Prototyping an End-User User Interface for the Solid Application Interoperability Specification Under GDPR

Hadrien Bailly[1](✉), Anoop Papanna[1], and Rob Brennan[2]

[1] Dublin City University, Glasnevin Campus, Dublin 9, Ireland
{hadrien.bailly2,anoop.papanna2}@mail.dcu.ie

[2] ADAPT, University College Dublin, Belfield, Dublin 4, Ireland
rob.brennan@adaptcentre.ie

Abstract. This paper describes prototyping of the draft Solid application interoperability specification (INTEROP). We developed and evaluated a dynamic user interface (UI) for the new Solid application access request and authorization extended with the Data Privacy Vocabulary. Solid places responsibility on users to control their data. INTEROP adds new declarative access controls. Solid applications to date have provided few policy interfaces with high usability. GDPR controls on usage are rarely addressed. Implementation identified specification and Semantic Web tool issues and also in the understandability of declarative policies, a key concern under GDPR or data ethics best practices. The prototype was evaluated in a usability and task accuracy experiment, where the UI enabled users to create access and usage control policies with an accuracy of between 72 and 37%. Overall, the UI had a poor usability rating, with a median SUS (system usability scale) score of 37.67. Experimental participants were classified according to the Westin privacy scale to investigate the impact of user attitudes to privacy on the results. The paper discusses the findings of the study and their consequences for future data sovereignty access request and authorization UI designs.

Keywords: Solid · Access Control · User Interface · GDPR · Consent

1 Introduction

The Solid Platform was designed to address the loss of control over personal data [30]: enabling users from all backgrounds to regain control over their personal data and freely choose with whom to or not to share it. However, to date little attention has been given to how users - and particularly non-power users - would actually exert their control of access to their Solid Personal Online Datastore (*POD*). Another deficit has been Solid's use of Access Control List (ACL) policies to specify who can access which data [7], a process that is tedious and prone to errors [1], and does not allow users to specify what usage of their data is allowed – in spite of the fact that they are frequently unhappy with what happens

C. Pesquita et al. (Eds.): ESWC 2023, LNCS 13870, pp. 557–573, 2023.
https://doi.org/10.1007/978-3-031-33455-9_33

to their data after it was shared [24]. In 2022, the *Solid Data Interoperability Panel* (INTEROP) proposed a new, more intuitive specification to represent application access requests and user grants [2] with a more flexible policy model. In parallel, the Data Privacy Vocabulary (DPV) [22] has emerged for describing data processing purposes based on the European General Data Protection Regulation (GDPR). This legal exactness also comes with a potential cost in usability for non-experts. The INTEROP specification and DPV ontology are more expressive than Solid's original ACL approach and create innovation possibilities, but it is still not obvious how end users will be able to understand and manipulate their complex expressions.

This paper explores the following research question: "*To what extent can an access request and authorization UI effectively enable Solid users to specify INTEROP access policies and DPV usage control policies?*" It describes the design and evaluation of a new application access request and authorization UI for end users, based on first independent prototyping of the *Solid Data Interoperability Panel* (INTEROP) specification [2] combined with the *Data Privacy Vocabulary* (DPV) [22].

The contributions of the paper are as follows:

1. A new UI design for presenting Solid users with the access needs and intended processing actions of an application requesting access to their POD using the INTEROP specification;
2. A validation of the INTEROP specification through prototyping, resulting in a set of identified design issues and improvement suggestions;
3. A new open source prototype UI implementation of the INTEROP specification extended with the DPV, that generates fine-grained access and usage control authorizations according to the INTEROP data shape format;
4. A user evaluation of the new UI usability and user satisfaction, based on task completion accuracy, the System Usability Scale (SUS), Net Promoter Score (NPS) and a profile of users using the Westin Privacy scale.

The paper is structured as follows: Sect. 2 provides a motivating use case and lists the UI requirements; Sect. 3 overviews related work in Solid access and usage control and policy UI usability; Sect. 4 describes the UI design and its implementation; Sect. 5 details the user experiment; Finally, Sect. 6 describes conclusions.

2 Use Case and Requirements

This section illustrates the use of an application access request and authorization UI with an example use case drawn from the INTEROP specification [2].

Alice is a Solid User: she owns a POD, in which she stores both professional and personal data. Alice has several projects and tasks stored in her POD, and has found the Projectron app to manage them. Alice has never authorized Projectron before, so when she starts the app, she is presented with the application access request and authorization UI. First, Alice reviews Projectron's access needs, as defined by Projectron developers. These needs outline which (types of) data Projectron needs to access, what operations it will perform, and for what purpose, so Alice can understand what access she is asked to grant. Alice inspects each need, and decides that she will allow Projectron access her POD as described in the access request. She configures the scope of access for each need, then approves the request: the UI automatically generates a set of authorizations for Projectron. Projectron is now authorized. Later, when Projectron attempts to perform an operation on data in Alice's POD, the set of authorizations is used by an authorization agent to check if Projectron 1) was authorized by Alice; 2) is allowed to access that particular data in Alice's POD, and perform the requested operations on it; 3) has a purpose for performing the operation agreed to by Alice in the authorizations. If the attempt passes all checks, it can proceed, otherwise it is rejected.

This leads to these user and technical requirements for the application access request and authorization UI:

R1 When an arbitrary collection of access requirements in INTEROP format is present in a Solid request from an application, the UI shall graphically present it to the user. It must precisely identify what data the application requires and what operations it intends to conduct.
R2 The UI shall directly consume and produce Solid and RDF resources.
R3 The UI shall produce authorizations that can be used to allow/deny specified operations by a given application on any existing or future resource located in the user' POD.
R4 The UI shall enable users to "consent to the processing of their personal data for one or more specific purposes" (*EU General Data Protection Regulation (GDPR)*, art. 6.) [21].
R5 The UI shall support users "deciding whether it makes sense for them to share their personal data" [16], but avoid information overload.
R6 The UI shall be user-oriented [29], task-based [28], and enable users to create accurate authorizations.

