Explainability-by-Design: A Methodology to Support Explanations in Decision-Making Systems

Trung Dong Huynh, Niko Tsakalakis, Ayah Helal, Sophie Stalla-Bourdillon, and Luc Moreau

Abstract—Algorithms play a key role nowadays in many technological systems that control or affect various aspects of our lives. As a result, providing explanations to address the needs of users and organisations is increasingly expected by the laws and regulations, codes of conduct, and the public. However, as laws and regulations do not prescribe how to meet such expectations, organisations are often left to devise their own approaches to explainability, inevitably increasing the cost of compliance and good governance. Hence, we put forth "*Explainability by Design*", a holistic methodology characterised by proactive measures to include explanation capability in the design of decision-making systems. This paper describes the technical steps of the Explainability-by-Design methodology in a software engineering workflow to implement explanation capability from requirements elicited by domain experts for a specific application context. Outputs of the Explainability-by-Design methodology are a set of configurations, allowing a reusable service, called the Explanation Assistant, to exploit logs provided by applications and create provenance traces that can be queried to extract relevant data points, which in turn can be used in explanation plans to construct explanations personalised to their consumers. Following those steps, organisations will be able to design their decision-making systems to produce explanations that meet the specified requirements, be it from laws, regulations, or business needs. We apply the methodology to two applications, resulting in a deployment of the Explanation Assistant demonstrating explanations capabilities. Finally, the associated development costs are measured, showing that the approach to construct explanations is tractable in terms of development time, which can be as low as two hours per explanation sentence.

Index Terms—explainability by design, automated decision-making, data provenance.

1 Introduction

The 1990s saw a movement of privacy-enhancing technologies [1] aimed to address increasing requirements for data privacy and, by extension, data protection. This led to the concept of "Privacy by Design" and "Privacy by Default", which has been now adopted by key legal frameworks [2]. In the same vein, the requirements for explainability to address the needs of users and organisations are now being recognised widely. Increasingly, laws, regulations, and codes of conduct put a duty on organisations that make automated decisions about their customers or users to provide them with an explanation of the 'logic' associated with such decisions. This has been dubbed the 'right to explanation' [3]. However, there is still relatively little methodological steps that can be followed in order to build systems that offer the level of explanations required [4]. Besides, while the production of explanations is usually tied to a limited set of rights (i.e., the right to information, to access data, or not to be subject to solely automated decision making), the potential of explanations conceived as detective controls goes beyond the exercise of these three individual rights [5]. In this work, we put forward "*Explainability by Design*", a methodological approach which tackles this issue in the design and architecture of IT systems and business practices. Here, Explainability-by-Design (EbD) is a methodology that is characterised by proactive measures to include explanations in the design rather than relying on reactive measures to bolt on explanation capability after the fact, as an add-on. According to this view, explainability is integral to the system, enriching its functionality, with capabilities that can address regulatory requirements but also functional and business needs. The methodology consists of a set of technological steps ingesting explanation requirements in order to construct an explanation capability that system's triggers can activate, in order to generate the required explanations. As a result, the system's behaviour is said to be explainable by design.

Explanations for a decision, however, are not sought by end users or regulators for their own sake but as the means to evaluate the basis of decision-making against broader principles, including fairness and the right to due process. Explanations, therefore, must be sufficiently meaningful to provide all stakeholders with a better understanding of how a decision was made and, specifically to end users, to enable them to exercise their legal rights [6]. While the recent plethora of research in Explainable Artificial Intelligence (XAI) [7] has been focusing mostly on explaining the inner workings of machine learning (ML) models employed in decision-making systems, they are only one part of wider pipelines and processes that influence the decision making. In order to be sufficiently meaningful to users and organisations, explanations cannot just focus on the 'black box' but also have to provide information about events that have influenced the outcome of a decision before and after the application of the algorithmic models [8]. Such information must comprise the key data used to arrive at a decision, their origins, the actions of the organisations and their staff involved in the decision-making process, including designing/setting up the systems making deci-

sions. Nowadays, with the common use of ML models, explaining a decision pipeline surrounding a model involves, for instance, understanding how the model was derived, which datasets were involved, their origin, the governance that was in place when datasets were selected and the model was trained, etc. [9] Thus, "a record that describes the people, institutions, entities, and activities involved in producing, influencing or delivering [10]" a decision is a valuable source of data from which to generate explanations for the decision. This is precisely the definition of W3C PROV provenance [10], a standardised form of knowledge graph providing an audit trail of what a system performed. It includes references to people, data sets and organisations involved in making decisions; attribution of data; and data derivation. It captures not only how data is used and updated but also how data flows in the system and their causal dependencies (this type of information is also referred to as causal attribution [11]). We show here that tracking the provenance of decisions can assist with making complex algorithmic systems transparent and accountable [12]. This allows us to tackle a broad category of explanations, usually referred to as *ex post* explanations, which are constructed after the corresponding decision is made and have the ability to refer to the data and the specific processing of that decision.

Against this background, this article describes the technical steps in the EbD methodology. They are to be carried out by a data engineer within a socio-technical context in which requirements are elicited by means of an explanation requirement analysis conducted by domain experts, whereas the explanations are evaluated by application stakeholders. Following those steps, organisations will be able to enrich their decision-making systems to produce ex-post explanations that are tailored to an individual decision from the provenance of the decision. By so doing, the explanations are specific to the circumstances pertaining to the decision and targeted to their recipient. In particular, the contributions of this article are as follows:

- A detailed description of the Explanation Technical Design phase of the Explanation-by-Design methodology, a software engineering workflow that takes the requirements for explainability as input and produces a stand-alone service able to receive application data and generate explanations.
- PLEAD Explanation Assistant service[1], a reusable software service taking the artefacts resulting from the above methodology, and demonstrating its ability to produce explanations for two different application scenarios.
- Cost metrics measured from the enactment of the methodology over two applications, showing that the approach is tractable in terms of development time.

In the remainder of the article, we discuss the related work in Sec. 2 and present the two application scenarios on which the EbD methodology was executed to generate explanations in Sec. 3. We then overview the EbD methodology in Sec. 4 and describe the tasks comprising its Explanation Technical Design phase in Sec. 5. The reference implementation of the methodology is reported in Sec 6. In Sec. 7, we evaluate the development costs of implementing the methodology over the two application scenarios. Finally, we conclude the article with extension points for the methodology in Sec. 8.

1. Available online at https://explain.openprovenance.org.

2 Related Work

With the prevalent adoption of automated and machine-assisted decision-making systems, the public's attention on explanations of their decisions has become widespread [13]. Recognising this, in Europe, the GDPR[2] includes a right to access to "meaningful information about the logic involved, as well as the significance and the envisaged consequences of such processing" of personal data [14, Article 15]. Similar provisions are included in the French Digital Republic Act and the Modernization of the Convention 108 (see [15] for a comparison). A key challenge with automated decision-making systems, however, lies in the technical implementation of such obligations given their typical complexity. The academic community has also recognized the importance of tackling the above explainability concerns, which have become an active research topic world-wide. In the US, the Defense Advanced Research Projects Agency's eXplainable AI program[3] specifically focuses on the explainability of ML-based approaches and quality metrics of these explanations. A number of international events are specifically dedicated to this topic: ACM Conference on Fairness, Accountability, and Transparency (ACM FAccT), the Explainable AI workshop at International Joint Conferences on Artificial Intelligence (IJCAI); the Fairness, Accountability, and Transparency in Machine Learning (FAT/ML) workshop.[4]

To combat the opaqueness of ML models, a variety of techniques have been proposed (see [16] for an overview and [7], [17] for extensive surveys). Approaches include designing the learning process to ensure interpretability of results, approximating a learned model in a more readily intelligible substitution; and offering tools to interact with the model to get a sense for its operation. In software engineering, ML techniques are also being applied to improve software development; and in some cases, explanations are offered to help the developers understand why a given recommendation is made [18]. For instance, neural networks are employed to identify self-admitted technical debts from code comments and terms picked by the model are shown to help explain the intuition behind the model's decisions [19]. In another instance, explanations are offered to provide explanations for refactoring solutions to help users understand how the solutions are generated [20]. However, all the above work has been focusing mostly on explaining the inner workings of the algorithms employed in decision-making systems and often fails to account for influential events outside the black box [21]. Organisations, for instance, might not be fully aware of the exact practices that generate the training data upon which ML models are built [22].

