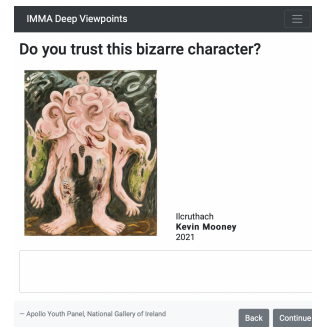
Figure 3.6: Setting the title, description and homepage artwork of the script.



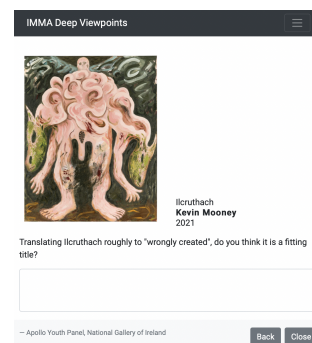
(a) Stage 1 of the script.



(c) Stage 3 of the script.



(b) Stage 2 of the script.



(d) Stage 4 of the script.

Figure 3.7: Script stages.

4 The SPICE Linked Data Hub

The SPICE Linked Data Hub capitalises on the findings of [15]. The platform incorporates technologies that allow users to configure content adapters to a variety of sources in a flexible way, supporting an open-ended heterogeneity of data sources, which include: (a) downloadable files, (b) data incorporated in HTML websites, and (c) Web APIs. In addition, the LDH supports fine-grained access control management, visibility and discoverability of assets, and brokering to negotiate access and use. Data managers can detail information on copyright, licensing, and attribution related to any asset managed by the system. The platform incorporates a content monitoring solution that supports data managers in the identification of personally identifiable information (PII), helping them in complying with the General Data Protection Regulation (GDPR). Linked Data Transformers can be developed using the novel system SPARQL Anything¹ [19], to support extracting information from those sources, specifically, considering data in CSV, JSON, and HTML formats. For example, collection metadata of the Irish Museum of Modern Art (IMMA) was extracted from the museum Website².

As part of the preparations for the Process Analysis Layer and also WP4.2 requirements 1, 5, 29 and 30 (Table 3.1 in D4.2 [20]), the LDH API implements a detailed and comprehensive activity log providing a full audit trail of all API operations. Log entries take the form of JSON-LD documents and are managed internally by the API using the same JSON document store that is used for all other SPICE LDH data.

Since the activity log itself resides within a regular LDH dataset, the full suite of searching, filtering, sorting and paging operations made available by the API can be used on log entries. This functionality enables us to further develop process analysis applications and layers that are able to reconstruct timelines of activity pertaining to specific segments of interest within the LDH. This feature also opens up the possibility of plugging in replication agents alongside the LDH that can rebuild either all or parts of LDH datasets on external platforms that may be better suited to application-specific querying, workflows or reasoning layers. This approach was experimented with by SPICE, for example, with the Semantic Annotator component developed in WP3 (See D6.8 [21]).

Below is a typical application log entry:

```

1 {
2   "_id": "621dd7bd0763c253c53a1de9",
3   "@id": "621dd7bd0763c253c53a1de9",
4   "@context": "https://mkdf.github.io/context",
5   "@type": [
6     "al:Create",
7     "al:ActivityLogEntry"
8   ],
9   "al:summary": "Create[POST] - http://api2.mksmart.org/object/<datasetID> - Dataset: <datasetID> - Key:***removed*** - Create new document",
10  "al:request": {
11    "@type": "al:HTTPRequest",
12    "al:agent": {
13      "@type": "al:Agent",
14      "al:key": "***removed***"
15    },
16    "al:endpoint": "http://api2.mksmart.org/object/<datasetID>",
17    "al:httpRequestMethod": "al:POST",
18    "al:parameters": [],

```

¹<http://sparql-anything.cc>

²<https://github.com/SPARQL-Anything/showcase-imma>

```

19     "al:payload": "{ \"_id\": \"1099\", \"category\": \"52-a\", \"value\": 44.7
20     },
21     "al:documentId": "1099",
22     "al:datasetId": " <datasetID> ",
23     "_timestamp": 1646122941,
24     "_timestamp_year": 2022,
25     "_timestamp_month": 3,
26     "_timestamp_day": 1,
27     "_timestamp_hour": 8,
28     "_timestamp_minute": 22,
29     "_timestamp_second": 21
30 }
    
```

The activity log is described in JSON-LD format and records the type of operation performed. For example, `al:Create` indicates that a new object was added to the dataset. The `al:request` property shows the type of HTTP request and content. The content of the request is registered in the attribute `al:payload`. Finally, the log includes the dataset, the document identifier, and the time the operation was issued by what client application (registered key, here omitted).

In D4.2 we introduced the features as a back-end operation only available to LDH administrators and LDH plugin components such as the RDF replication agent. As an evolution of this component, the LDH activity log is now available to an API endpoint (`/changes`) that lets users access a subset of details from these log entries relevant to the dataset. Combined with filtering and sorting functions, as well as appropriate access controls, this functionality enables third parties to build and run their own replication agents for individual datasets.

The changes endpoint of the API can be used to query the Linked Data Hub's activity log for updates to a single dataset. Create, update and delete operations are returned. By default, the newest items are returned first. We illustrate its usage as follows.

GET /changes/{dataset-uuid} Retrieve a list of changes for a single dataset providing the most recent first. Optional parameters (supplied as URL query parameters) include:

document-id Only return changes to a single document

timestamp Only return entries that have occurred since this timestamp

limit The maximum number of entries to return

sort Setting this parameter to '1' reverses the sort order of results to return the oldest items first

We show now some API invocations as examples. With the following request, we retrieve a list of changes for dataset 1234567"

```

curl -X GET "https://api2.mksmart.org/changes/1234567" -H "accept: */*" -H "
  Authorization: Basic_ZHNmc2RmOnNkZnNkZg=="
    
```

Retrieve a list of changes for document 'doc05678', within dataset 1234567, since 01/01/2023 and ordered oldest to newest:

```

curl -X GET "https://api2.mksmart.org/changes/1234567?document-id=doc-56789&
  timestamp=1672531200&sort=1" -H "accept: */*" -H "Authorization: Basic_
  ZHNmc2RmOnNkZnNkZg=="
    
```

Further details can be found in D6.8 - APIs Specification and Deployment [21].

5 The Data Journey Ontology (DJO)

In this Chapter, we first introduce a general definition of computational data journey and we discuss it in relation to citizen curation systems. Next, we describe the data journey ontology and its extension to support citizen curation traces in the context of the SPICE Linked Data Hub management system.

5.1 Definition

The notion of a data journey has been discussed in the data studies literature. Specifically, [1] defined it as the “movement of data from their production site to many other sites in which they are processed, mobilised and repurposed.”. The work in data studies emphasises the difficulty of understanding data journeys empirically because of a multitude of perspectives. Our definition has the objective of being consistent with the one of [1] but also to relate with the literature from Web semantics, specifically, data provenance [22]. Hence, we introduce a layered semantics perspective to the definition of data journeys:

A Data Journey is a multi-layered, semantic representation of a data processing activity, linked to the digital assets involved (code, components, data).

Thus, a journey is multi-layered, as to allow a multiplicity of perspectives that can be overlaid to describe the process. This multiplicity can help to capture (parts of) the context around a data journey while still allowing for computational analysis to be performed. Hierarchical, because any useful representation needs to be linked to the concrete assets involved, either directly or via intermediate abstractions.

Although our definition is open-ended and allows for multiple (even alternative) perspectives to co-exist, in this work, we conceptualise data journeys in the following layered structure:

- *Resources*: resources used in the data journey such as source code files, software libraries, services, or data sources.
- *Source Code*: human readable and machine executable instructions, for human authoring, such as a Python script.
- *Machine Representation*: any machine interpreted representation of the instructions, such as an Abstract Syntax Tree (AST) or a query execution plan.
- *Datanode Graph*: as defined in [23], a graph of *data-to-data* relationships, such as variables, imported libraries, and input and output resources. Such abstraction provides a structure of the data flows, abstracting from issues such as control flow, and focusing on *data-to-data* dependencies.
- *Activity Graph*: a graph of high-level activities, inspired by the notion of Workflow Motifs [10].