3 Related Work

This section briefly describes Solid, Solid access and usage control schemes, Solid access control user interfaces, and usability evaluation techniques.

Solid and Usage and Access Control. Solid aims to return data sovereignty to web users [18]. It has two declared ambitions: 1) allow users to store data in a decentralized POD, and reuse it for multiple purposes (applications), 2) allow users to share data from their POD securely with multiple service providers. Solid relies on the *Resource Description Framework* (RDF), which enables third parties to locate and interact with a user's resources. Sharing resources requires that their owners can protect it from unwanted or unauthorized access. The ability of a data owner to authorize access and use of a resource has two dimensions:

access and usage control [27], where **access** denotes the authorization to read and edit some resources, and **usage** the conditions and obligations associated with this authorization. Traditionally, access control models encompass *Mandatory Access Control* (MAC), *Discretionary Access Control* (DAC), *Role-Based Access Control* (RBAC), and *View-Based Access Control* (VBAC) models. Additional models have been proposed, the most notable being the *Attribute-Based Access Control* (ABAC), and *Context-Based Access Control* (CBAC) models [15].

Simple access control models using access control lists (ACL) can take a document-centric view of RDF that applies to groups of triples in a document. More fine-grained controls use declarative constraint languages to describe what a resource contains (or should contain). To that end, RDF systems can make use of two constraint languages: SHACL[1] and ShEx (*Shape Expression*)[2], which define the *shape* with which the graph of a given resource is expected to comply. To describe complex sets of resources, multiple shapes can be assembled together to form *Shape Trees*[3]. Shapes and shape trees can be used to scope access or usage controls in the INTEROP specification.

Table 1. Access & Usage Control Protocols in Solid

		Personal Data Sovereignty				*User experience*	
Protocol		**Model**	**Access**	**Usage**	**GDPR**	**UI**	**Task-based**
Web Access Control (WAC)	[6]	DAC	✓	✗	✗	✓ [12]	✗
Access Control Policy (ACP)	[3]	CBAC	✓	✓	✗	✗	✗
Data Interoperability Panel Specification (INTEROP)	[2]	DAC	✓	✗	✗	?	✓
eXtensible Access Control Markup Language (XACML)	[26]	ABAC	✓	✗	✗	✗	✗
Open Data Rights Language (ODRL)	[11]	CBAC	✓	✓	✗	✗	✗
Open Regulatory Compliance Profile (ORCP)	[14]	CBAC	✓	✓	✓	✗	✗
ODRL Profile for Access Control (OAC)	[9]	CBAC	✓	✓	✓	✓SOPE [8]	✗

Currently, Solid resources are mainly secured using the DAC model and ACLs. Table 1 summarises the main protocols that have been proposed for Solid. *Web Access Control* (WAC) [6] is the current Solid standard. It uses ACL permission files to define the operations that an application can execute on a given resource. WAC offers only unrestricted read, write, append or control access to a resource, or no access at all – context or usage restrictions can only be supplied by the application. *Access Control Policy* (ACP) [3] was presented as an alternative, using context and conditions to limit operations on resources beyond binary access authorization. In the new INTEROP specification [2] rather than controlling individual resources or resource containers, users dynamically create policies for the needs of applications as a whole. When connecting with a new application, users are presented with an access request outlining a set of access needs for the application. Users can define either explicit authorizations to given

[1] https://www.w3.org/TR/shacl/.
[2] https://shex.io/shex-semantics/index.html.
[3] https://shapetrees.org/TR/specification/.

resources, or implicit ones to (existing or future) resources of a given data shape and location. An authorization agent is then responsible for inferring effective permissions for each resource in the POD. The access policies created under the INTEROP specification do not include usage control. The Open Data Rights Language (ODRL) Profile for Access Control (OAC) [9] aligns access and usage permissions with the GDPR for Solid. It ensures usage control policies can be written in compliance with the GDPR's requirements using the Data Privacy Vocabulary (DPV) [22].

Solid Access Control User Interfaces. Solid users can only currently create and manage WAC and OAC policies, using off-the-shelf file editors [12], and summarized in Table 1. The INTEROP specification has not to date been implemented in a user interface – although a wireframe design[4] was presented. Off the shelf editors are inappropriate to manage security: the policy editing workload is often too heavy, even for simple tasks [5] and most users, even experts, fail to recognize the implications of changes [1]. The design of Access Control UIs must balance between the need for accurate information and the reluctance or inability of users to process lengthy policies [19]. These UIs should be simple and task-oriented, and not require previous knowledge of the underlying technical model [28]. The level of understanding of ACL permissions, and security policies in general, is dependent on two factors: the technical knowledge of the user and the design of the UI [10]. Solid aims to address all categories of users, including members of the public. The access control UI design must reflect this.

Usability and Satisfaction Evaluation for Privacy Systems. The System Usability Scale (SUS) [4] is one of the most frequently used techniques to evaluate usability. SUS is a 10-item questionnaire that generates a 0–100 score from each participant. A system is considered above average usability if it scores above 68 for more than 50% of the participants. SUS does not identify causes of usability problems. To detect these, software engineers can instead use the Nielsen's 10 Design Heuristics and think-aloud [20]. This allows a limited number of evaluators to consistently identify most of the issues in a UI.

User satisfaction is also important in the success of a system. The *Net Promoter Score* (NPS) [25] is a tool frequently used to quickly evaluate the user satisfaction. It consists of a single rating question, ranging from 1 to 10 and asking whether a user is likely to recommend a product to family, colleague or friend. If the respondents rate the UI under 7, then they are said to be *detractors*. NPS can be followed by an open question to allow the respondents to elaborate. The pertinence of the NPS alone as an accurate measure for user satisfaction is debated [31], and it is advised to cross-check with other metrics and feedback.