In contrast, Explainability-by-Design builds on several concepts analogue to the seven foundational principles of Privacy by Design [23] in a holistic approach.

2. The General Data Protection Regulation enacted by the European Union [14] and later adopted in domestic law by the United Kingdom.
3. www.darpa.mil/program/explainable-artificial-intelligence.
4. ACM FAccT conference: https://facctconference.org. FAT/ML workshop: https://www.fatml.org.

- **Proactive**: EbD does not wait for a request for explanations to investigate forensically how a decision was reached.
- **Explainability by default**: EbD seeks to deliver the maximum degree of explainability, ensuring that all decisions are potentially explainable, without requiring any action from the user.
- **Explainability embedded in the design**: EbD is not bolted on as an add-on but is embedded in the design of systems, so that it becomes an essential component delivering functionality for the application.
- **End-to-end Explainability**: EbD is applied to all flows of information, during the entire lifecycle of the data involved.
- **Visibility and Transparency**: EbD seeks to assure all stakeholders are provided with explanations addressing their needs. Explanations themselves can be audited and potentially independently verified.

This work follows a sequence of methodological approaches for building systems with provenance capabilities. PRIME, a provenance incorporation methodology [24], was defined as a stand-alone methodology to elicit provenance requirements and design applications that surface the appropriate flows, so that they can be exported in a provenance description language (which preceded the PROV Data Model [10]). The Provenance Template [25] approach allows provenance to be specified declaratively, using a template with placeholders composed at design-time, to be instantiated with runtime values. In the reference implementation of the EbD methodology (Sec. 6), provenance templates were adopted to assist with the Provenance Modelling task. UML2PROV [26] conceptually combined provenance templates with the Unified Modelling Language (UML): it is an adjunct methodology to UML, able to leverage suitably annotated UML diagrams in order to record runtime values required to instantiate provenance templates constructed from such diagrams. All the three methodological approaches, PRIME, Provenance Templates, and UML2PROV, are intended to address a broad set of requirements for provenance [27]. Here, EbD can be seen as a methodology focusing on a specific subset of requirements, namely those requiring applications to be explainable. It addresses specifically the explainability and transparency requirements of software systems, one of the key software quality characteristics desired in AI-based systems [28].

The methodology presented in this article is part of the wider work of the PLEAD project.[5] Previously, provenance of loan decisions was shown to be able to support various GDPR-related explanations for such decisions [29]. Such explanations are demonstrated to be effective means as both internal detective controls (to benefit data controllers) and external detective controls (to benefit data subjects and regulators), helping organisations meet their accountability and data protection-by-design obligations [5]. This article describes the Explanation Technical Design phase (B) of EbD in details but refers to only a few technically relevant dimensions of Explanations Requirements. The complete details on the various characteristics of an explanation requirement are presented in a separate article [30].

5. See https://plead-project.org.

3 Application Scenarios

Before discussing the EbD methodology, it is useful to have a couple of concrete application scenarios to provide examples to illustrate the methodology's steps.

3.1 Credit Card Application

Credit card applications nowadays are typically assessed by automated systems, taking into account the applicant's credit history provided by a credit referencing agency (CRA). Such applications are often approved or rejected within seconds, without human intervention. In this scenario, an applicant applies for a new credit card with a bank. The bank employs a CRA who searches the applicant's credit history and provides a credit report. Such reports typically contain a credit score, representing the applicant's overall creditworthiness, in addition to a variety of credit-related records: past payments (to existing credit agreements), addresses, electoral roll, and so on. Assuming that an application is rejected by the bank, the applicant is entitled to some explanations for the bank decisions according to the prevailing regulations.

3.2 School Allocation

Every year, over a million of parents apply for school places in primary and secondary schools for their children in the UK. Semi-automated school application systems are employed by local education authorities to help manage the great number of applications, match parents' school preferences against a wide variety of ranking criteria set by individual schools, and allocate available school places according to such criteria. School allocation decisions are communicated to the parents, informing which school their children are to be admitted to, with little information on the individual circumstances of a child that influenced the decision. Understandably, parents who do not get their first choices are keen to understand the reason why, typically resulting in a huge volume of calls and inquiries from disappointed parents following the national offer day. Against this background, producing decision letters that are tailored to the individual circumstances of a child's application will help parents understand better why their children are allocated to a particular school (and why not to their first choice). At the same time, such targetted explanations should reduce enquiries on their allocation decisions to local authorities, and would also assist operators at call centres with providing targeted answers.

4 Explainability-by-Design Methodology

In the context of this work, an *explanation* is a statement providing details or reasons that relate to or underlie a decision in order to enable the recipient of the statement: (i) to better-understand the decision and/or (a particular aspect of) the decision-making process (e.g. its fairness, accuracy); and where necessary (ii) to take action (e.g. contest a decision, correct the decision-making process). A *decision* can be any output of a data processing pipeline, produced after consideration of one or more data points. The output is often followed by an action to bring upon a resolution to a situation. The EbD methodology (Fig. 1) is a socio-

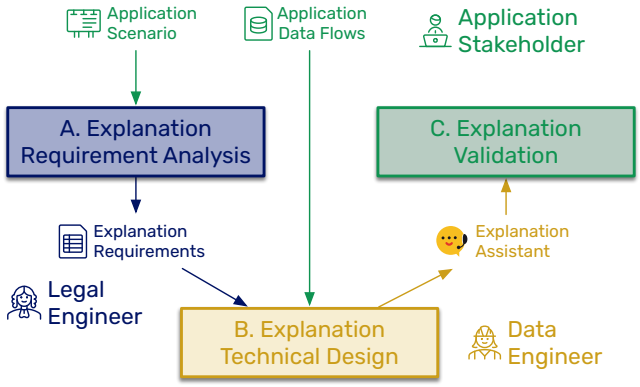

Fig. 1. Overview of the Explainability-by-Design methodology.

technical process, involving application stakeholders, domain experts, and data engineers. It provides organisations with a clear recipe to build explanation capability into their decision-making systems while addressing regulatory and business requirements, such as those outlined in the presented application scenarios. To elicit the requirements for explanations in these two scenarios, within the scope of this work, a legal engineer acts as a domain expert analysing the applicable regulatory obligations for explanations. Therefore, we refer to the legal engineer interchangeably with the domain expert in the texts henceforth. Alternatively, business management, policy makers, or sociologists could also be included in the role of the domain expert.