As a guide example, we can consider the following python code (Kaggle: <https://www.kaggle.com/dansbecker/random-forests>):

```

1 import pandas as pd
2 # Load data
3 melbourne_file_path = '../input/melbourne-housing-snapshot/melb_data.csv'
4 melbourne_data = pd.read_csv(melbourne_file_path)
5 # Filter rows with missing values
6 melbourne_data = melbourne_data.dropna(axis=0)
    
```

Line 1 and 3 mention references to *resources*: a software library (pandas) and a CSV file. The Kaggle notebook itself is the human readable and machine executable *source code*. When the python interpreter executes this script, it does so by means of an intermediate *machine representation*, for example an abstract syntax tree is produced by

parsing the code, and this is then exploited by the interpreter. However, the code structure implies knowledge that is not explicitly stated. This includes the *data flow*, as defined in [23], which provides an abstraction of the way data is transferred and manipulated from the file in Line 3 to the variable `melbourne_data` in Line 6 (*datanode graph*). Finally, this source *hides* high-level intentions, for example, the fact that libraries and data are reused and some data preparation is performed, i.e., a column is dropped in Line 6 (*activity graph*).

While the first three components pre-exist the data journey, i.e. they do not pertain to the *knowledge level* [24], the remaining represent two distinct, although interconnected, representation layers.

5.2 Ontology description

In this section, we report the details of the Data Journeys Ontology. In this deliverable, we use this ontology as a general reference representation. We describe the ontology in detail, although we are not necessarily going to use all of its elements to represent citizen curation applications.

We build on previous work and design an ontology as the reference knowledge model of a data journey, satisfying our definition. Our methodology is based on reusing successful, relevant models, and completing them with concepts derived from the data used in this work. Specifically, we reuse concepts from three approaches: the W3C Provenance Ontology PROV-O [25], the Workflow Motifs Ontology [10] and the Datanode ontology [7]. Compared to the pre-existing models, DJO provides a unified view of the data journey, linking the two layers (the data flow layer and the activity layer). In addition, it extends the Datanode Ontology by specifying node types.

The Data Journey Ontology (DJO) reuses fundamental concepts from those three ontologies into a new conceptual model with layered semantics. The namespace of the DJO is:

```
1 djo: <http://purl.org/datajourneys/>
```

The design rationale of DJO is one of layered semantics. *The ontology should be able to capture fine-grained data flows but also high-level activities, as super-imposed abstractions.* An overview of the ontology is provided in Figures 5.1 and 5.2¹. We note that we align with broader software engineering concepts rather than specific machine learning notions as the aim here is to represent a generic data processing pipeline.

The core components are three classes: `Datanode`, `Activity`, and `Journey`.

Datanode The `Datanode` class represents a data object. We performed a thematic analysis of 20 randomly chosen data science notebooks in order to identify possible data node types, depending on the role they have in the process.

The result is as follows:

- **Constant**: a value hard-coded in the source code, for example, the value of an argument of a machine learning instruction
- **Input**: a pre-existing data object served to the program for consumption and manipulation.
- **Output**: a data node produced by the program
- **Parameter**: any data node which is not supposed to be modified by the program but is needed to tune the behaviour of the process. For example, the process splits the data source into two parts, 20% for the test set and 80% for the training set. 2, 20%, and 80% are all parameters.
- **Support**: any data node pre-existing the program, which is used without manipulation. Includes:
 - **Reference**: any datanode used as background knowledge by the program, for example, a lookup service or a knowledge graph. Such datanode pre-exists the program and is external to the program.
 - **Capability**: any datanode which provides capabilities to the program, including pre-existing modules, functions, and imported libraries.

¹The visual representations are generated with OWLGrEd [26]: http://owlgred.lumii.lv/online_visualization.

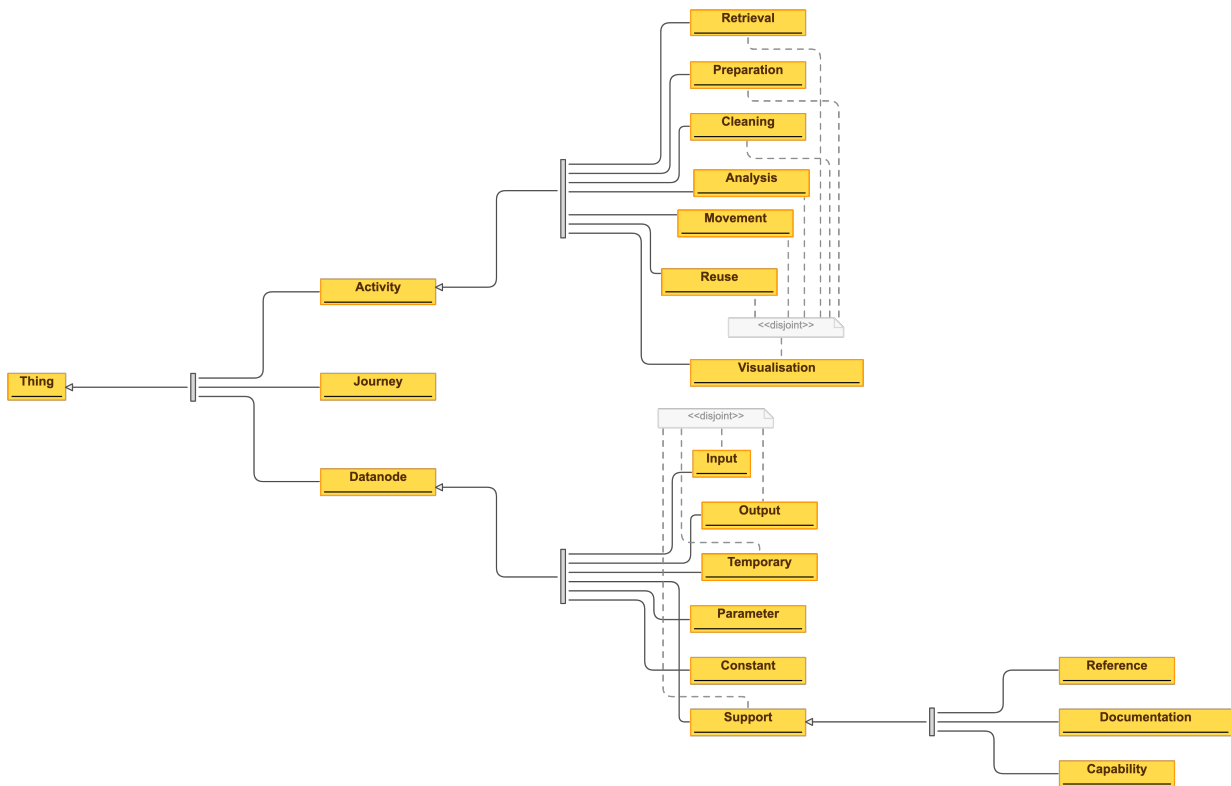


Figure 5.1: DJO: overview of the classes.

– Documentation: any datanode which does not affect the operation of the program, such as source code comments and documentation.

- Temporary: any datanode produced and then reused by the program that is not intended to be the final output

The following groups of classes are mutually disjoint:

- Input, Output, Support, and Temporary
- Capability, Documentation, and Reference

Datanodes are connected to each other in a data flow akin to a provenance graph. Possible relationships subsume the provenance notion of *derivation*. Table 5.1 shows the hierarchy of object properties, with references to the notions reused from existing sources, and documented in the ontology with a `rdfs:seeAlso` relation.