Finally, it has been shown that the *profile* of users also plays a significant role in their perceived user experience [17]. When it comes to sharing private

[4] https://github.com/solid/data-interoperability-panel/blob/main/proposals/primer/images/authorization-screen.svg.

data, the popular Westin Scale defines three classes of users [16], which can be uncovered with a short 5-item questionnaire: 1) 34% of users are *fundamentalists*. They are concerned about their privacy, and proactively refuse to provide data; 2) 8% of users are *unconcerned*, least protective of their data and considering that the benefits of sharing outweigh the risks of breach; 3) The remaining users (58%) are *pragmatic* in their approach to privacy. They evaluate the pros and cons, and share data if it makes sense to them.

In summary, Solid relies heavily on the personal control of data sharing, yet the current access control protocols are coarse-grained and based on DAC models that do not support application needs well (**R2**, **R3**). The INTEROP specification aims to fix this with more expressive shape-based policies based on application needs (**R1**, **R5**), but users are not currently supported by appropriate access control UIs (**R1**, **R5**, **R6**) and INTEROP will place even more demands on them. In addition, GDPR-compliant usage control (**R4**) is not directly addressed.

4 A Proposed Application Access Request and Authorization User Interface for INTEROP

This section describes the architecture, features and implementation of a new dynamic access control UI for INTEROP (Fig. 1). The UI prototyping discussed identified gaps and two design issues in the draft INTEROP specification.

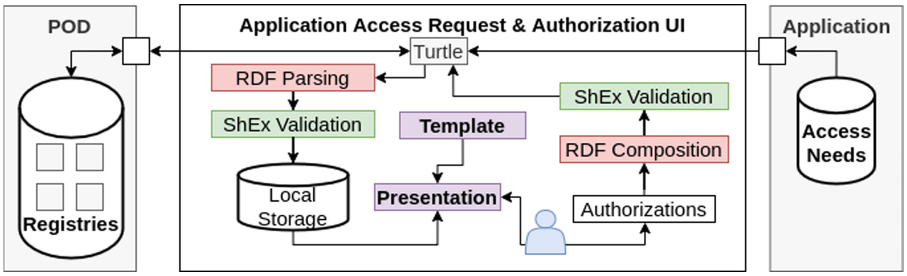


Fig. 1. Application Access Request and Authorization User Interface Architecture

4.1 Design

This UI was designed to meet the requirements in Sect. 2. It enables users to review arbitrary access requests from applications (**R1**) to use RDF resources (needs) (**R2**), and create INTEROP authorizations corresponding to the access needs (**R3**). When presenting a request, the UI provides explicit information about which resources are needed and for what GDPR-style purpose (**R4**), and gives users the opportunity to grant or refuse access (**R5**).

To display the access request and create the authorization, the UI requires input from two sources: 1) information about the content (registries) from the user's POD, and 2) the collection of access needs requested by an application. As per INTEROP, the input is received in RDF, parsed, and validated against ShEx shapes. When data has been received, an application access request UI is dynamically generated from a template and presented to the user for review (as show in Fig. 1). The user can inspect the request, select/deselect optional access needs, specify the extent (scope of) access, and decide whether to accept the request (**R1**, **R5**). If accepted, then the corresponding access and data authorizations are generated according to the user specification, composed into RDF triples and validated against ShEx, before being returned to the POD for enforcement by the INTEROP authorization agent (**R3**).

The design combines the INTEROP specification and the DPV to provide users with structured information about the application access request (**R1**, **R4**). It uses the INTEROP definition of access needs to present the access requirements of the application – in terms of type of resources accessed (shapes) and technical ACL permissions. Then this is supplemented with DPV terminology: first to describe the sensitivity of the data access and the purpose of use, and second to record the purpose to which the users consent to (e.g. *dpv:Service Provision* vs *dpv:Analytics*).

To discover what are the access needs of an application, and generate the access request UI, it draws a directed tree containing all the access needs and descriptions, starting from the application profile resource (`interop:Application`). However, this is currently impeded by two INTEROP design choices: (1) the use of collections and (2) several predicates incompatible with a directed tree structure (*backlinks*). For instance, a description is linked to an access need using the predicate `AccessNeedDescription.hasAccessNeed`, while no other node in the graph possesses an IRI pointing to this description. As a result, to obtain the description of an access need, one needs to 1) return to the parent access need group, 2) retrieve the corresponding access description set and unnamed collection of descriptions, and 3) iterate over all access need descriptions resources until one is found with a backlink `hasAccessNeed` to the need.

To overcome these issues, new predicates were introduced into the shape expressions:

– `accessNeed.hasAccessNeed` to obtain the list of dependent access needs, and
– `accessNeedGroup.hasAccessNeedGroupDescription` and `accessNeed.hasAccess-NeedDescription` to obtain the corresponding descriptions by language.

It was also detected that the INTEROP specification is inconsistent in its use of the predicates `skos:prefLabel` and `skos:definition` to provide labels and descriptions. Several shapes miss either a label, a description, or both – such as `SocialAgent` or `DataInstance` (cf. Table 2). Other shapes use them inconsistently, e.g. between `AccessNeedDescription` using `prefLabel` as label and `AccessNeedGroupDescription` as definition. This does not prevent the creation of directed trees or presentation of authorization requests, but it precludes the use of human-readable labels over IRI in a dynamic user interface and violates ontology design best practices P20 [23]. New (`skos`) predicates were again introduced into the shape expressions of a number of resources to enforce the presence of these predicates for access request UI generation (cf. Table 4).