The EbD methodology comprises three main phases:

A. **Explanation Requirement Analysis**: Starting from an application scenario provided by its stakeholders, the requirements of explanations for the application are captured by the legal engineer analysing the prevailing regulations applicable to the scenario. This phase aims to enumerate and characterise all the explanations as mandated by regulations or business needs applicable to decisions made by the application. Collectively, the output from this phase is called the *socio-technical specification of explanations* for the chosen application; it provides a list of explanation requirements to support. Those requirements specify the characteristics of the explanations, including, for instance, their goals, the minimum required content, when it should be presented and to whom. For brevity, the whole specification is referred to as the Explanations Requirements.

B. **Explanation Technical Design** (ETD): This phase consists of a set of technological steps, ingesting explanation requirements in order to construct an explanation capability that a system's triggers can activate, in order to generate the required explanations. At a high level, the ETD phase produces an instantiated Explanation Assistant service that generates explanations according to the Explanations Requirements provided by the Explanation Requirement Analysis phase.

C. **Validation**: The explanations generated by the Explanation Assistant are evaluated by the application stakeholders to determine their suitability for the intended audience and their effectiveness with respect to the explanation goals specified in the first phase.

This article focusses on the Explanation Technical Design phase (B) and describes it in full in Sec. 5. The complete socio-technical context, involving the requirement analysis (A) and explanation validation (C), is beyond the scope of the paper, as it may involve completely different contexts, e.g. legal, business, robotic, etc. It is, however, assumed that the explanation requirement analysis (A) has been done and the resulted Explanations Requirements are available before the ETD phase (B) starts. The Explanations Requirements should enumerate all the explanations to be supported and, for each of them, describe the following characteristics (amongst others, see [30] for the details):

- **Minimum required content**: the information required to be present in the explanation in order for it to satisfy the explanation's mandate.
- **Intended audience**: for whom the explanation is intended (e.g., end-users, customer service agents).
- **Exemplar narrative**: text sentences crafted by the legal engineer for a specific explanation.

As an illustration, in the context of the credit card application scenario (Sec. 3.1), the legal engineer determines that the GDPR applies to decisions made by the bank's system because it processes personal data. Amongst the explanations mandated by the GDPR, an explanation requirement with respect to the data subject's right to obtain an explanation about the decision reached, for example, has the following attributes:

- Minimum required content: explanation of automated decision, the data used, the main points in the decision making process.
- Intended audience: the data subject (i.e., the applicant).
- Exemplar narrative: "We made this decision with an automated scoring system that takes into account the information provided in your application as well as a credit score produced by a credit referencing agency."

5 Explanation Technical Design Phase

This section describes the tasks that constitute the ETD phase. From the Explanation Requirements provided by the previous phase, the data engineer (1) models the audit trails in the application, (2) creates queries to extract data from the application's audit trails as required by a particular explanation, (3) builds explanation plans to render the queried data in narratives that are suitable for end-user consumption, and (4) deploys the Explanation Assistant service for the specified application (see Fig. 2). This phase produces the Provenance Patterns (to record audit trails), the Provenance Queries, and the Explanation Plans to support all the required explanations. They are then all packaged into a configuration to instantiate an Explanation Assistant service that produces explanations (as specified by the Explanation Requirements) from the application's data.

5.1 Task 1: Modelling the Provenance of a Decision

This step provides the means for the application to record audit trails of its decisions, enabling us to trace back a decision to its input data and to identify the responsibility for each of the activities that led to the decision. Such an

TABLE 1
Glossary of the Artefacts generated or used by the Explanation Technical Design phase.

Artefact	Description
Application Scenario	A document describing the application, how it works, its context, its users, and decisions that it makes.
Data Flows	Diagrams or descriptions of the conceptual flows of data within the specified application, including some sample decisions and their associated application data.
Explanation Requirements	The socio-technical specification of explanations for the specified application, which enumerates all the explanations mandated either by applicable regulations or by business needs and their characteristics.
Explanation Assistant	A software service that provides explanation generation capability for the application for which it is configured.
Provenance Patterns	The set of patterns describing the shape of the provenance to be recorded for a specified application.
Provenance Traces	Full provenance traces of a number of sample decisions following the specified Provenance Patterns.
Provenance Queries	The set of queries to extract the relevant data from a decision's provenance to support explanation generation.
Explanation Plans	The syntax trees specifying the explanation narratives to be produced by the natural language generation engine.
Explanation Dictionary	A collection of application-specific linguistic configurations, including profiles of the supported target audience.

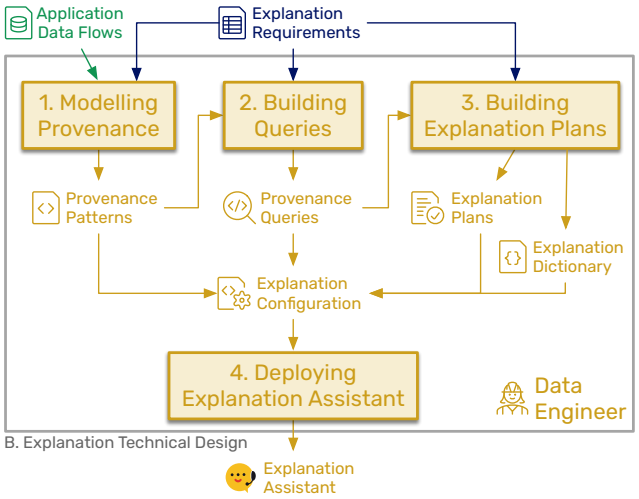

Fig. 2. The tasks in the Explanation Technical Design phase.

audit trail is also known as the *provenance of the decision*. Paraphrasing the World Wide Web Consortium's definition of provenance [10], we define the provenance of a decision as "a record that describes the people, institutions, entities, and activities involved in producing, influencing, or delivering" that decision. Such records provide the *history* of the decision and are a valuable source of data from which to generate explanations about what happened that led to the decision. In order to do so, however, the provenance of the decision must contain sufficient details to support all the specified explanations, i.e. providing all their minimum required content. In addition, it must satisfy the following *provenance requirements* to support explanation generation [29]. The provenance must allow for:

PR1 describing the various types of data of the universe of discourse, e.g. an application, an applicant, data, data providers, a decision, and so on;
PR2 tracing back outcomes to their influencers;
PR3 attributing or assigning responsibility to software systems or humans for actions or outcomes; and
PR4 describing the various activities, their respective timing, and their contribution to outcomes.

Any provenance data model (e.g. PROV Data Model [10], Open Provenance Model [31]) or, indeed, knowledge graph (such as RDF graphs [32]) that supports the above requirements can be employed to model the provenance of a decision in the target application. With a chosen data model, the provenance trace of a decision can then be modelled following a modelling methodology such as PrIMe [24]. However, to support explanation generation via queries (Task 2), in addition to the above provenance requirements, the provenance traces to be recorded must follow some predicted patterns in order to allow for information to be retrieved from them. Hence, at the minimum, this task needs to document the expected Provenance Patterns and share them with Task 2 to facilitate the construction of provenance queries there. Such patterns can be described using the Shapes Constraint Language (SHACL) [33], if the provenance is recorded as RDF graphs, or PROV-Template [25], if the PROV Data Model is used.