Activity In a Data Journey, sibling datanodes are supposed to be grouped together in *activities*, in order to provide a more abstract representation of the data journey. The activities included in DJO are mutually disjoint sub-classes of Activity: Analysis, Cleaning, Movement, Preparation, Retrieval, Reuse, and Visualisation. Except for Reuse, the other classes are derived from concepts defined by the Workflow Motifs ontology. Activities instances are connected to each other in a sequence, using the object property `previous` (and its inverse `next`). We note that the semantics of these two properties does not imply derivation but merely state the

Table 5.1: List of object properties connecting instances of class `Datanode` in the Data Journey Ontology. Sources: PROV-O (PO), Datanode Ontology (DN), and Workflow Motifs (WM).

| Property | PO | DN | WM |
|-----------------------|----|----|----|
| derivedFrom | ✓ | ✓ | |
| analysedFrom | | ✓ | ✓ |
| cleanedFrom | | ✓ | ✓ |
| computedFrom | | ✓ | |
| copiedFrom | | ✓ | |
| movedFrom | | ✓ | ✓ |
| optimizedFrom | | ✓ | |
| preparedFrom | | | ✓ |
| augmentedFrom | | | ✓ |
| combinedFrom | | ✓ | ✓ |
| extractedFrom | | ✓ | ✓ |
| filteredFrom | | ✓ | ✓ |
| formatTransformedFrom | | | ✓ |
| groupedFrom | | | ✓ |
| sortedFrom | | | |
| splitFrom | | | |
| refactoredFrom | | ✓ | |
| remodelledFrom | | ✓ | |
| retrievedFrom | | | ✓ |
| visualisedFrom | | | ✓ |

sequence of activities in a data journey. This approach in representing lists is inspired by the Sequence ontology design pattern² [27, 28] (see also [29] for a survey on modelling solutions for treating sequential information in RDF). Each activity instance is then connected to the involved datanodes through the property `includesDatanode` (and its inverse `inActivity`).

Journey Finally, both activities and Datanodes are meant to be connected to one data journey via the functional property `inJourney`.

Some data science ontologies are tailored specifically to represent machine learning applications and it is useful to look into how those relate to DJO. Table 5.2 illustrates how DJO relates to fundamental concepts of data mining and machine learning workflows, as defined by two domain-specific ontologies for data mining (DMOP [30]) and machine learning (ML-Schema [31]). DJO can be extended by direct reuse of those concepts. In addition, DJO covers a broader set of notions familiar to any data science experiment, including, for example, visualisation and reuse.

²Sequence ODP: <http://ontologydesignpatterns.org/wiki/Submissions:Sequence>.

Table 5.2: Conceptual mappings between DJO, DMOP and ML-Schema.

| DJO | DMOP | ML-Schema |
|------------------------------------|--|--|
| <i>Journey</i> | DM-Workflow, DM-Process, DM-Experiment | Study, Experiment, Run |
| <i>Datanode</i> | | |
| Datanode > Constant | | HyperParameterSetting |
| Datanode > Input | IO-Object, DM-Data | Data, Dataset |
| Datanode > Output | IO-Object, DM-PatternSet, DM-Model | Model, ModelEvaluation |
| Datanode > Parameter | Parameter | HyperParameter |
| Datanode > Temporary | | Dataset, Feature |
| Datanode > Support | | |
| Datanode > Support > Reference | Measure | Data, Dataset, Evaluation-Measure, EvaluationSpecification, EvaluationProcedure |
| Datanode > Support > Capability | DM-Software, DM-Operator, DM-Model | Model, Feature, Software, Implementation |
| Datanode > Support > Documentation | DM-Task, DM-Algorithm, DM-Hypothesis, DM-Operation | Task, Algorithm, ModelCharacteristic, DataCharacteristic, ImplementationCharacteristic |
| <i>Activity</i> | DataProcessingTask | |
| Activity > Analysis | InductionTask, Modeling-Task, PatternDiscovery-Task | |
| Activity > Cleaning | DataCleaningTask | |
| Activity > Movement | | |
| Activity > Preparation | DataReductionTask, DataAbstractionTask, DataTransformationTask | |
| Activity > Retrieval | | |
| Activity > Reuse | | |
| Activity > Visualisation | | |

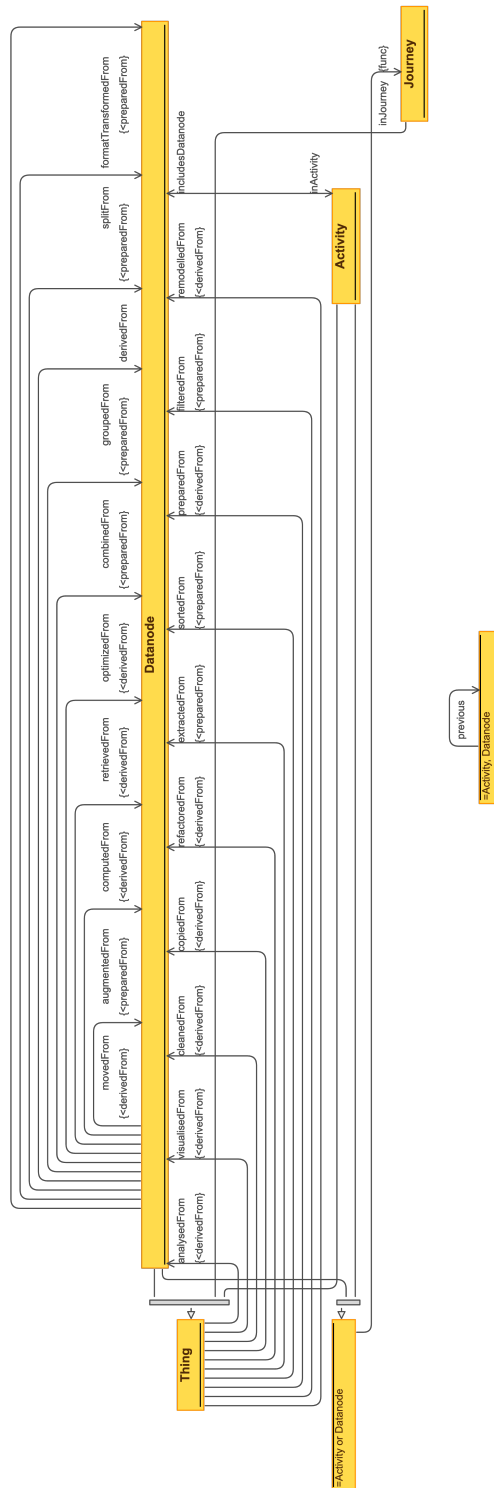


Figure 5.2: DJO: overview of the properties.

6 Data journeys for citizen curation

In this chapter we illustrate how the *data journeys* paradigm can be applied to citizen curation of cultural heritage and how such semantic layer can be generated from data managed by the SPICE Linked Data Hub, to support advanced provenance and process analysis. First, we describe how data journeys can be used to conceptualise content derived from LDH activity logs of citizen curation applications, considering the case of the Deep Viewpoints application. Next, we illustrate the knowledge generation process. We conclude the chapter with an exemplary data journey and discuss analytics that can be derived thanks to this semantic representation.

6.1 Data journeys for citizen curation applications

The data journeys definition is a sufficiently general definition that can be easily applied to citizen curation applications:

- *Resources*: resources pertaining to citizen curation applications managed in the LDH are data objects in the form of JSON *objects*.
- *Source Code -> Object Versions*: instead of source code, users of citizen curation applications operate via advanced user interfaces, the execution trace of such operations are the starting point of our data journeys, equivalent to the source code of data science applications.
- *Machine Representation -> Event Traces*: such execution traces (object versions and associated operations) are captured in a machine-representation, the LDH Activity Log.
- *Datanode Graph*: as defined in [23], a graph of *data-to-data* relationships. Abstracting from issues such as control flow, we focus on linking the sequence of versions of objects (snapshots) in the LDH as *data-to-data* dependencies.
- *Activity Graph*: a graph of high-level activities, selected according to the citizen curation application at hand (in our study, the Deep Viewpoints system).

In the following, we apply DJO for citizen curation, and extend it with terms useful to capture elements of the SPICE LDH activity logs and the Deep Viewpoints (DV) application. We define the following types of activities:

Create script The user – either a museum practitioner or a visitor – creates a new script, starting the design process.

Name script The user assigns a name to the script.

Select artwork The user selects an image from the catalogue of artworks of the museum.

Edit themes The user assigns or removes a theme associated to the script.