Table 2. INTEROP Concepts Missing Labels and Descriptions

Resource	Has Label	Has Description
Social Agent	✗	✗
Access Need Group*	✓	✓
Access Need*	✗	✓
Data Registry	✗	✗
Data Instance	✗	✗

**resources using linked description*

Finally, we propose the following changes in the INTEROP specification: (1) to alter the signification of the scope of access `Inherited` and (2) to include a new scope `Dependent`. This new scope of access enables users to reuse – instead of duplicate – the scope of access that has been selected in a similar access need. See Table 3 and Listing 1 for a comprehensive description of the new scope `Dependent` and revision of the existing scope `Inherited`, and Fig. 2 for an example.

Table 3. Revised Scopes of Access

Scope of Access	Dependency	Inheritance
Keyword	`Dependent`	`Inherited`
Predicate	`hasAccessNeed`	`inheritsFromNeed`
Relationship	Links to access needs whose registered shape tree is referenced by the shape tree associated with the current access need	Links to another access need whose registered shape tree is the same as the current access need
Example	*Assignees depend on a project*	*Assignees inherit from contacts*

Listing 1. Proposed DataAuthorizationDependentShape

```
<DataAuthorizationDependentShape> {
  a [ interop:DataAuthorization ] ,
    interop:grantee               IRI   ,
    interop:registeredShapeTree   IRI   ,
    interop:satisfiesAccessNeed   IRI?  ,
    interop:accessMode            @<#AccessModes>+  ,
    interop:creatorAccessMode     @<#AccessModes>*  ,
  interop:scopeOfAuthorization    [ interop:Dependent ]  ,
  interop:dependsFromAuthorization IRI
}
```

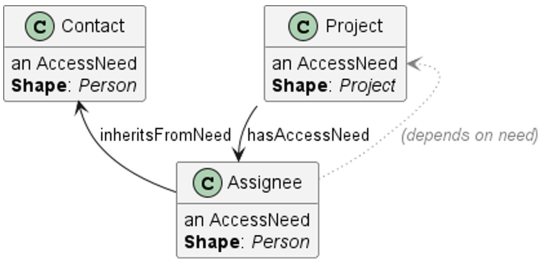


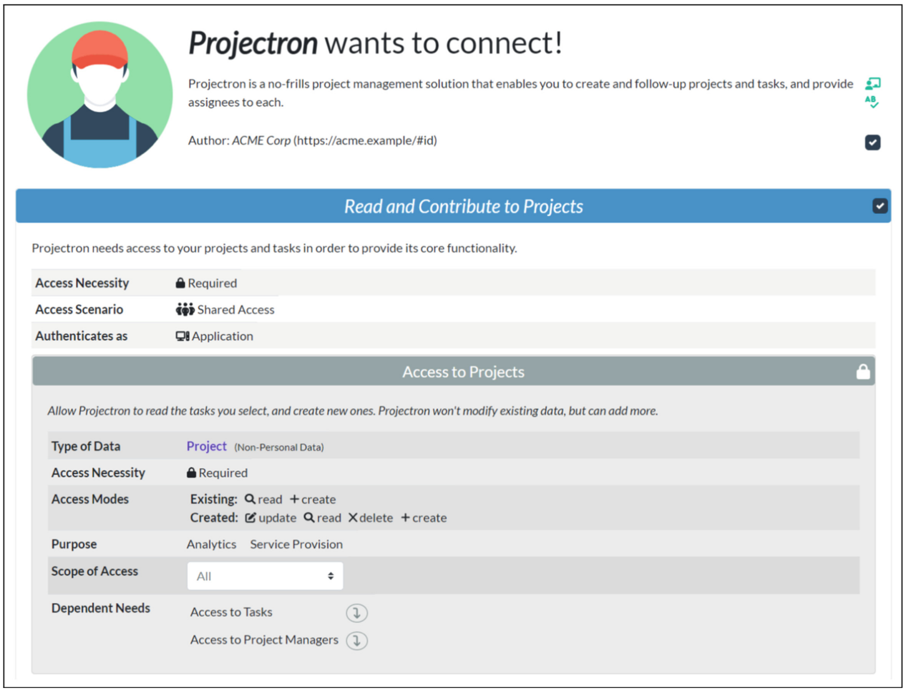
Fig. 2. Revised Scope of Access - Example

Table 4. INTEROP Shapes Suggested Predicates Changes Summary

Shape	Predicates Changes	
AccessNeedShape	+	interop:hasAccessNeed
	+	interop:hasAccessNeedDescription
	+	interop:hasPurpose
AccessNeedGroupShape	+	interop:hasAccessNeedGroupDescription
	–	interop:hasAccessDescriptionSet
AccessNeedDescriptionShape	+	skos:definition
DataRegistrationShape	+	ldp:contains
DataAuthorization-SelectedFromRegistryShape	–	interop:hasDataRegistration
SocialAgentShape	+	foaf:name
	+	foaf:givenName
	+	foaf:familyName

4.2 Implementation

The open source prototype implementation is a Vue.js web application with a Java/PostgreSQL back-end[5] available under a GNU GPL v3.0 license.


Projectron wants to connect!

Projectron is a no-frills project management solution that enables you to create and follow-up projects and tasks, and provide assignees to each.

Author: *ACME Corp* (https://acme.example/#id)

Read and Contribute to Projects

Projectron needs access to your projects and tasks in order to provide its core functionality.

Access Necessity	Required
Access Scenario	Shared Access
Authenticates as	Application

Access to Projects

Allow Projectron to read the tasks you select, and create new ones. Projectron won't modify existing data, but can add more.