In Task 1, the Data Engineer inspects the Data Flows provided by the Application Stakeholder to locate in the flows the data entities required to express the minimum content for *all* explanations (specified by the Explanation Requirements) (**PR1**, Step 1.1). From those data entities and in reference to the Data Flows, in Step 1.2, the Data Engineer determines how they contribute to a decision by the application; and in many cases, this also involves identifying intermediary data entities and activities in the flows and how they are linked to the decisions (**PR2**). Next, the activities that either use or generate the above data entities are identified (**PR4**). At the same time, the Data Engineer identifies who or what is responsible for those data entities and activities (**PR3**). The knowledge is then captured in a set of Provenance Patterns, typically one for each identified activity, to model the interactions between the influencers and the outcomes of that activity (Step 1.3). Finally, using some sample decisions and sample application data, the Data Engineer instantiates the Provenance Patterns to produce a set of Provenance Traces for those decisions (Step 1.4). This step aims to make sure that the resulting Provenance Traces satisfy all the provenance requirements (including that the audit trails are fully connected) and contain the minimum information content required for all the specified explanations.

5.2 Task 2: Building Queries

An explanation is specific to a particular decision made by the application and it must include the data, activities,

TABLE 2
Task 1: Modelling Provenance

What: Constructing the provenance patterns to support tracking all data entities, activities, people, organisations, and systems that influence and lead to a decision.
Inputs: Explanation Requirements, Data Flows.
Outputs: Provenance Patterns, Provenance Traces.

1.1 Inspect the application's Data Flows and locate the minimum required content for all the specified explanations.
1.2 Identify how the data entities identified in Step 1.1 lead to the decision, the possible intermediary data entities and/or activities that connect them, and who/what are responsible for all those (i.e., the agents).
1.3 Capture all the above provenance elements and how one influences another in a set of Provenance Patterns.
1.4 Instantiate the Provenance Patterns with sample application data to ensure that the resulted sample Provenance Traces satisfies the provenance requirements.

TABLE 3
Task 2: Building Queries

What: Constructing provenance queries for each explanation to extract the data to be included in the explanation's narrative.
Inputs: Explanation Requirements, Provenance Traces.
Outputs: Provenance Queries.

For each explanation requirement:
2.1 Locate the data elements required by an explanation's exemplar narratives in the sample provenance traces.
2.2 Construct a generic graph pattern that link a decision to all the identified data elements using a graph query language.

and responsibilities specific to that decision making to be personalised. Thanks to the Provenance Modelling task, all such information should be included in the provenance of a decision. However, the provenance itself is *not* an explanation for the decision and is also typically unsuitable to be presented to end users in its original form (e.g. an RDF graph or a PROV serialisation). The information or data required by an explanation needs to be extracted from the decision's provenance to construct an explanation that is targeted to the intended recipient (and in a form that is easy to digest). As the provenance (of a decision) can be represented as a graph, the required information, contained within such a graph, can be found therein by tracing back from the decision in question (**PR2**).

In Task 2, for *each* explanation specified in the Explanation Requirements, the Data Engineer examines the explanation's exemplar narrative and highlights the data elements therein that are specific to a particular decision. Those data elements are then identified in the sample Provenance Traces produced as a result of Task 1 from the Provenance Patterns (Step 1.4). Based on where the required data elements are found in the Provenance Traces, the Data Engineer constructs a generic graph pattern to describe how those elements are linked to the decision and, hence, can later be found in the decision's provenance trace (Step 2.2). For this purpose, a graph query language (e.g. SPARQL [34], Cypher [35]) is needed to describe the graph pattern. The language must allow a query to select over provenance constructs, joining them via provenance relations, and filtering by their attributes (e.g., data types, time, values) to identify the data required for the explanation.

TABLE 4
Task 3: Building Explanation Plans

What: Constructing the syntax tree for each explanation, defining the explanation's narrative to be generated.
Inputs: Explanation Requirements, Provenance Queries.
Outputs: Explanation Plans, Explanation Dictionary.

For each explanation requirement:
3.1 Construct the NLG syntax tree from the exemplar narrative of the explanation.
3.2 Replace the narrative's data elements with the corresponding variables from the provenance query for the explanation.
3.3 Specify the linguistic elements specific to each profile to be supported in the Explanation Dictionary.

TABLE 5
Task 4: Deploying the Explanation Assistant

What: Instantiating an Explanation Assistant service configured specifically to support the application's explanation requirements.
Inputs: Provenance Patterns, Provenance Queries, Explanation Plans, Explanation Dictionary, Data Flows.
Outputs: Explanation Assistant, Sample Explanations.

4.1 Configure the Explanation Assistant service with the Provenance Patterns, Provenance Queries, Explanation Plans, and Explanation Dictionary produced in the previous tasks.
4.2 Test the instantiated Explanation Assistant with sample data from the Data Flows to produce Sample Explanations.

5.3 Task 3: Building Explanation Plans

To construct an explanation that is easy to digest for end users, in this task, the Data Engineer builds explanation plans to dictate how a narrative is generated from the data elements retrieved for an explanation by its Provenance Queries. An explanation plan is a linguistic syntax tree, drawing its foundation from Natural Language Generation (NLG) pipelines [36], created for each sentence provided in an explanation's exemplar narrative (as specified by the Explanation Requirements). Such a sentence typically mentions data or information specific to a decision within otherwise static texts. For example, in the explanation about a credit card decision, the text "... your *credit score* produced by *credit referencing agency*..." contains 'credit score' and 'credit referencing agency', which are the data elements that can be queried from a decision's provenance, while the surrounding texts are unchanged with respect to application data from one decision to another. Hence, the syntax tree in an explanation plan does not only contain linguistic constituents but also references to variables to be bound to the results of the corresponding provenance query.

Explanations containing the same information may be presented to different audiences. For instance, the explana-

tion that is included in the web page viewed by a credit card applicant may also be provided to a bank's support team when responding to the same applicant's enquiry over the phone. To support possible different profiles of usage, we allow for the definition of *profiles* in an Explanation Dictionary to be shared by all the explanation plans constructed for a specific application. For example, from the same explanation plan, a 'customer support' profile may generate the phrase "... *the borrower's* credit score produced by credit referencing agency..." instead of "... *your* credit score" (as addressed to the applicant). The linguistic syntax tree in an explanation plan can then be later configured by a profile defined in the Explanation Dictionary.

In summary, in Task 3, for *each* sentence in an explanation's exemplar narrative, the Data Engineer constructs a linguistic syntax tree (Step 3.1). The data elements specific to a particular decision in the syntax tree are then replaced by references to variables provided from the corresponding Provenance Queries (Step 3.2). For each profile of usage to be supported, relevant linguistic elements are to be defined in the application's Explanation Dictionary and referred to in the syntax tree (Step 3.3). The outputs of this task are, hence, a set of Explanation Plans for all the required explanations and the associated Explanation Dictionary. During the execution of an explanation plan, values from the corresponding provenance query (run over the provenance of a decision) are to replace variable references in its syntax tree. The 'instantiated' syntax tree will then be processed by a surface natural language generation (NLG) engine [36] (such as SimpleNLG [37]) to generate the final sentence to be presented to the target audience.

5.4 Task 4: Deploying Explanation Assistant

All the artefacts produced by the three previous tasks, i.e. Provenance Patterns, Provenance Queries, Explanation Plans, and Explanation Dictionary, are bundled together to configure a reusable software component called Explanation Assistant (Step 4.1). The resulting Explanation Assistant then provides the application with a service to support all the explanations as specified by the Explanation Requirements. While Sec. 6 outlines the architecture of the service, it is worth mentioning here that it provides access points to an Application Programming Interface (API) for the application to post application data required by the Provenance Patterns logged from its execution to record the provenance of a decisions it makes. Another set of APIs is provided for the application to retrieve specific explanations for a specific decision. To ensure that the explanation service works correctly with the application, in Step 4.2, the Data Engineer works with the Application Stakeholders to integrate the Explanation Assistant service with the target application and test the integration over all the supported explanations.