Add stage The user adds a new stage to the script. A stage includes a prompt question with a media object associated with it (an image, a video, ...).

Edit stage The user modifies a stage.

Delete stage The user removes a stage from the script.

Include artwork The user includes the selected artwork alongside the stage prompt.

Edit status The user changes the status of the script, so that other users can see it (and run it) or, it makes it private for further modifications.

Include exhibition The script is associated with a specific exhibition (this feature allows the collection of multiple scripts relevant to a specific museum event).

Include homepage Use the artwork to present the script on the homepage of Deep Viewpoints.

Respond The script was run and user responses are collected.

Approve response The response was monitored and published on the website.

6.2 Generation of data journeys

In this section, we describe the generation of a Knowledge Graph of Data Journeys from the Deep Viewpoints application data.

The LDH activity log includes information about changes in data objects. For example, in the case of the Deep Viewpoints application, the JSON objects describe Scripts, which are curated by museum practitioners or citizens, and Activities, which are essentially produced by users interacting with a script. Because each change preserves the state of the JSON object, we can derive a *data journey* as the creation and update of interlinked data objects, and look into the changes performed to the objects to *explain* the process.

The knowledge graph construction process was performed using SPARQL Anything [32] and was organised as follows¹. We consider the activity log entries of the Deep Viewpoints dataset in the SPICE Linked Data Hub² as a collection of JSON files, each one including details of a change event in the dataset. For convenience, we downloaded the files locally and designed a set of SPARQL Anything queries to transform the data into an RDF knowledge graph according to DJO. At the time of this work, the dataset included 15798 activity log events on 2018 objects. Table 6.1 shows statistics of changes per object in the Deep Viewpoints dataset.

Table 6.1: Number of changes per object in the dataset

| Changes | Objects |
|---------|---------|
| < 10 | 1656 |
| < 50 | 170 |
| < 100 | 168 |
| > 100 | 24 |
| - | 2018 |

In what follows, we refer to the data and queries in the supplementary material at <https://github.com/spice-h2020/spice-data-journeys>. We list the content of the folder ‘json/’ with the query in Listing 6.1 and we save the output to a CSV file.

Listing 6.1: Listing JSON files

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX odr1: <http://www.w3.org/ns/odr1/2/>
4 PREFIX fx: <http://sparql.xyz/facade-x/ns/>
5 PREFIX xyz: <http://sparql.xyz/facade-x/data/>
6 PREFIX dct: <http://purl.org/dc/terms/>
7 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
8
9 SELECT ?file ?fname
10 WHERE {
11     SERVICE <x-sparql-anything:> {
12         fx:properties fx:location "./json/" ;
13         fx:archive.matches ".*deep_viewpoints.*".
14     } [] fx:anySlot ?file
15 } .
16 BIND ( replace(substr( ?file, fx:String.lastIndexOf(?file, "/" ) + 2), ".json", "" ) AS ?fname )
17 }
    
```

Output is saved locally into a CSV of 15798 items. In the second step, we query each JSON file to generate DJO activity graphs from the JSON logs, including datanodes representing the version of the object at the time of the

¹Source code can be accessed at <https://github.com/spice-h2020/spice-data-journeys>

²<https://spice.kmi.open.ac.uk/dataset/details/138>

change and entities representing the objects (as evolving entities). This operation is illustrated in Listing 6.2. At this stage, the relations between activities and datanodes are not there yet. We will generate 'previous' relations later.

Listing 6.2: Generate RDF representation of changes in the activity log, as LDH activities (Create, Update, Delete), datanotes (versions of objects at a given point in time), and object entities (representing the evolving data object)

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX odrl: <http://www.w3.org/ns/odrl/2/>
4 PREFIX fx: <http://sparql.xyz/facade-x/ns/>
5 PREFIX xyz: <http://sparql.xyz/facade-x/data/>
6 PREFIX dct: <http://purl.org/dc/terms/>
7 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
8 PREFIX dv: <http://w3id.org/spice/ldh/deep-viewpoints/>
9 PREFIX djo: <http://purl.org/datajourneys/>
10 PREFIX djcc: <http://purl.org/citizencuration/>
11 PREFIX al: <https://mkdf.github.io/context/activity-log#>
12 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
13
14 CONSTRUCT {
15     ?changeEntity a djo:Activity ;
16     rdfs:label ?actionLabel ;
17     rdfs:comment ?actionComment ;
18     djcc:timestamp ?timestamp ;
19     #djcc:modified ?timestampDate ;
20     djcc:object ?objectEntity ;
21     djcc:operation ?actionTypeEntity ;
22     al:documentId ?changeId
23     .
24     ?datanode a djo:Datanode ;
25     rdfs:label ?datanodeLabel ;
26     rdfs:comment ?description ;
27     djo:hasActivity ?changeEntity ;
28     al:versionOf ?objectEntity ;
29     dv:authorName ?authorName ;
30     dv:author ?author ;
31     dv:editor ?editor ;
32     dv:scriptId ?scriptId ;
33     dv:scriptOpen ?scriptOpen ;
34     dv:scriptVisible ?scriptVisible ;
35     dv:hasArtwork ?hasArtwork ;
36     dv:hasTheme ?hasTheme ;
37     dv:hasExhibition ?hasExhibition ;
38     dv:snapshot ?payload
39     .
40     ?objectEntity a ?objectTypeEntity ; al:documentId ?documentId ; dv:id ?id ; rdfs:label ?objectLabel ;
41     rdfs:comment ?description .
42 } WHERE {
43
44     SERVICE <x-sparql-anything:> {
45         SERVICE <x-sparql-anything:> {
46             BIND (?_file as ?file) .
47             fx:properties fx:location ?file .
48             [] xyz:_id ?changeId ;
49             xyz:al%3AdocumentId ?documentId ;
50             xyz:_timestamp ?timestamp ;
51             xyz:al%253Arequest [ xyz:al%3Apayload ?payload ] ;
52             xyz:%2540type/rdf:_1 ?actionType .
53             FILTER (!fx:String.startsWith(str(?b), str(rdf:)))
54         } .
55
56         BIND(?payload as ?content) .
57         fx:properties fx:content ?content ;
58         fx:media-type "application/json" .
59         ?o a fx:root; xyz:id ?id ; xyz:type ?objectType .
60         optional { ?o xyz:authorname ?authorName } .
61         optional { ?o xyz:name ?name } .
62         optional { ?o xyz:description ?description } .
63         optional { ?o xyz:editor ?editor } .
64         optional { ?o xyz:author ?author } .
65         optional { ?o xyz:script/xyz:id ?scriptId } . # only activity
66         optional { ?o xyz:stage/xyz:id ?stageId } .
67         optional { ?o xyz:open ?scriptOpen } . # only scripts
68         optional { ?o xyz:visible ?scriptVisible } . # only scripts
69         optional { ?o xyz:artwork/rdf:_1 ?artwork } .

```

```

70     optional { ?o xyz:themeids/rdf:_1 ?theme } .
71     optional { ?o xyz:exhibitionids/rdf:_1 ?exhibition } .
72 }
73 BIND( IF(BOUND(?artwork), true, false) AS ?hasArtwork )
74 BIND( IF(BOUND(?theme), true, false) AS ?hasTheme )
75 BIND( IF(BOUND(?exhibition), true, false) AS ?hasExhibition )
76 ##
77 BIND("1970-01-01T00:00:00Z"^^xsd:dateTime + strdt(concat("PT", str(?timestamp), "S"), xsd:duration) AS ?
78     timestampDate ) .
79 BIND( CONCAT( ?objectType, " ", str(?id), ":", ?name ) AS ?datanodeLabel ) .
80 BIND( CONCAT( ?objectType, " ", str(?id) ) AS ?objectLabel ) .
81 BIND( CONCAT( replace(?actionType, "al:", "" ), " of ", ?objectLabel ) AS ?actionLabel ) .
82 BIND( CONCAT( replace(?actionType, "al:", "" ), " of ", ?objectLabel, " on ", str(?timestampDate) ) AS ?
83     actionComment ) .
84 ##
85 BIND( fx:entity(dv:, "change/", ?changeId ) AS ?changeEntity ) .
86 BIND( fx:entity(dv:, "object/", ?documentId ) AS ?objectEntity ) .
87 BIND( fx:entity(dv:, "type/", ?objectType ) AS ?objectTypeEntity ) .
88 BIND( fx:entity(dv:, "datanode/", ?documentId, "/", ?timestamp ) AS ?datanode ) .
89 BIND( fx:entity( al:, replace(?actionType, "al:", "" ) ) AS ?actionTypeEntity ) .
90 }

```

The query is executed on each one of the JSON data objects to generate an RDF representation according to DJO (the output is saved in the 'rdf/' folder, see supplementary material).