Type of Data	Project (Non-Personal Data)
Access Necessity	Required
Access Modes	Existing: read +create Created: update read ✕delete +create
Purpose	Analytics Service Provision
Scope of Access	All
Dependent Needs	Access to Tasks Access to Project Managers

Fig. 3. Solid INTEROP Application Access Request User Interface

[5] **Source Code**: https://github.com/HBailly/solid-auth-ui/.

It has two goals: 1) implement the application access requests and authorization from the INTEROP specification, and 2) prototype our UI architecture for experiments enabling users to view and interact with application requests, and create corresponding authorizations. The scope does not include realizing an INTEROP authorization agent, nor enforcing the authorizations (although they are validated, see experiment).

The prototype dynamically loads access requests from a set of files stored on the server, validates the contents, and populates a Vue.js template. Users are presented with the populated template (Fig. 3), and can review the application access needs, then customize the authorizations they want to grant. On submission, the server stores the authorizations in a PostgreSQL database. All data exchanges are handled natively in RDF. The front-end depends on the `shex-codegen`[6] and `shex-methods`[7] Node.js libraries to parse and interact with the turtle files: All inputs/outputs are converted from and to RDF triples, and verified against ShEx models. The implementation uses the referenced RDF ontologies to fetch term descriptions and generate UI elements and tooltips.

`shex-codegen` is used during development to generate JavaScript objects (*sha-pes*) from a list of ShExes. These shapes can then be used at runtime with `shex-methods` to manipulate RDF nodes from the turtle files. To generate all shapes associated with the INTEROP specification, we used the ShEx provided along the specification. This revealed an important limitation in the `shex-codegen` library, as it does not support typed string data types. Only plain RDF literals are adequately supported. This issue was raised with the maintainer, and is under investigation[8]. It was overcome in the prototype by rewriting ShEx files to temporarily use `xsd:string` for the shape generation, then editing the generated files.

This work also revealed an issue with the `shex-methods` library, specifically with its dependency `rdflib.js`, which handles the creation and persistence of RDF resources. Several files from the INTEROP specification make use of escaped text sequences. While this is permitted under the RDF specification[9], it is currently not supported by the N3 parser in `rdflib.js`. The issue was reported to the maintainers of the library[10]. A temporary fix was applied in a local version of the library, which requires manual building into `shex-methods`, and the prototype[11].

This process enabled us to build a working prototype of our UI architecture that could dynamically create interactive UI elements from the turtle files provided by a test harness application.

[6] https://github.com/ludwigschubi/shex-codegen.
[7] https://github.com/ludwigschubi/shex-methods.
[8] https://github.com/ludwigschubi/shex-codegen/pull/139
https://github.com/ludwigschubi/shex-codegen/pull/140.
[9] https://www.w3.org/TR/turtle/.
[10] https://github.com/linkeddata/rdflib.js/pull/523
https://github.com/linkeddata/rdflib.js/pull/557.
[11] https://github.com/HBailly/solid-auth-ui/blob/main/pom.xml#942.

5 Evaluation

This section presents two evaluations of the prototype, verifying whether the design met the requirements **R5** and **R6**. The first evaluation was based on the Nielsen Heuristics and took place during the design and implementation phase of the prototype, while the second evaluation focused on the SUS, NPS and user feedback about the definitive version in a structured online experiment.

5.1 Demos and Usability Heuristics

The prototype was first demoed to participants of the *COST EU Workshop on Privacy Issues in Distributed Social Knowledge Graphs* (PIDSKG)[12]. Approximately 20 attendees were introduced to the research, shown the user interface, and were able to ask questions or raise concerns. The most frequently expressed concerns were about the ease of use, and the lack of an option to quickly select/deselect all optional access needs.

Next, the prototype was submitted to three preliminary evaluators from the ADAPT Research Centre, who were familiar with Linked Data and programming. They were tasked with evaluating the prototype using Nielsen's design heuristics. The evaluations took place remotely, in short sessions of 30 min, where the evaluators were also introduced to the research topic. They were then invited to use the UI and find violations of the heuristics (see Table 5). They raised issues regarding the vocabulary used in the interface, which was often complex and technical. They also pointed out several visibility issues with the appearance of buttons and relations between sections of the UI.

Table 5. Heuristics Breaches Identified Count

	Heuristic	Description	Severity				
			Minor	Moderate	Serious	Severe	Critical
1	Visibility	*Visibility of system status*	0	0	0	0	0
2	Information	*Match between system and real world*	2	2	1	0	0
3	Consistency	*Consistency and standards*	0	0	0	0	0
4	Recognition	*Recognition rather than recall*	2	1	0	0	0
5	Flexibility	*Flexibility and efficiency of use*	0	0	0	0	0
6	Minimalism	*Aesthetic and minimalist design*	0	0	0	0	0

Post-Evaluation UI Refinement. The UI was reworked partially to include cascading selection of access needs and other shortcuts. The INTEROP ontology was also locally edited, and the terms identified as too technical by the evaluators were updated.

5.2 SUS, NPS, Access Authorization Validity and User Feedback

The second evaluation was a formal experiment to validate the prototype usability and user-friendliness, that it met the requirements from Sect. 2, and

[12] https://cost-dkg.eu/.

empowered users to correctly create access authorizations. It was an asynchronous, opt-in experiment.

Participants were selected from two categories in two areas of expertise: experts in the Solid ecosystem and/or in GDPR legal requirements, and students in computing or in privacy law. The participants were invited from: the COST ACTION DKG Conference on SOLID 2022 attendees, the Solid community Forum, the Solid Data Interoperability Panel Gitter, LinkedIn, the DCU Master in Computing 2022, and personal communications. The experiment was freely available online: any person with the link could connect, complete the tasks and the questionnaires. A major power failure in the host site reduced participant numbers during the collection period.