6 Reference Implementation

The Explainability-by-Design methodology was executed to produce explanations for decisions in the credit card and school allocation applications (Sec. 3). The implementation is available as an online demonstrator accessible at https://explain.openprovenance.org. Since personal data is used for credit card applications and school applications, the data could not be shared and we were not allowed to access the actual systems (to instrument them with explanation capability). Therefore, we opted to simulate the decision making processes of the two applications from the technical inputs and anonymised data provided by our industry partners.The applications then generate fictitious data to feed into the simulated credit card application/school allocation pipelines, resulting in (simulated) decisions to be explained. This section discusses the technical choices made in this reference implementation with respect to the tasks in the ETD phase. Due to the limited space, the technical description of the implementation cannot be fully included; however, examples are provided to show how each task could be concretely executed.

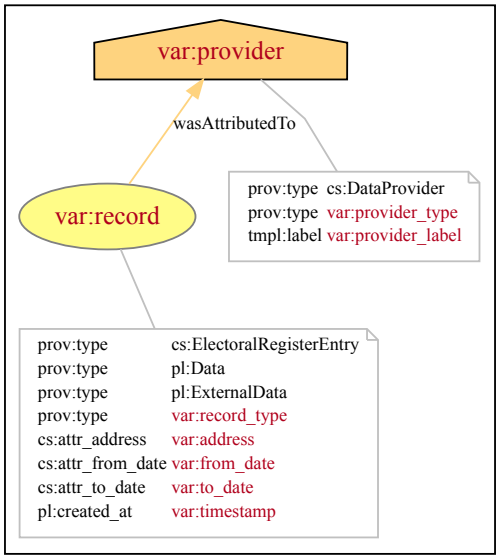

Fig. 3. An example provenance template describing that an entity identified by `var:record` was attributed to an agent identified by `var:provider`; the entity is of type `cs:ElectoralRegistryEntry` and the agent is of type `cs:DataProvider`.

6.1 Modelling Provenance

In this implementation, we adopted the PROV data model (PROV-DM) [10], the *de jure* provenance standard by W3C, for modelling the provenance of a decision. The PROV data model defines three core concepts: *entity*, *activity*, and *agent*; which can be related to one another by PROV relations. In brief, provenance describes the generation and use of entities by some activities, which may be influenced in some ways by agents. In other words, PROV-DM provides a provenance vocabulary that satisfies all the provenance requirements to support explanation generation (Sec. 5.1).

Since it is not always possible or easy to instrument an (existing) application to directly record provenance records of sufficient quality, to facilitate the systematic capturing of the provenance of decisions, we adopt the PROV-Template approach [25] to *decouple* the modelling of the provenance and its recording. It allows the data engineer to describe the provenance information to be recorded in a set of *provenance*

```
1  prefix pl <http://openprovenance.org/ns/plead#>
2  prefix cs <http://openprovenance.org/ns/
       ↪ creditscoring#>

4  select * from decision a prov:Entity

6  from derivation a provext:WasDerivedFromStar
7    join decision.id = derivation.generatedEntity

9  from record a prov:Entity
10   join derivation.usedEntity = record.id

12 where decision[prov:type] >= 'pl:Decision'
13 and record[prov:type] >= 'cs:CreditRecord'
14 group by decision aggregate record with Seq
```

Listing 1. An example provenance query.

templates which contain provenance records with placeholders, called *variables*, to be filled in later by values logged by an application during its runtime, called *bindings*. In other words, those provenance templates are an implementation of Provenance Patterns (Task 1) using the PROV vocabulary. Fig. 3, for example, presents a simple provenance template describing that an electoral record, an entity, was attributed to a data provider, an agent. The identities of the actual record (**`var:record`**) and the provider (**`var:provider`**), i.e. the bindings for those variables, are to be logged by the application at run time. When the full provenance (of a decision) is required, all the provenance templates designed for the application are *expanded*, or instantiated, with the values of the corresponding bindings recorded by the application. The results are a set of expanded provenance records constituting the full provenance records of the decision. The provenance template shown in Fig. 3, a corresponding binding, and the expanded provenance are provided in the Supplementary Materials as examples.

6.2 Building Queries

As the result of Task 1, the provenance templates can be expanded with application data to produce sample Provenance Traces, which are PROV provenance records. There are two key characteristics expected from a query in this context. First, the query should be able to return relevant PROV expressions and, second, it should be able to return a connected subgraph, typically linking a decision to one or more factors affecting it. From this subgraph, an explanation will be constructed (see Sec. 6.3), extracting the relevant data points, such as names, dates, and so on, necessary to create an explanation that is personalised to its recipient.

As graph query languages such as SPARQL [34] or Cypher [35] were designed to be domain-agnostic, they do not readily support the PROV data model. PROV records need to be encoded in or translated to a representation supported by either SPARQL or Cypher. In addition, PROV relations are not merely labelled edges in a graph but also contain associated data, such as the time a data entity was used by an activity. Therefore, the query language must also be able to extract the "edges," or PROV-relation records, so that the data associated with them can be referred to and included in an explanation. The original SPARQL, while supporting path queries, is not able to extract subgraphs, as it only returns the start and end of such path. Proprietary extensions of SPARQL[6] and Cypher allow for subgraphs to be extracted.

Against this background, our implementation relies of an extension for the relational query engine LegoBase [38] to work directly with PROV records and allow both PROV elements (i.e. nodes) and relations to be selected and returned in a query result. The definition of the query language is not in scope but an example is provided in Listing 1. We conjecture that our queries can be compiled to Cypher.

In essence, the query language allows binding variables to both PROV elements (e.g., Line 4) and relations (e.g., Line 6) using PROV terms; linking them via PROV relations by means of joins (Lines 7, 10); and, filtering the matched records by their attributes (Lines 12–13) to identify the right records required. Starting from a 'decision' (Line 4) of type `pl:Decision` (Line 12), the example query looks for all `prov:Entity` records (Line 9) of type `cs:CreditRecord` linked to it via `prov:WasDerivationFrom`[7] record(s) (Line 6); all the records found are to be aggregated into a list (Line 14). Executed against the provenance trace of a decision, if the query matches, the query variables (i.e. **`decision`**, **`derivation`**, **`record`**) are bound to the corresponding provenance records in the trace. Those matched provenance records, along with their properties, will then be available for explanation plans in constructing sentences.

6.3 Building Explanation Plans

For each sentence of explanation to be generated (provided by the Explanation Requirements), a linguistic syntax tree is constructed and passed to SimpleNLG [37], a surface NLG engine, to produced the final sentence. For example, Listing 2 shows the syntax tree for the sentence "Your credit score was impacted by [records details]" (constructed for the Credit Card scenario). Most of the sentence can be broken up into the various part-of-speech components: the verb 'impact' (Line 2), the noun phrase 'credit score' (Lines 4–5) with a possessive specifier reference `##borrower-possessive` (Line 6), and the indirect object (Lines 7–31). The syntax tree also specifies other linguistic feature of the sentence, such as this is a passive sentence in the past tense (Lines 32–33).