Next, we query the union of all the graphs generated in the previous step and extract the sequential changes per object. Specifically, we order the changes according to their occurring time and group them by object. We generate a sequence that will be used later to produce the 'previous' relations of the DJO activity and datanode graphs. Listing 6.3 shows the query that generates the sequence of changes.

Listing 6.3: Generation of sequence of changes according to the timestamp.

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX odrl: <http://www.w3.org/ns/odrl/2/>
4 PREFIX fx: <http://sparql.xyz/facade-x/ns/>
5 PREFIX xyz: <http://sparql.xyz/facade-x/data/>
6 PREFIX dct: <http://purl.org/dc/terms/>
7 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
8 PREFIX dv: <http://w3id.org/spice/ldh/deep-viewpoints/>
9 PREFIX djo: <http://purl.org/datajourneys/>
10 PREFIX djcc: <http://purl.org/citizencuration/>
11 PREFIX al: <https://mkdf.github.io/context/activity-log#>
12 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
13 SELECT ?changeEntity ?timestamp ?objectEntity
14 WHERE {
15     graph ?g {
16         ?changeEntity a djo:Activity ;
17         djcc:timestamp ?timestamp ;
18         djcc:object ?objectEntity
19     }
20 }
21 ORDER BY ?objectEntity ASC(?timestamp)

```

We use the list to generate the 'previous' relation first at Activity level (Listing 6.4) and then, querying the resulting graph, at Datanode level (Listing 6.5).

Listing 6.4: Linking datanodes with the DJO properties 'previous'.

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX odrl: <http://www.w3.org/ns/odrl/2/>
4 PREFIX fx: <http://sparql.xyz/facade-x/ns/>
5 PREFIX xyz: <http://sparql.xyz/facade-x/data/>
6 PREFIX dct: <http://purl.org/dc/terms/>
7 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
8 PREFIX dv: <http://w3id.org/spice/ldh/deep-viewpoints/>
9 PREFIX djo: <http://purl.org/datajourneys/>
10 PREFIX djcc: <http://purl.org/citizencuration/>
11 PREFIX al: <https://mkdf.github.io/context/activity-log#>
12 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
13

```

```

14 CONSTRUCT {
15     ?afterEntityIri djo:previous ?beforeEntityIri
16 } WHERE {
17     SERVICE <x-sparql-anything:> {
18         fx:properties fx:location "./data/deep-viewpoints-changes-as-list.csv" ; fx:csv.headers "true"
19
20         # changeEntity,timestamp,objectEntity
21         [] a fx:root ;
22         ?before [ xyz:changeEntity ?beforeEntity ; xyz:objectEntity ?objectEntity ] .
23
24         #
25         BIND ( fx:next(?before) AS ?after ) .
26
27         #
28         [] ?after [ xyz:changeEntity ?afterEntity ; xyz:objectEntity ?objectEntity ]
29
30     }
31
32     BIND(fx:entity(?afterEntity) AS ?afterEntityIri ) .
33     BIND(fx:entity(?beforeEntity) AS ?beforeEntityIri ) .
34 }

```

Listing 6.5: Linking datanodes with the DJO properties ‘previous’.

```

1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX odrl: <http://www.w3.org/ns/odrl/2/>
4 PREFIX fx: <http://sparql.xyz/facade-x/ns/>
5 PREFIX xyz: <http://sparql.xyz/facade-x/data/>
6 PREFIX dct: <http://purl.org/dc/terms/>
7 PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
8 PREFIX dv: <http://w3id.org/spice/ldh/deep-viewpoints/>
9 PREFIX djo: <http://purl.org/datajourneys/>
10 PREFIX djcc: <http://purl.org/citizenuration/>
11 PREFIX al: <https://mkdf.github.io/context/activity-log#>
12 PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
13
14 CONSTRUCT {
15     ?dnAfter djo:previous ?dnBefore
16 } WHERE {
17
18     ?activityAfter djo:previous ?activityBefore .
19     ?activityBefore a djo:Activity ;
20     ?actionBefore ?dnBefore . ?dnBefore a djo:Datanode .
21     ?activityAfter a djo:Activity ;
22     ?actionAfter ?dnAfter . ?dnAfter a djo:Datanode
23 }

```

6.3 An example data journey: a script on Ilcruthach by Kevin Mooney

Here we show an example of a data journey generated with the approach illustrated in the previous section. The data journey is illustrated in Figure 6.1 and 6.2, while Table 6.2 shows the content of the journey in detail. Each change in the LDH activity log is represented as a datanode (column DN). In column ‘Diff’, we report on the changes that occurred to the JSON object (or the original content if created). These can be a list of ‘updated’ or ‘deleted’ JSON properties. Each one of these datanodes is then mapped to activities of specific types, according to the Deep Viewpoints application. The very first action selects an artwork from the catalogue – Ilcruthach by Kevin Mooney. A stage is created but not properly populated yet (DN 3). Assigning a theme seems to be used as a way of reflecting on the type of prompts/questions that the designer may find interesting, in this example, the artwork is associated with ‘Nature’, ‘Fear’, ‘Vulnerability’, and ‘Uncomfortably Beautiful’. It can be seen how several datanodes are mapped to the same activity type (the theme assignment spans DN 4 to 7). After that, the user starts developing the actual questions (stages). The Add stage and Edit stage activities are repeated alternating each other (DN 8 to 12). The example shows how scripts are created and edited before they are assigned a name (DN 13). Moreover, the name itself is not necessarily explicative (“Ilcruthach 1”), maybe the images speak for themselves. After that, the user keeps refining the prompts (stages), including deleting failed attempts (DN 18). Stages include the artwork, to keep the visual content in sight (DN 22 to 24). When the design is completed, the script can be published and made available to other users (DN 25). Final steps include the association of the script to the exhibition and its display on

the main page. Finally, the script is run (DN 28) and responses collected. The user reviews the citizen generated content and accepts it for publishing (DN 29).

Table 6.2: Data journey showing the design and execution of a citizen curation script.

| DN | Diff | A Type | A | Comment |
|----|--|----------------|---|---|
| 1 | <code>{'type':'script','id':112,'name':'Untitled script','visible':false,'author':'Apollo Youth'}</code> | Create script | 1 | Creation of a new script. |
| 2 | <code>{update:{'artworkids':['63ffb7871cae1429486c79d7']}}</code> | Select artwork | 2 | Adding to the script the artwork Ilcruthach by Kevin Mooney, 2021 |
| 3 | <code>{update:{'stages':[{'type':'stage','id':1,'title':'','stagetype':'statement','includeartworks':[],'body':'Statement goes here']}}</code> | Add stage | 3 | Adding a statement stage to the script. |
| 4 | <code>{update:{'themeids':['627d1c6d2285e7418f56c3d3'],'stages':{0:{update:{'body':'' } } } }</code> | Edit themes | 4 | Adding the theme Nature to the script. |
| 5 | <code>{update:{'themeids':{insert:[(1,'62a9d15944c4e83e6b0874c6')] } } }</code> | Edit themes | 4 | Adding the theme Fear to the script. |
| 6 | <code>{update:{'themeids':{insert:[(2,'619e44b6472c910f00130523')] } } }</code> | Edit themes | 4 | Adding the theme Vulnerability to the script. |
| 7 | <code>{update:{'themeids':{insert:[(3,'623f185171a618214b53eac')] } } }</code> | Edit themes | 4 | Adding the theme Uncomfortably Beautiful to the script./ |
| 8 | <code>{update:{'stages':{0:{update:{'body':'This strange character evokes a number of emotions and ideas. These cannot always be represented through words, but rather an eerie feeling within. ' } } } }</code> | Edit stage | 5 | Updating the statement stage with an interpretation of the artwork. |
| 9 | <code>{update:{'stages':{insert:[(1,{'type':'stage','id':2,'title':'Question','stagetype':'question','includeartworks':[],'body':'Question goes here','question':{'type':'question','title':'Question goes here','choice':False,'options':[] } })] } }</code> | Add stage | 6 | Adding a new Question stage to the script. |
| 10 | <code>{update:{'stages':{1:{update:{'title':'Do you trust this bizarre character? ','question':{update:{'title':'' } } } } } }</code> | Edit stage | 7 | Adding the question text to the question stage. |