The experiment itself was designed as follows: 1) Participants first completed a questionnaire on demographic data, relevant expertise and the Westin privacy classification questions. 2) Participants were then introduced to the research topic and provided with brief background, as well as links to further internet resources, short videos[13] introducing the user interface, and a glossary of terms. 3) Participants received guidelines on three directed tasks. During each task, they were requested to grant access authorizations corresponding to a gold standard, using the prototype. They were also presented with the situation they were asked to imagine themselves in (next paragraph). Participants conducted the three tasks sequentially in the same order. The task output and execution time were recorded. 4) Participants were invited to score the user interface using SUS and NPS questionnaires, and to provide feedback about their experience[14].

The three directed tasks placed the participant into the position of *Alice* as per our Use Case (Sect. 2) derived from the INTEROP specification. Alice is presented with the application's access request and must decide what access to authorize (*permissions*). Each task is a variation of the access request, and requires the user to select the same permissions each time. The variations are of increasing complexity: i) a simple access request with five access needs (*flat hierarchy*); ii) introduces dependency between the access needs, and similar needs within different dependencies (*nested hierarchy*); and iii) includes inheritance on top of the dependency (*nested hierarchy with inheritance*).

5.3 Results and Discussion

In total, there were **15** participants to the experiment: Most were male, aged less than 36, and all had at least a bachelor's degree. Using Jensen 2005's Westin classification questionnaire [13], we observed a slightly more privacy-concerned distribution of the participants (ordinarily 26-64-10% [16], 33.3-60-6.7% in the experiment). Many had previous experience with programming, and there were experts in Solid, INTEROP and/or GDPR privacy law.

Tasks: On average, it took between 2.5 and 4.5 min to complete a task. The observed the number of errors follows the increasing complexity of the tasks.

[13] https://github.com/HBailly/solid-auth-ui/tree/main/tutorials/.
[14] https://github.com/HBailly/solid-auth-ui/tree/main/docs/questionnaires.

The sharing of contacts proved to be the most challenging part of the directed tasks (78.5% average error rate). Moreover, none of the participants de-selected an optional access need during any of the directed tasks, even if they had a high desire for privacy (perhaps due to the low stakes or personal involvement in the tasks). Table 6 summarizes the authorizations crafted by the participants, and compares them with the gold standard requested by the tasks.

Table 6. User-granted Authorization Deviations from Gold Standard

Access Need	Gold Standard	OK	Authorization granted when deviating from gold standard[a]							Total
			All	All From Registry	Selected From Registry	All From Agent	Dependent	Inherited	None	
Version 1										
`need-project`	*All from Registry*	9 60.0%	2 13.3%	0 0%	3 20/0%	1 6.7%	0 0%	0 0%	0 0%	15
`need-task`	*All from Registry*	11 73.3%	2 13.3%	1 6.7%	1 6.7%	0 0%	0 0%	0 0%	0 0%	15
`need-contact`	*Selected From Registry*	10 66.7%	0 0%	2 13.3%	0 0%	0 0%	3 3.%	0 0%	0 0%	15
`need-account-details`	*All*	13 86.7%	0 0%	1 6.7%	0 0%	1 6.7%	0 0%	0 0%	0 0%	15
`need-credit-details`	*Selected From Registry*	11 73.3%	1 6.7%	3 20.0%	0 0%	0 0%	0 0%	0 0%	0 0%	15
Version 2										
`need-project`	*All from Registry*	10 66.7%	1 6.7%	0 0%	3 20%	1 6.7%	0 0%	0 0%	0 0%	15
`need-task`	*All from Agent*	10 66.7%	1 6.7%	2 13.3%	2 13.3%	0 0%	0 0%	0 0%	0 0%	15
`need-contact-project`	*None*	0 0%	3 10.3%	0 0%	7 0%	1 6.7%	4 26.7%	0 0%	0 0%	15
`need-contact-task`	*Selected From Registry*	5 33.3%	1 6.7%	1 6.7%	2 13.3%	2 13.3%	4 26.7%	0 0%	0 0%	15
`need-account-details`	*All*	11 73.3%	0 0%	4 26.7%	0 0%	0 0%	0 0%	0 0%	0 0%	15
`need-credit-details`	*Selected From Registry*	13 86.7%	0 0%	2 13.3%	0 0%	0 0%	0 0%	0 0%	0 0%	15
Version 3[b]										
`need-project`	*All from Registry*	7 50%	5 35.7%	0 0%	1 7.1%	1 7.1%	0 0%	0 0%	0 0%	14
`need-task`	*All from Agent*	7 50%	3 21.4%	2 14.3%	2 14.3%	0 0%	0 0%	0 0%	0 0%	14
`need-contact`	*None*	0 0%	4 28.6%	0 0%	9 64.3%	1 7.1%	0 0%	0 0%	0 0%	14
`need-contact-project`	*None*	3 21.4%	0 0%	0 0%	1 7.1%	1 7.1%	0 0%	9 64.3%	0 0%	14
`need-contact-task`	*Selected From Registry*	1 7.1%	1 7.1%	0 0%	1 7.1%	1 7.1%	0 0%	10 71.4%	0 0%	14
`need-account-details`	*All*	10 71.4%	0 0%	4 28.6%	0 0%	0 0%	0 0%	0 0%	0 0%	14
`need-credit-details`	*Selected From Registry*	11 78.6%	1 7.1%	2 14.3%	0 0%	0 0%	0 0%	0 0%	0 0%	14

[a]A deviation value of 1 with the same scope as the gold standard (e.g. *All from Registry*) indicates that a participant selected the right type of scope (*All from Registry*), but a wrong value (an incorrect registry).
[b]One of the participants skipped the version 3 task, and is thus excluded from the analysis of this version.