In addition to linguistic elements, a syntax tree in the reference implementation can contain references to variables in the corresponding provenance query. The '[records details]' part of the sentence, for instance, refers to data that is specific to a particular decision and, hence, requires the result of the associated provenance query (Listing 1) after being executed over the provenance of the decision. In order to support the (potentially) multiple instances of PROV records returned by the query and to represent them in a user-friendly format, a number of original auxiliary functions can be employed directly in the syntax tree. For instance, an iterator (Line 12) is used to connect the representation of all the PROV records bound to the variable **`record`** (Line 14) in the sentence. For each of the records, the value of its `prov:type` property is retrieved, and looked up in a Explanation Dictionary (Lines 17–22) to be included

6. See https://www.stardog.com/blog/a-path-of-our-own/.
7. We extended PROV terms in the query language to support querying the transitive closure of `prov:WasDerivationFrom` relations by querying for the `provext:WasDerivedFromStar` relation.

```
{ "type": "clause",
  "verb": "impact",
  "object": {
      "type": "noun_phrase",
      "head": "credit␣score",
      "specifier": "##borrower-possessive" },
  "indirect_object": {
      "type": "clause",
      "object": {
          "type": "coordinated_phrase",
          "conjunction": "and",
          "@iterator": {
              "type": "@iterator",
              "@variable": "record",
              "@clause": "coordinates",
              "@element": {
                "type": "@funcall",
                "@object": "record",
                "@property": "prov:type",
                "@function": "lookup-type",
                "@arg1": "noun_phrase",
                "@arg2": "csd",
                "post-modifiers": [{
                  "type": "adjective_phrase",
                  "head": {
                    "type": "@funcall",
                    "@object": "record",
                    "@field": "id",
                    "@function": "noun+localname" }
                }] } } },
      "complementiser": "by" },
  "features": { "type": "features",
    "tense": "past", "passive": "true" }
}
```

Listing 2. An example syntax tree (represented in JSON).

in the sentence as a noun phrase, along with its identifier (Lines 25–29). This allows sentences such as "Your credit score was impacted by your salary (records/960) and a late payment (records/956)". In addition (not shown in the figure), the explanation plan also has the possibility of specifying HTML markup to present record identifiers as hyperlinks. This allows for explanations to be navigated by their recipient, allowing them to drill down in the contents according to their interest.

Finally, the explanation sentence can be further customised to suit different 'profiles' of the target audience. For instance, explanations for a credit card decision could be presented to two difference audiences: the applicant and the bank's staff. These two profiles are defined in an Explanation Dictionary with suitable application-specific terms for each of the them. In the above example, the reference `##borrower-possessive` marks an entry to be looked up from the application's Explanation Dictionary. It will be converted into a second-person possessive determiner ("your") if the former profile is selected, or a third-person possessive determiner ("their") if the latter is selected.

6.4 Explanation Assistant Service

In this implementation, we developed an independent software component called the Explanation Assistant service to encapsulate all the above functionality to support provenance templates (logging data and expanding templates), executing provenance queries, and instantiating explanation plans (into sentences). We adopted a document-driven and document-transformation architecture for the service (see

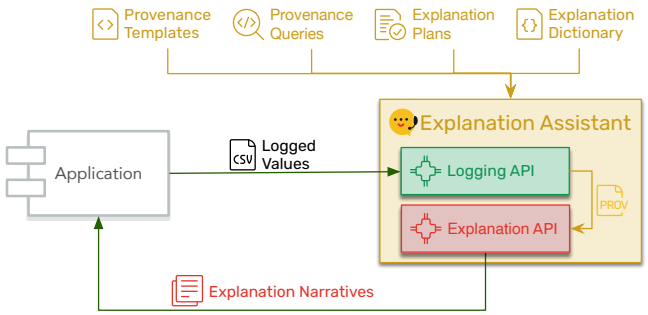

Fig. 4. Software architecture for deploying an Explanation Assistant.

Fig. 4). After the Explanation Assistant is configured with the artefacts produced in the previous three tasks (i.e., Provenance Templates, Provenance Queries, Explanation Plans, and Explanation Dictionary), the service exposes two APIs to provide the configured explanation capability to the associated application: Logging API and Explanation API. For each decision made by an application, the associated data as specified by the provenance templates are recorded in a CSV file and submitted to the Explanation Assistant. With the configured provenance templates, the CSV file is transformed into the full provenance of the decision in a PROV document. The provenance then can be queried on and explanations generated on demand when the application requests those via the supplied Explanation API.

7 Evaluation

In this section, we assess the effort executing the ETD phase for the two application scenarios (Sec. 3) by some cost metrics based on the artefacts it produces in each task (Sec. 7.1). Sec. 7.2 discusses those costs and the development effort incurred in executing the methodology.

7.1 Quantifying costs of the ETD phase

The amount of effort required to execute the ETD phase is proportionate to the numbers of Explanation Requirements to be supported. Here, we propose a way of measuring the cost of the ETD tasks based on the number of artefacts produced in each task and their complexity. Such measurements provide an indication of the efforts expensed during a task (also see Table 6):

- Provenance Modelling: This task is concerned with the data flows of the whole application and is not specific to an explanation requirement. The effort of this task can be estimated by the number of provenance templates it produces and their complexity, defined by the total number of provenance elements and relations contained therein.
- Building Queries: This task produces Provenance Queries and its cost can be estimated by the number of queries and their complexity. The latter is defined as the number of 'joins' and the number of filtering conditions it requires to describe the graph pattern to be matched by a query.
- Building Explanation Plans: The Explanation Plans consist of a syntax tree, whose complexity can be quantified

TABLE 6
Cost metrics of ETD tasks

Task	Cost
Provenance Modelling	Total number provenance records in all templates
Building Queries	Total number of joins and filters in all queries
Building Explanation Plans	Total number of nodes in all syntax trees Numbers of supported profiles and dictionary terms
Iterative development	Numbers of revisions of templates, queries, explanation plans

TABLE 7
The artefacts produced to support the explanations requirements for the Credit Card and School Allocation applications (Sec. 3).

Artefact	Credit Card	School Allocation
Explanation Requirements	7	4
Template	11	7
Queries	11	4
Explanation Plans	22	11
Sentences	34	12
Dictionary Terms	16	10
Profiles	2	2

by the number of linguistic elements they contain. We define the complexity of an explanation plan as the total number of nodes within its narrative's syntax trees. The effort of this task is also increased by the number of profiles supported in the Explanation Dictionary, reflected in the number of dictionary terms defined therein.

- Deploying Explanation Assistant: The PLEAD Explanation Assistant is a prebuilt software service, which does not require further development effort. This task requires only packaging the artefacts produced in the previous tasks into a configuration to instantiate the Explanation Assistant service; hence, the effort is minimal.
- Iterative development: Due to the interdependencies of the artefacts produced by the ETD activities, one may need to, say, revise a provenance template (Task 1) to facilitate the writing of a provenance query (Task 2). Therefore, the cost of the whole process will also need to take into account the number of revisions of provenance templates, provenance queries, and explanation plans.

To provide some indication of the efforts involved, we report the numbers of artefacts produced to support the explanations for the two applications Credit Card and School Allocation (described in Sec. 3) in Table 7. The two applications have 7 and 4 input Explanation Requirements, respectively, elicited by a legal engineer. The explanation narrative for an explanation requirement may consist of multiple sentences; to facilitate the development of the supporting explanation plans, those sentences may be divided into multiple explanation plans. We, therefore, also report the total number of explanation sentences produced for each application. Further detailed cost metrics are provided in the Supplementary Materials.

TABLE 8
The estimated developer time in person days incurred to support the explanations for the Credit Card* and School Allocation† applications.

Metric	Credit Card	School Alloc.	Ratio
Total dev. time (person day)	29.6*	3.2†	9.25
Dev. time/sentence (hour)	6.5*	2†	3.25

* including Explanation Assistant co-development time
† *no* co-development required

7.2 Discussion

The technical development of explanations for the two application scenarios was carried out by a team of three developers: D1, D2, and D3. Their roles in the development of the two applications were as follows:

D1 is a *novice* developer, spending 50% of their time on this project, and responsible for crafting the explanation plans (constructing the NLG syntax trees from provenance query results, i.e., Task 3).