| | | | | |
|----|---|--------------|----|---|
| 11 | <pre>{update:{'stages':{insert:[(2,{'type':'stage','id':3,'title':'Questions','stagetype':'multiquestion','includeartworks':[],'body':['First question goes here','Second question goes here'],'questions':[{'type':'question','title':'First question goes here','choice':False,'options':[]},{'type':'question','title':'Second question goes here','choice':False,'options':[]}],'help':'Try as many of the following questions as you like.','sequential':True)}}]}</pre> | Add stage | 8 | Adding a new Multi-question stage to the script. |
| 12 | <pre>{update:{'stages':{2:{update:{'questions':{0:{update:{'title':'Are you threatened or comforted by its presence?'}},1:{update:{'title':'If threatened, why? '}}}}}}}</pre> | Edit stage | 9 | Adding the text of two questions to the Multi-question stage. |
| 13 | <pre>{update:{'name':'Ilcruthach 1'}}</pre> | Name script | 10 | Adding "Ilcruthach 1" as the name of the script. |
| 14 | <pre>{update:{'stages':{insert:[(3,{'type':'stage','id':4,'title':'Question','stagetype':'question','includeartworks':[],'body':'Question goes here','question':{'type':'question','title':'Question goes here','choice':False,'options':[]}})}}]}</pre> | Add stage | 11 | Adding a new Question stage to the script. |
| 15 | <pre>{update:{'stages':{3:{update:{'title':'','question':{update:{'title':'Translating Ilcruthach roughly to "wrongly created",do you think it is a fitting title? '}}}}}}}</pre> | Edit stage | 12 | Adding text to the new Question stage. |
| 16 | <pre>{update:{'stages':{insert:[(4,{'type':'stage','id':5,'title':'','stagetype':'statement','includeartworks':[],'body':'Statement goes here'})]}}}</pre> | Add stage | 13 | Adding a new statement stage to the script. |
| 17 | <pre>{update:{'stages':{0:{update:{'body':'This strange character evokes a number of emotions and ideas. These cannot always be represented through words,but rather an eerie feeling within. \nhttps://youtu.be/l2wj6lJtg_8'}}}}}</pre> | Edit stage | 14 | Updating the text of the original statement stage. |
| 18 | <pre>{update:{'stages':{delete:[4]}}}</pre> | Delete stage | 15 | Deleting the second statement stage added in step 12. |

| | | | | |
|----|--|--------------------|----|--|
| 19 | <code>{update: {'stages': {0: {update: {'body': 'This strange character evokes a number of emotions and ideas. These cannot always be represented through words, but rather an eerie feeling within. \nhttps://youtu.be/l2wj6lJtg_8\nhttps://m.youtube.com/watch?v=l2wj6lJtg_8&feature=youtu.be'}}}}}</code> | Edit stage | 16 | Go back to the first stage and edit it |
| 20 | <code>{update: {'stages': {0: {update: {'includeartworks': ['63ffb7871cae1429486c79d7']}}}}}</code> | Include artwork | 17 | Display the artwork in the first stage (a Statement stage) |
| 21 | <code>{update: {'stages': {0: {update: {'body': 'This strange character evokes a number of emotions and ideas. These cannot always be represented through words, but rather an eerie feeling within. \n\nhttps://m.youtube.com/watch?v=l2wj6lJtg_8&feature=youtu.be'}}}}}</code> | Edit stage | 18 | Updating the text of the original statement stage. |
| 22 | <code>{update: {'stages': {1: {update: {'includeartworks': ['63ffb7871cae1429486c79d7']}}}}}</code> | Include artwork | 19 | Display the artwork in the second stage (a Question stage) |
| 23 | <code>{update: {'stages': {2: {update: {'includeartworks': ['63ffb7871cae1429486c79d7']}}}}}</code> | Include artwork | 20 | Display the artwork in the third stage (a Multiquestion stage) |
| 24 | <code>{update: {'stages': {3: {update: {'includeartworks': ['63ffb7871cae1429486c79d7']}}}}}</code> | Include artwork | 21 | Display the artwork in the fourth stage (a Question stage) |
| 25 | <code>{update: {'visible': True, 'autoapproved': True}}</code> | Edit status | 22 | Set the script to visible and auto-moderation of responses |
| 26 | <code>{update: {'exhibitionids': ['64049d0333892124c85e50b0']}}}</code> | Include exhibition | 23 | Associate the script with the exhibition Kevin Mooney: Revenants |
| 27 | <code>{insert: {'homepageartworkid': '63ffb7871cae1429486c79d7'}}</code> | Include homepage | 24 | Set the artwork to display with the script on the homepage |
| 28 | Object created | Response | 25 | Response to script added |
| 29 | <code>{update: {'approved': True}}</code> | Approve response | 26 | Response approved to display on the website |