Usability: Overall, the prototype only scored over 37.67 for more than 50% of the participants of the survey using the System Usability Scale (the details of which are presented in Table 7), which indicates *poor* usability. This score was independent of the nationality, level of education, expertise, or Westin privacy classes. Participants were primarily concerned by the technicality, complexity and cumbersomeness of the UI, which directly exposed the complex application resource requests. Most participants also found that the process was too long, but more than half thought that it was worthwhile. The participants with the most negative attitude towards the UI were also those finding the duration excessive.

Table 7. SUS Questionnaire

Question	*Strongly disagree*	*Disagree*	*Neutral*	*Agree*	*Strongly agree*
I think that I would like to use this system frequently	4 26.7%	2 13.3%	4 26.7%	5 33.3%	0 0%
I found the system unnecessarily complex	1 6.7%	4 26.7%	3 20%	3 20%	4 26.7%
I thought the system was easy to use	4 26.7%	3 20%	4 26.7%	4 26.7%	0 0%
I think that I would need the support of a technical person to be able to use this system	1 6.7%	4 26.7%	1 6.7%	9 60%	0 0%
I found the various functions in this system were well integrated	0 0%	2 13.3%	5 33.3%	7 46.7%	1 6.7%
I thought there was too much inconsistency in this system	2 13.3%	7 46.7%	5 33.3%	0 0%	1 6.7%
I would imagine that most people would learn to use this system very quickly	6 40%	4 26.7%	1 6.7%	4 26.7%	0 0%
I found the system very cumbersome to use	0 0%	3 20%	3 20%	7 46.7%	2 13.3%
I felt very confident using the system	7 46.7%	2 13.3%	4 26.7%	1 6.7%	1 6.7%
I needed to learn a lot of things before I could get going with this system	0 0%	6 40%	2 13.3%	5 33.3%	2 13.3%

Participants were divided when declaring their satisfaction with the prototype, as highlighted by the Net Promoter Score (5.4 on average - *detractor* stance), and the propensity to use it again (*Positive*: 5; *Neutral*: 4; *Negative*: 6). Noticeably, participants belonging to the fundamentalists were more positive about the prototype (*average*: 8; *count*: 5), whereas participants belonging to the pragmatic class were more likely to be detractors (*average*: 3.67; *count*: 9)[15]. Participants with a strong Solid or interop background also had a higher Net Promoter Score than the others.

When asked what version of the task they found the most appropriate/informative, participants either chose the first (*flat hierarchy*) (in majority) or the third version (*nested hierarchy with inheritance*). We found that the dividing line was also the Westin classification of participants: pragmatic users largely preferred the flat version (80%), while fundamentalists preferred the nested version (40%) or none (40%) (See Footnote 15). The most frequent reasons given by the pragmatic users were that it was simpler and offered less choice, hence less

[15] There was not enough data to evaluate the unconcerned.

confusion, and that the dependency and inheritance mechanisms were not given away by the prototype. On the contrary, 60% of the fundamentalists praised the hierarchy, as it allowed them to better assess the consequences of granting access (the other 40% did not notice the difference). More generally, participants were united in pointing out how there was too much information displayed, with many technical terms, whereas they would have preferred a more incremental process, perhaps with a wizard, or a multilayered UI.

Finally, participants rated positively the visual and textual aids of the prototype. No participant questioned the actual contents of the shapes requested, even the fundamentalists, and found them moderately to very informative. Participants did not agree on the utility of the GDPR purpose added in this design: participants without little knowledge of privacy law and fundamentalists found them informative, whereas pragmatics and people with declared expertise did not notice them. Interestingly, 67% of participants rated the access modes informative. When investigating whether they knew the difference between the modes for existing instances, and instances created by the application after it was authorized, only the participants with Solid expertise could tell correctly.

6 Conclusion

This paper prototyped the Solid INTEROP specification with an original UI to enable users to review application access requirements and define access and usage authorizations. We identified areas for improvement in the INTEROP draft specification, both in terms of Semantic Web best practice and to better support dynamic access control UI generation (Sect. 4). We showed that Solid is not using best practices in policy interfaces (Sect. 3) by relying on text editors, and that usage control remains an open issue under INTEROP. Our dynamic UI prototype based on Semantic Web tools identified limitations with current Semantic Web tools and libraries. Usability evidence collected also suggested that the fine-grained and expressive controls of INTEROP will require careful UI design to avoid overwhelming users (Sect. 5). The main reasons were the length and complexity of the requests and the terminology used. Participants felt in control of their data, but repeatedly failed in tasks to correctly generate authorizations and could not explain what access modes they granted. This is a limited study (n = 15), mainly conducted with experts, and a broader study should be conducted next.

Acknowledgement. This research had the financial support of Science Foundation Ireland under the ADAPT Centre for AI-driven Digital Content Technology, SFI Research Centres Programme (Grant 13/RC/2106_P2). For the purpose of Open Access, the authors have applied a CC-BY public copyright license to any author accepted manuscript version arising from this submission.