D2 is an *expert* developer, spending 25% of their time on this project, and responsible for modelling the provenance for the two applications (Task 1) plus designing the provenance queries (Task 2) to support D1's task.

D3 is an *advisor*, spending 5% of their time on this project training/supporting D1 by providing a few initial example explanation plans to help them get started.

The involvement of the three developers in the development of the Credit Card and School Allocation applications are visualised on the timelines provided in the Supplementary Materials (due to the limited space). The development timelines are summarised in in Table 8. We now discuss the development of explanations for the two applications in terms of the development time, iterative development, and reusability.

Development time: The explanations for the Credit Card application were co-developed while the Explanation Assistant's support for provenance queries and explanation plans were also being implemented in parallel (April and May 2021). Therefore, the number of edits to queries and explanation plans are higher when compared with those in the School Allocation application. In contrast, the explanations for the School Allocation application were developed after all the technical support had been in place. As a result, the development effort in the School Allocation application (Table 8) was more typical of the required effort to carry out the ETD tasks. In fact, the majority of the effort was made between 7th and 12th of July 2021 (inclusively), a total of 4 working days. Considering the levels of commitment of the three developers, this can be estimated at about 3.2 full days of developer time, to support 11 explanation plans from Task 1 to Task 3. The effort outside the above time window is minimal and mostly was to refine the explanation narratives. In comparison, the majority of the development for the Credit Card application was done between 6th April and 27th May 2021 (inclusively), a total of 37 working days. The total developer time for this application is, hence, estimated at 29.6 days, or 6.5 hours per sentence on average (assuming 7.5 working hours/day). In contrast, it took on average 2 hours to support one explanation sentence for the School Allocation application.

Iterative Development: In order to find and include some information in an explanation plan, sometimes the required data was not readily available from the plan's corresponding provenance query. This then necessitated the revising of the query to retrieve the required data. However, in some cases, the required data was not captured in the (sample) provenance traces; and to fix this, some provenance templates were revised to record the extra data from the application. In other cases, a mistake identified in a query is fixed resulting in edits in explanation plans depending on the query. Inspecting the development history of provenance templates, queries, and explanation plans (recorded in the git repository for the above artefacts), we identify 13 occasions of such iterative revisions, 7 of those involving edits by two developers (see the Supplementary Materials). Hence, a close collaboration within the development team is expected to support such an iterative process.

Reusability: In both applications, the numbers of provenance queries are significantly lower than the numbers of explanations plans: 22 explanation plans are supported by 11 queries in the Credit Card application, and 11 by 4 in the School Allocation application (Table 7). The reason is that many of the queries are shared by multiple explanation plans. For instance, one sentence may expand from the information provided by the previous sentence by providing more details that are available in the result of the same query used by the previous sentence (see the Supplementary Materials for details of how queries were shared between explanation plans). In addition, the profiles of target audience in many case can be easily modified to be used in a new application. In fact, the two profiles crafted for the Credit Card application to address explanations to (1) the borrower and (2) the bank staff were copied to use for the School Allocation application to address explanations to (1) the applicant and (2) the organisation staff. The main modifications required were changing the words 'borrower' to 'guardian' and 'the bank' to 'the school'.

8 Conclusions

The increasing dependence of decision-making on some level of automation has naturally led to discussions about the trustworthiness and the fairness of such automation, resulting in calls for transparent automated decision-making and making such systems explainable. In this article, we put forth Explainability-by-Design, a methodology to enable organisations to address their explainability systematically and holistically: (A) analysing explanation requirements within a social/legal context, (B) technically implementing the requirements, and (C) validating the produced explanations. We have fully described the Explanation Technical Design phase (B) and produced a reusable Explanation Assistant service available as an online demonstrator. Following the methodology, organisations will be able to design their decision-making systems to track the provenance of decisions and produce individually tailored explanations for them to meet legal, social, or business requirements. In addition, we presented various cost metrics associated with the methodology measured from the enactment of the methodology over two applications: we have shown that, thanks to the well delineated steps and reusable service, the approach requires modest development time per explanation to be generated.

While the chosen two applications drew their explanation requirements from the regulatory requirements for explaining credit card and school allocation decisions, there is no obligation for the methodology to be legally driven. In fact, explanation requirements could equally arise from business requirements (e.g., to improve customer trust, to reduce customer enquiries, to support internal audits) or somewhere else. In machine planning, for instance, which is now used in safety-critical applications (such as shipping, oil-well drilling, and controlling swarms of unmanned vehicles), explaining why a certain action was chosen over another in a plan is key in engendering trust in the users who are responsible and accountable for authorising the execution of a plan [39].

The Explainability-by-Design methodology can be extended in a number of ways. Notably, it does not attempt to explain the inner working of algorithmic models employed by decision-making systems but takes a holistic approach to produce explanations about the data and processes surrounding such models. At the same time, it also does not preclude the use of existing XAI techniques to explain a recommendation by an ML model such as LIME [40] or SHAP [41]. In fact, such techniques would complement and augment explanations about the decision-making pipelines currently supported by EbD by drawing out the relations between the input data and a model's recommendation in a particular decision. The provenance of a decision could then be extended to include those relations, allowing provenance queries to pinpoint the input data that was influential and, by so doing, making the explanations more specific.

Although EbD is designed to produce ex-post explanations about what happened, its technical approach can be extended to produce counter-factual explanations, which aim to provide insights into which external factors could be different to arrive at a desired outcome [42]. In essence, a system can simulate counter-factual cases and produce the corresponding (hypothetical) decisions, with their provenance recorded as in an actual case. The provenance traces of the counter-factual cases can then be summarised or compared with that of the actual decision to generate counter-factual explanations. An initial exploration of the idea was included in the loan demonstrator reported in [29].

The reference implementation of EbD for the credit card and school allocation applications was done without any tooling support available. The reported development time would have been significantly reduced if software tools were available to support the developers in their tasks. Such tools could help with visually crafting Provenance Queries (Task 2) and linguistic syntax trees (Task 3), they could also check whether the Provenance Traces are all connected (**PR2**, Task 1) and if the variable references in Explanation Plans match those available in Provenance Queries.

Finally, we have demonstrated that the Explanation Assistant is a reusable system, playing a critical part in keeping the EbD execution time low for the two showcased applications. The methodology is, however, independent of the actual provenance templating system, the provenance query languages, and the explanation planning engine. To increase the reusability of the approach further, well de-

fined interfaces would allow alternative templating engines, query languages and explanation planner to be plugged in.

ACKNOWLEDGMENTS

The work presented in this article has been supported by the UK Engineering and Physical Sciences Research Council (EPSRC) via the Grants [EP/S027238/1] and [EP/S027254/1] for the PLEAD project, [EP/R033722/1] for the THuMP project, and [EP/V00784X/1] for the Trustworthy Autonomous Systems Hub.