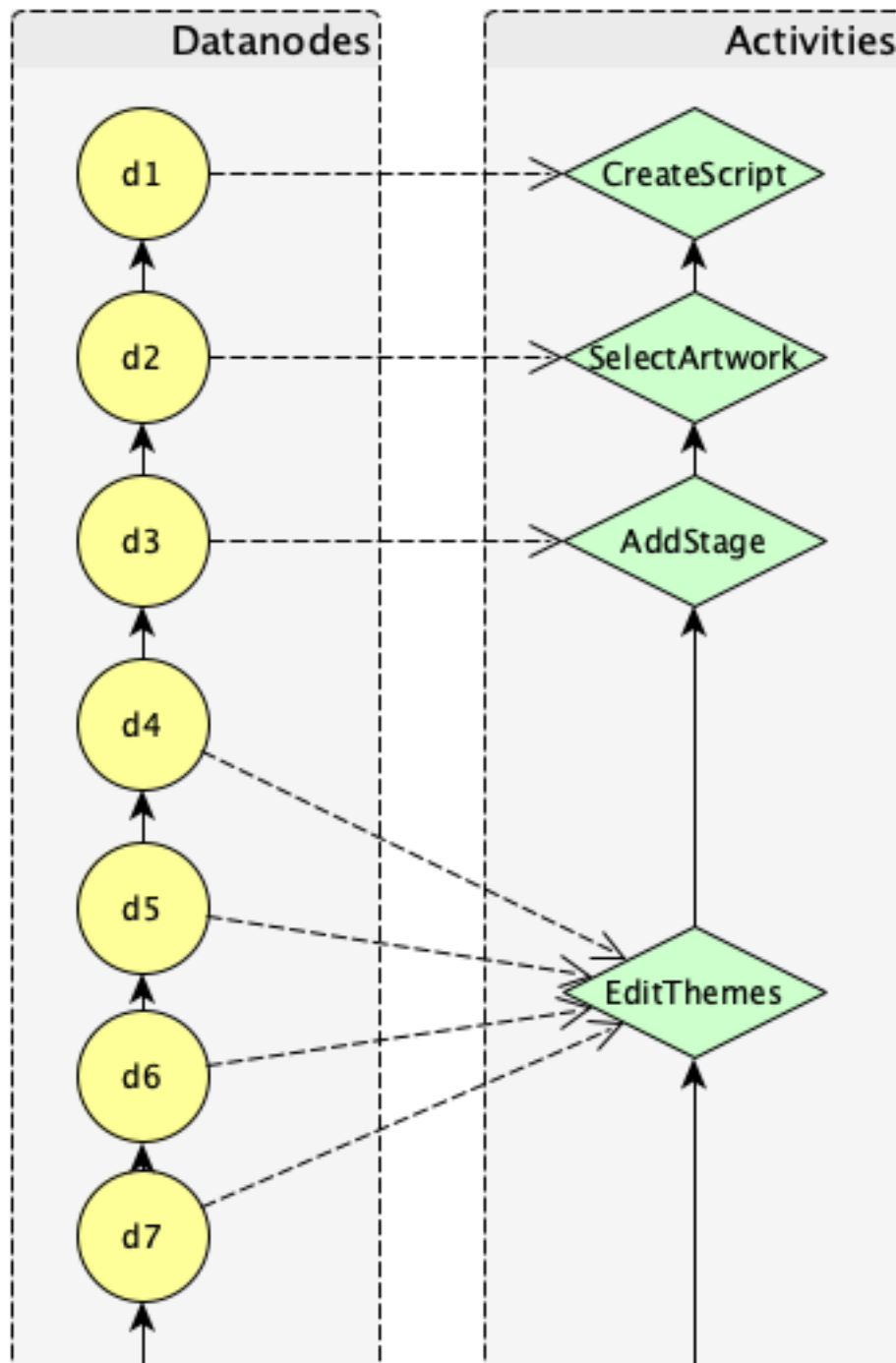


Figure 6.1: An example data journey: a script on Ilcruthach by Kevin Mooney. The image shows datanodes *d1* to *d7*. Note how the nodes are linked from the last to the first (reading from the bottom) in the spirit of provenance graphs.

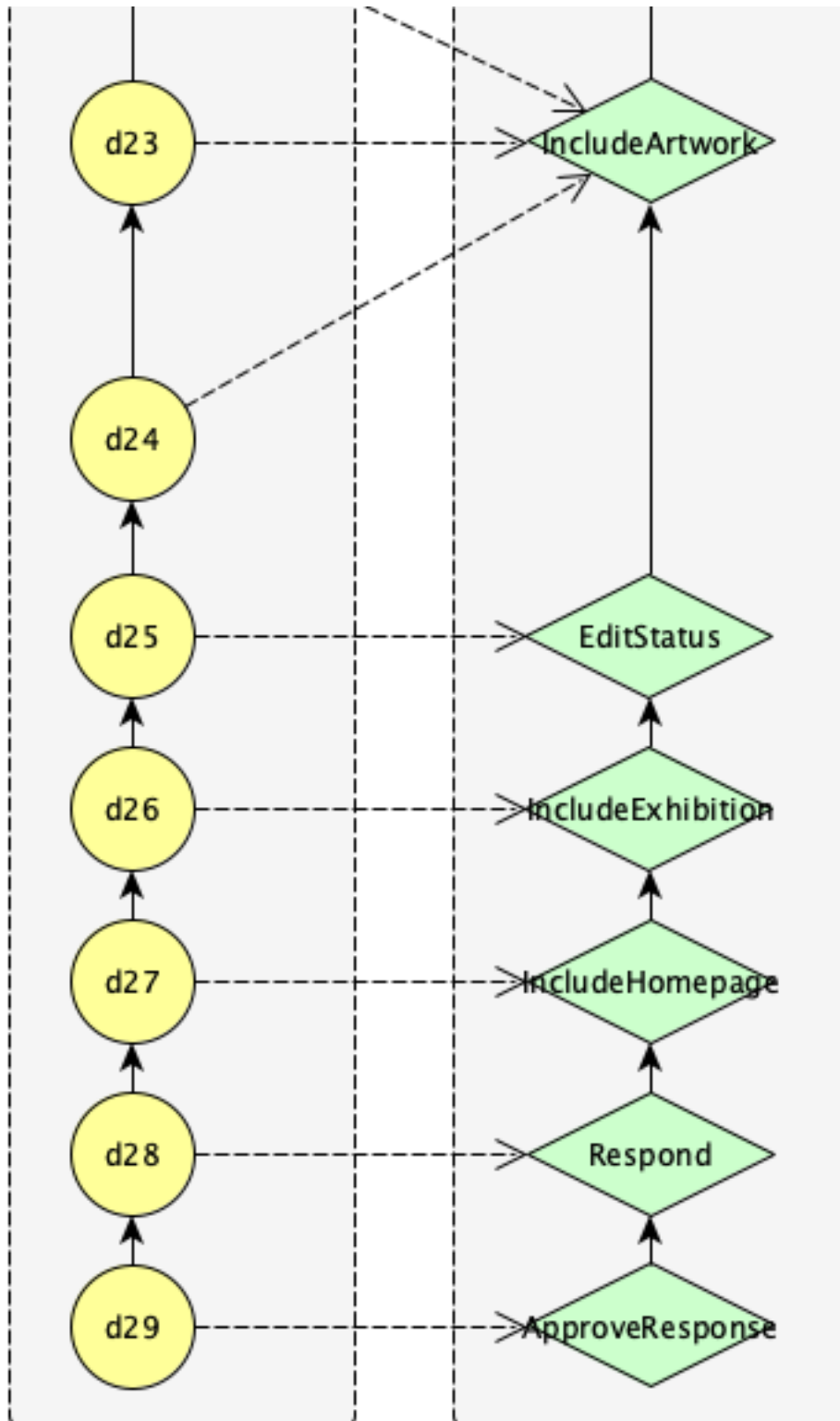


Figure 6.2: An example data journey: a script on Ilcruthach by Kevin Mooney. The image shows datanodes *d23* to *d29*. Note how the nodes are linked from the last to the first (reading from the bottom) in the spirit of provenance graphs.

7 Discussion and conclusions

In this deliverable, we presented the features of the SPICE linked data hub targeted to provenance and process analysis. We reviewed relevant literature on provenance and process analysis, focusing on methods for representing and reasoning on process traces in cultural heritage infrastructure, and highlighted the need for methods based on the notion of *Data Journey*. We illustrated the Deep Viewpoints system for scripting citizen curation in Chapter 3. To support provenance and process analysis, the SPICE LDH now includes an activity log that applications can query to analyse users' behaviour and derive insights on the process of citizen curation (Chapter 4), extending what was originally reported in D4.1 [33], D4.2 [20], and D4.7 [34]. In Chapter 5, we described the Data Journey Ontology (DJO), developed in SPICE, to represent and reason on complex data science pipelines, like the ones resulting from citizen curation. In Chapter 6 we have demonstrated the applicability of DJO to citizen curation data, thanks to the SPICE LDH activity log capability. We have shown how a data journey can be generated from activity logs using SPARQL Anything, a tool developed by SPICE to construct knowledge graphs from any format (JSON in the example). Finally, we illustrated an exemplary data journey from the Deep Viewpoints system (a script on IICruthach by Kevin Mooney). We observed how citizen curation involves a design process that can be complex, showing how tracing user behaviour during the design activity via data journeys can cause interesting insights into their process to emerge. The SPICE LDH offers functionalities that are domain-independent, meaning that it is left to application developers to decide how to represent their data. Although it is not possible to design a one-size-fits-all method for extracting data journeys for any application, we have established that this is possible by focusing on tracing user operations and analysing the changes in the content. Future work includes applying this technique to analyse other citizen curation applications, for example, considering the case study on cross-modal (image and audio) experiences in the EU-funded project Polifonia: a Digital Harmoniser of Musical Heritage Content.