References

1. Beznosov, K., et al.: Usability meets access control, vol. 2807, p. 73. ACM Press (2009). https://doi.org/10.1145/1542207.1542220
2. Bingham, J., Prud'Hommeaux, E., Pavlik, E.: Solid Application Interoperability. W3C Editor's Draft (2022). https://solid.github.io/data-interoperability-panel/specification/
3. Bosquet, M.: Access Control Policy (ACP). Solid Editor's Draft (2022). https://solid.github.io/authorization-panel/acp-specification/
4. Brooke, J.: SUS: a quick and dirty usability scale. In: Usability Evaluation In Industry. CRC Press (1996). Chap. Off-the-Shelf Evaluation Methods. https://doi.org/10.1201/9781498710411-35
5. Cao, X., Iverson, L.: Intentional access management: making access control usable for end-users. In: Proceedings of the Second Symposium on Usable Privacy and Security, SOUPS 2006, Pittsburgh, Pennsylvania, USA, pp. 20–31. Association for Computing Machinery (2006). https://doi.org/10.1145/1143120.1143124
6. Capadisli, S., Berners-Lee, T.: Web Access Control. Version 1.0.0. Editor's Draft (2022). https://solid.github.io/web-access-control-spec/
7. Capadisli, S., et al.: Solid Protocol. Version 0.9.0 (2021). https://solidproject.org/TR/protocol
8. Esteves, B.: Solid ODRL access control Policies Editor. GitHub (2022). https://github.com/besteves4/solid-sope
9. Esteves, B., et al.: Using the ODRL profile for access control for solid pod resource governance. In: Extended Semantic Web Conference (ESWC) (2022). https://doi.org/10.5281/zenodo.6614777
10. Hamid, E., Jaafar, A., Choo, A.M.: A review of 'human-computer interaction' influence to home network. Jurnal Teknologi **75**, 21–27 (2015). https://doi.org/10.11113/jt.v75.5038
11. Iannella, R., Villata, S.: ODRL Information Model. Version 2.2. W3C Recommendation (2018). https://www.w3.org/TR/odrl-model
12. Inrupt Inc., Access Policies: Universal API (2022). https://docs.inrupt.com/developer-tools/javascript/client-libraries/tutorial/manage-access-policies/
13. Jensen, C., Potts, C., Jensen, C.: Privacy practices of Internet users: self-reports versus observed behavior. Int. J. Hum.-Comput. Stud. **63**(1–2), 203–227 (2005). https://doi.org/10.1016/j.ijhcs.2005.04.019
14. Kirrane, S., De Vos, M., Padget, J.: ODRL Regulatory Compliance Profile. Version 0.2. W3C Unofficial Draft (2020). https://ai.wu.ac.at/policies/orcp/regulatory-model.html
15. Kirrane, S., Mileo, A., Decker, S.: Access control and the resource description framework: a survey. In: Grau, B.C. (ed.) Semantic Web 8, pp. 311–352 (2016). https://doi.org/10.3233/SW-160236
16. Kumaraguru, P., Cranor, L.: Privacy indexes: a survey of Westin's studies. Technical report, Carnegie Mellon University, Pittsburgh, PA (2005). https://www.cs.cmu.edu/ponguru/CMU-ISRI-05-138.pdf
17. Liu, Y., Osvalder, A.-L., Karlsso, M.A.: Considering the importance of user profiles in interface design. In: User Interfaces, p. 270 (2010). https://doi.org/10.5772/8903
18. Mansour, E., et al.: A demonstration of the solid platform for social web applications. In: Proceedings of the 25th International Conference Companion on World Wide Web, WWW 2016, Companion, pp. 223–226. ACM Press, New York (2016). https://doi.org/10.1145/2872518.2890529

19. Meier, Y., Schäwel, J., Krämer, N.C.: The shorter the better? Effects of privacy policy length on online privacy decision- making. Media Commun. **8**, 291–301 (2020). https://doi.org/10.17645/mac.v8i2.2846
20. Nielsen, J.: Thinking Aloud: The #1 Usability Tool. Nielsen Norman Group (2012). https://www.nngroup.com/articles/thinking-aloud-the-1-usability-tool/
21. Official Journal of the European Union. General Data Protection Regulation (2016/679). Brussels (2016). http://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679
22. Pandit, H.J., et al.: Creating a vocabulary for data privacy. In: Panetto, H., Debruyne, C., Hepp, M., Lewis, D., Ardagna, C.A., Meersman, R. (eds.) OTM 2019. LNCS, vol. 11877, pp. 714–730. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-33246-4_44
23. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: OOPS! (OntOlogy Pitfall Scanner!): an on-line tool for ontology evaluation. Int. J. Semant. Web Inf. Syst. (IJSWIS) **10**(2), 7–34 (2014)
24. Rainie, L., Duggan, M.: Americans' Opinions on Privacy and Information Sharing. Pew Research Center (2016). https://www.pewresearch.org/internet/2016/01/14/privacy-and-informationsharing/
25. Reichheld, F.F.: The One Number You Need to Grow. Growth Strategy (2003). https://hbr.org/2003/12/the-one-number-youneed-to-grow
26. Rissanen, E.: eXtensible Access Control Markup Language (XACML). Committee Draft 03. Version 3.0. Oasis (2010). http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cd-03-en.pdf
27. Sandhu, R., Park, J.: Usage control: a vision for next generation access control. In: Gorodetsky, V., Popyack, L., Skormin, V. (eds.) MMM-ACNS 2003. LNCS, vol. 2776, pp. 17–31. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45215-7_2
28. Thomas, R.K., Sandhu, R.S.: Conceptual foundations for a model of task-based authorizations. In: Proceedings of the Computer Security Foundations Workshop, pp. 66–79 (1995). https://doi.org/10.1109/CSFW.1994.315946
29. Vaniea, K., et al.: Access control policy analysis and visualization tools for security professionals. In: SOUPS Workshop on Usable IT Security Management (USM) 2008, Pittsburgh, PA, USA, pp. 7–15 (2008). https://cups.cs.cmu.edu/soups/2008/USM/vaniea.pdf
30. Verborgh, R.: Re-decentralizing the Web, for good this time. In: Seneviratne, O., Hendler, J. (eds.) Linking the World's Information: A Collection of Essays on the Work of Sir Tim Berners-Lee. ACM (2022). https://ruben.verborgh.org/articles/redecentralizing-the-web/
31. Zaki, M., et al.: The Fallacy of the Net Promoter Score: Customer Loyalty Predictive Model. University of Cambridge (2016). https://cambridgeservicealliance.eng.cam.ac.uk/system/files/documents/2016OctoberPaper_FallacyoftheNetPromoterScore.pdf