REFERENCES

[1] J. Heurix, P. Zimmermann, T. Neubauer, and S. Fenz, "A taxonomy for privacy enhancing technologies," *Computers & Security*, vol. 53, pp. 1–17, sep 2015.
[2] P. Hustinx, "Privacy by design: delivering the promises," *Identity in the Information Society*, vol. 3, no. 2, pp. 253–255, 2010.
[3] M. E. Kaminski, "The right to explanation, explained," *Berkeley Technology Law Journal*, vol. 34, no. 1, pp. 189–218, 2019.
[4] The UK Information Commissioner's Office, "Explaining decisions made with AI," Tech. Rep., 2020.
[5] N. Tsakalakis, S. Stalla-Bourdillon, L. Carmichael, D. Huynh, L. Moreau, and A. Helal, "The dual function of explanations: Why computing explanations is of value," in *Data Protection and Privacy*, D. Hallinan, R. Leenes, and P. De Hert, Eds. Hart Publishing, 2022, vol. 14, ch. 5, pp. 127–156.
[6] A. D. Selbst and J. Powles, "Meaningful information and the right to explanation," *International Data Privacy Law*, vol. 7, no. 4, pp. 233–242, 2017.
[7] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52 138–52 160, 2018.
[8] J. Cobbe, "Administrative law and the machines of government: Judicial review of automated public-sector decision-making," *Legal Studies*, vol. 39, no. 4, pp. 636–655, 2019.
[9] A. Burt, S. Shirrell, B. Leong, and X. G. Wang, "Beyond explainability: A practical guide to managing risk in machine learning models," Future of Privacy Forum, Tech. Rep., 2018.
[10] L. Moreau and P. Missier, "PROV-DM: The PROV data model," World Wide Web Consortium, Tech. Rep., 2013, W3C Recommendation. [Online]. Available: http://www.w3.org/TR/2013/REC-prov-dm-20130430/
[11] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial Intelligence*, vol. 267, pp. 1–38, feb 2019.
[12] J. Singh, J. Cobbe, and C. Norval, "Decision provenance: Harnessing data flow for accountable systems," *IEEE Access*, vol. 7, pp. 6562–6574, 2019.
[13] A. Rieke, M. Bogen, and D. G. Robinson, "Public scrutiny of automated decisions: Early lessons and emerging methods," Upturn and Omidyar Network, Tech. Rep., 2018.
[14] European Union, "Regulation 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation)," *Official Journal of the European Union*, vol. 59, no. L 199, pp. 1–88, 2016.
[15] L. Edwards and M. Veale, "Enslaving the algorithm: From a "right to an explanation" to a "right to better decisions"?" *IEEE Security & Privacy*, vol. 16, no. 3, pp. 46–54, may 2018.
[16] C. Molnar, *Interpretable Machine Learning*, 2019.
[17] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Surveys*, vol. 51, no. 5, pp. 1–42, jan 2019.
[18] C. Tantithamthavorn, J. Jiarpakdee, and J. Grundy, "Explainable AI for software engineering," 2020. [Online]. Available: https://arxiv.org/abs/2012.01614
[19] X. Ren, Z. Xing, X. Xia, D. Lo, X. Wang, and J. Grundy, "Neural network-based detection of self-admitted technical debt: From performance to explainability," *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 3, 2019.
[20] C. Abid, D. Rzig, T. Ferreira, M. Kessentini, and T. Sharma, "X-SBR: On the use of the history of refactorings for explainable search-based refactoring and intelligent change operators," *IEEE Transactions on Software Engineering*, vol. 14, no. 8, 2021.
[21] J. Cobbe, "Administrative law and the machines of government: judicial review of automated public-sector decision-making," *Legal Studies*, vol. 39, no. 4, pp. 636–655, jul 2019.
[22] V. C. Storey, R. Lukyanenko, W. Maass, and J. Parsons, "Explainable AI," *Communications of the ACM*, vol. 65, no. 4, pp. 27–29, 2022.
[23] Information and Privacy Commissioner of Ontario, "Privacy by design: Seven foundational principles," 2018. [Online]. Available: https://www.ipc.on.ca/resource/privacybydesign/
[24] S. Miles, P. Groth, S. Munroe, and L. Moreau, "PrIMe: A methodology for developing provenance-aware applications," *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 3, pp. 1–42, aug 2011.
[25] L. Moreau, B. V. Batlajery, T. D. Huynh, D. Michaelides, and H. Packer, "A templating system to generate provenance," *IEEE Transactions on Software Engineering*, vol. 44, no. 2, pp. 103–121, 2018.
[26] C. Sáenz-Adán, F. J. García-Izquierdo, B. Pérez, T. D. Huynh, and L. Moreau, "Automated and non-intrusive provenance capture with UML2PROV," *Computing*, vol. 104, no. 4, pp. 767–788, 2022.
[27] P. Groth, Y. Gil, J. Cheney, and S. Miles, "Requirements for provenance on the web," *International Journal of Digital Curation*, vol. 7, no. 1, pp. 39–56, mar 2012.
[28] S. Martínez-Fernández, J. Bogner, X. Franch, M. Oriol, J. Siebert, A. Trendowicz, A. M. Vollmer, and S. Wagner, "Software engineering for AI-based systems: A survey," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 2, pp. 1–59, apr 2022.
[29] T. D. Huynh, N. Tsakalakis, A. Helal, S. Stalla-Bourdillon, and L. Moreau, "Addressing Regulatory Requirements on Explanations for Automated Decisions with Provenance—A Case Study," *Digital Government: Research and Practice*, vol. 2, no. 2, pp. 1–14, 2021.
[30] N. Tsakalakis, S. Stalla-Bourdillon, T. D. Huynh, and L. Moreau, "A taxonomy of explanations to support explainability-by-design," Under review by a journal, 2022.
[31] "The Open Provenance Model core specification (v1.1)," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 743–756, jun 2011.
[32] G. Klyne, J. J. Carroll, and B. McBride, "RDF 1.1 concepts and abstract syntax," World Wide Web Consortium, Tech. Rep., 2014, W3C Recommendation.
[33] H. Knublauch and D. Kontokostas, "Shapes constraint language (SHACL)," World Wide Web Consortium, Tech. Rep. REC-shacl-20170720, 2017, W3C Recommendation.
[34] S. Harris and A. Seaborne, "SPARQL 1.1 query language," World Wide Web Consortium, W3C Recommendation REC-sparql11-query-20130321, 2013.
[35] N. Francis, A. Green, P. Guagliardo, L. Libkin, T. Lindaaker, V. Marsault, S. Plantikow, M. Rydberg, P. Selmer, and A. Taylor, "Cypher: An evolving query language for property graphs," in *Proceedings of the 2018 International Conference on Management of Data*. New York, NY, USA: ACM, may 2018, pp. 1433–1445.
[36] E. Reiter and R. Dale, "Building applied natural language generation systems," *Natural Language Engineering*, vol. 3, no. 1, pp. 57–87, 1997.
[37] A. Gatt and E. Reiter, "Simplenlg: A realisation engine for practical applications," in *Proceedings of the 12th European Workshop on Natural Language Generation*, ser. ENLG '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 90–93.
[38] A. Shaikhha, Y. Klonatos, and C. Koch, "Building efficient query engines in a high-level language," *ACM Transactions on Database Systems*, vol. 43, no. 1, pp. 1–45, apr 2018.
[39] B. Krarup, M. Cashmore, D. Magazzeni, and T. Miller, "Towards model-based contrastive explanations for explainable planning," in *2nd ICAPS Workshop on Explainable Planning*, 2019, pp. 21–29.
[40] M. T. Ribeiro, S. Singh, and C. Guestrin, ""Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, 2016, pp. 1135–1144.
[41] M. Sundararajan and A. Najmi, "The many Shapley values for model explanation," in *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020, pp. 9269–9278.
[42] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR," *Harvard Journal of Law & Technology*, vol. 31, no. 2, 2017.