Bibliography

- [1] S. Leonelli and N. Tempini, Eds., *Data Journeys in the Sciences*. Cham: Springer International Publishing, 2020. [Online]. Available: <http://link.springer.com/10.1007/978-3-030-37177-7>
- [2] “D4.5 provenance and process analysis layer: Requirements analysis,” *SPICE Project, European Union’s Horizon 2020 grant agreement No 870811*, vol. Deliverable 4.5, 2022.
- [3] “Stakeholders’ Survey on a European Collaborative Cloud for Cultural Heritage. Report on the online survey results,” European Commission Directorate-General for Research and Innovation Directorate D — People Unit D.3 — Fair Societies & Cultural Heritage, 2022.
- [4] M. Herschel, R. Diestelkämper, and H. B. Lahmar, “A survey on provenance: What for? what form? what from?” *The VLDB Journal*, vol. 26, no. 6, pp. 881–906, 2017.
- [5] L. Moreau, *The foundations for provenance on the web*. Now Publishers Inc, 2010.
- [6] L. Moreau and P. Groth, “PROV-overview,” W3C, W3C Note, Apr. 2013, <https://www.w3.org/TR/2013/NOTE-prov-overview-20130430/>.
- [7] E. Daga, M. d’Aquin, A. Gangemi, and E. Motta, “Describing semantic web applications through relations between data nodes,” *Technical Report kmi-14-05, Knowledge Media Institute, The Open University, Walton Hall, Milton Keynes*, 2014.
- [8] E. Daga, M. d’Aquin, A. Adamou, and E. Motta, “Addressing exploitability of smart city data,” in *2016 IEEE International Smart Cities Conference (ISC2)*. IEEE, 2016, pp. 1–6.
- [9] W. Oliveira, D. D. Oliveira, and V. Braganholo, “Provenance analytics for workflow-based computational experiments: A survey,” *ACM Comput. Surv.*, vol. 51, no. 3, May 2018. [Online]. Available: <https://doi.org/10.1145/3184900>
- [10] D. Garijo, P. Alper, K. Belhajjame, O. Corcho, Y. Gil, and C. Goble, “Common motifs in scientific workflows: An empirical analysis,” *Future Generation Computer Systems*, vol. 36, pp. 338–351, 2014.
- [11] F. Z. Khan, S. Soiland-Reyes, R. O. Sinnott, A. Lonie, C. Goble, and M. R. Crusoe, “Sharing interoperable workflow provenance: A review of best practices and their practical application in cwlprov,” *GigaScience*, vol. 8, no. 11, p. giz095, 2019.
- [12] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J. M. Gómez-Pérez, S. Bechhofer *et al.*, “Using a suite of ontologies for preserving workflow-centric research objects,” *Journal of Web Semantics*, vol. 32, pp. 16–42, 2015.
- [13] S. Soiland-Reyes, P. Sefton, M. Crosas, L. J. Castro, F. Coppens, J. M. Fernández, D. Garijo, B. A. Grüning, M. L. Rosa, S. Leo, E. Ó. Carragáin, M. Portier, A. Trisovic, R. Community, P. Groth, and C. A. Goble, “Packaging research artefacts with ro-crate,” *CoRR*, vol. abs/2108.06503, 2021. [Online]. Available: <https://arxiv.org/abs/2108.06503>
- [14] S. Soiland-Reyes, P. Sefton, M. Crosas, L. J. Castro, F. Coppens, J. M. Fernández, D. Garijo, B. Grüning, M. La Rosa, S. Leo, and *et al.*, “Packaging research artefacts with ro-crate,” *Data Science*, p. 1–42, Jan. 2022. [Online]. Available: <https://doi.org/10.3233/DS-210053>
- [15] E. Daga, L. Asprino, R. Damiano, M. Daquino, B. D. Agudo, A. Gangemi, T. Kuflik, A. Lieto, M. Maguire, A. M. Marras *et al.*, “Integrating citizen experiences in cultural heritage archives: requirements, state of the art, and challenges,” *ACM Journal on Computing and Cultural Heritage (JOCCH)*, vol. 15, no. 1, pp. 1–35, 2022.
- [16] D. Oldman and D. Tanase, “Reshaping the knowledge graph by connecting researchers, data and practices in researchspace,” in *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part II*, ser. Lecture Notes in Computer Science, D. Vrandečić,

- K. Bontcheva, M. C. Suárez-Figueroa, V. Presutti, I. Celino, M. Sabou, L. Kaffee, and E. Simperl, Eds., vol. 11137. Springer, 2018, pp. 325–340. [Online]. Available: https://doi.org/10.1007/978-3-030-00668-6_20
- [17] A. Stoneman, J. Carvalho, E. Daga, M. Maguire, and P. Mulholland, “Uncomfortable revelations: can citizen curation widen access to museums?” *Museum Ireland*, vol. 28, pp. 64–71, 2021.
- [18] “Curation scripting support,” *SPICE Project, European Union’s Horizon 2020 grant agreement No 870811*, vol. Deliverable 6.7, 2022.
- [19] L. Asprino, E. Daga, A. Gangemi, and P. Mulholland, “Knowledge graph construction with a façade: a unified method to access heterogeneous data sources on the web,” *ACM Transactions on Internet Technology*, 2022.
- [20] “Linked data server technology: integrating feedback from use case requirements,” *SPICE Project, European Union’s Horizon 2020 grant agreement No 870811*, vol. Deliverable 4.2, 2022.
- [21] “Linked data server technology: Final release and open-source distribution,” *SPICE Project, European Union’s Horizon 2020 grant agreement No 870811*, vol. Deliverable 6.8, 2023.
- [22] L. Moreau, P. Groth, S. Miles, J. Vazquez-Salceda, J. Ibbotson, S. Jiang, S. Munroe, O. Rana, A. Schreiber, V. Tan *et al.*, “The provenance of electronic data,” *Communications of the ACM*, vol. 51, no. 4, pp. 52–58, 2008.
- [23] E. Daga, M. d’Aquin, and E. Motta, “Propagating data policies: a user study,” in *Proceedings of the Knowledge Capture Conference*, 2017, pp. 1–8.
- [24] A. Newell, “The knowledge level,” *Artificial intelligence*, vol. 18, no. 1, pp. 87–127, 1982.
- [25] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, and J. Zhao, “Prov-o: The prov ontology,” 2013.
- [26] R. Liepinš, M. Grasmanis, and U. Bojars, “Owlgred ontology visualizer,” in *Proceedings of the 2014 International Conference on Developers*, vol. 1268. CEUR-WS. org, 2014, pp. 37–42.
- [27] E. Daga, E. Blomqvist, A. Gangemi, E. Montiel, N. Nikitina, V. Presutti, and B. Villazón-Terrazas, “D2.5.2 pattern based ontology design: methodology and software support,” 2008.
- [28] V. Presutti, E. Blomqvist, E. Daga, and A. Gangemi, “Pattern-based ontology design,” in *Ontology Engineering in a Networked World*. Springer, 2012, pp. 35–64.
- [29] E. Daga, A. Meroño-Peñuela, and E. Motta, “Sequential linked data: The state of affairs,” *Semantic Web*, no. Preprint, pp. 1–36, 2021.
- [30] C. M. Keet, A. Ławrynowicz, C. d’Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, and M. Hilario, “The data mining optimization ontology,” *Journal of web semantics*, vol. 32, pp. 43–53, 2015.
- [31] G. C. Publio, D. Esteves, A. Ławrynowicz, P. Panov, L. Soldatova, T. Soru, J. Vanschoren, and H. Zafar, “MI-schema: exposing the semantics of machine learning with schemas and ontologies,” *arXiv preprint arXiv:1807.05351*, 2018.
- [32] E. Daga, L. Asprino, P. Mulholland, and A. Gangemi, “Facade-X: an opinionated approach to SPARQL anything,” in *Proceedings of the 17th International Conference on Semantic Systems, 6-9 September 2021, Amsterdam, The Netherlands*, vol. 53. IOS Press, 2021, pp. 58–73.
- [33] E. Daga, P. Mulholland, J. Carvalho, R. Damiano, M. Daquino, B. D. Agudo, T. Kuflik, A. Lieto, and A. Bosca, “Linked data server technology: requirements and initial prototype,” *SPICE Project, European Union’s Horizon 2020 grant agreement No 870811*, vol. Deliverable 4.1, 2021.
- [34] “Linked data server technology: Final release and open-source distribution,” *SPICE Project, European Union’s Horizon 2020 grant agreement No 870811*, vol. Deliverable 4.7, 2023.