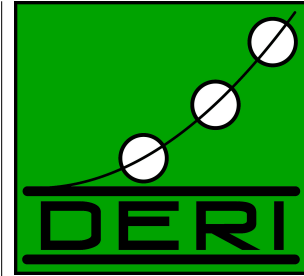


DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE



DISCOVERING RESOURCES ON THE WEB

Jürgen Umbrich Michael Hausenblas
Phil Archer Eran Hammer-Lahav
Erik Wilde

DERI TECHNICAL REPORT 2009-08-04
AUGUST 2009

DERI – DIGITAL ENTERPRISE RESEARCH INSTITUTE

DERI Galway
IDA Business Park
Lower Dangan
Galway, Ireland
<http://www.deri.ie/>

DERI TECHNICAL REPORT

DERI TECHNICAL REPORT 2009-08-04, AUGUST 2009

DISCOVERING RESOURCES ON THE WEB—A COMPARISON OF DISCOVERY MECHANISM FOR THE WEB OF DATA AND THE WEB OF DOCUMENTS

Jürgen Umbrich¹ Michael Hausenblas¹ Phil Archer²
Eran Hammer-Lahav³ Erik Wilde⁴

Abstract. Discovering information on the Web in a scalable and reliable way is an important but often underestimated task. Research on discovery itself is quite a young field. Hence, to date not many Web-compliant discovery mechanism exist. Firstly, we introduce a layered *Abstract Discovery Model* and discuss its features. Then, driven by use cases and requirements, we review three promising discovery proposals in the context of the Web of Data and the Web of Documents: XRD, POWDER, and void.

Keywords: discovery, Web of Data, Web of Documents, XRD, POWDER, void.

DISCLAIMER We note that this is work in progress. Most of the IETF drafts (LRDD, and its companioning drafts) are moving targets and at time of publication of the Technical Report at hand they are subject to discussions on various IETF and W3C lists. We will publish a new version of this report, once the respective drafts have been published as RFCs.

¹DERI, National University of Ireland, Galway, Ireland. {juergen.umbrich, michael.hausenblas}@deri.org

²i-sieve technologies, Athens, Greece. phil@i-sieve.com

³Yahoo!, United States. eran@hueniverse.com

⁴School of Information, UC Berkeley, United States. dret@berkeley.edu

Acknowledgements: This work has been carried out in the Linked Data Research Centre (LiDRC) and the authors are thankful for the many valuable discussions with LiDRC members and peer researchers.

Copyright © 2009 by the authors

Contents

1	Introduction	1
2	Use Cases and Requirements	2
2.1	Use Cases	2
2.1.1	UC1: End-User Content Customisation	2
2.1.2	UC2: Service Provider	2
2.1.3	UC3: Access Control	2
2.1.4	UC4: Optimising Access	2
2.1.5	UC5: Selective Browsing	3
2.1.6	UC6: Federated SPARQL queries	3
2.2	Requirements	3
3	Abstract Discovery Model	4
3.1	Reference Layer	4
3.2	Description Layer	6
3.2.1	XRD	6
3.2.2	POWDER	6
3.2.3	voID	6
4	Comparison and Mapping	7
4.1	Comparison	7
4.2	Mapping to the ADM	11
5	Discussion and Outlook	12

1 Introduction

One of the tenets of the Web at a large—both the Web of Data and the Web of Documents—is that *simplicity enables an initial implementation of a distributed system, while extensibility allows to avoid getting stuck forever with the limitations of what was deployed* [FT02]. We are now in a phase where we witness many new applications of Web technologies and resources, such as in the mobile domain, microblogging, and so forth. While we have made sound advances in terms of rich content, interactivity etc., resource discovery is still a neglected area.

Discovering information on the Web in a scalable and reliable way is often underestimated. One reason discovery is nowadays neglected might be that the benefits are not so apparent. We understand, that in many cases we can reduce costs and save time if a description of a Web resource is available. Such a resource description typically contains machine-readable information about a single resource or a group of resources, allowing agents to learn how to use the resource or how to optimise the interaction with a resource. Agents do not have to inspect the content (often referred to as “content sniffing”) to decide if the resource is relevant w.r.t. certain requirements, and hence preventing unnecessary HTTP requests. On the other hand, descriptions can be used to guide and advice agents about unknown resources.

Let us assume that a fictional start-up company wants to build a search engine for under-aged. We assume that images and videos are published under the Creative Commons License¹. Due to the targeted user group the content must not glorify violence, or other types of mature content. In this setup, resource descriptions can support gathering of relevant audio-visual material from the Web, such as regarding age restrictions or licensing of the resources. A crawler can—without the need to perform content sniffing—decide, if a certain document is relevant. Additionally, the start-up is able to offer an attractive service for their users by providing resource descriptions: the Web site can render the access information, mobile applications can select videos and images that are usable on the respective platform, and eventually the costly transfer of data can be reduced. The questions the start-up apparently has to answer are: How to discover the resource descriptions? What resource descriptions can be found on the Web? Which discovery methods and resource descriptions should be used based on the scenario?

When we talk about discovery, we differentiate [HL09b] between *descriptor discovery* and *service discovery*. Both types attempt to associate capabilities with resources, but they approach it from opposite ends. While service discovery centers on identifying the location of qualified resources, typically finding an endpoint capable of certain capabilities, descriptor discovery begins with a resource, trying to find which capabilities it supports. While our work deals with descriptor discovery, it is worthwhile to note that the two discovery types are closely related and are typically used in tandem. We note that the question “Which discovery methods and resource descriptions should be used based on the scenario?”, cannot be answered in general. Therefore, we compare the potential resource description candidates based on a selection of use cases ranging from end-user content customization to query optimization.

The remainder of the work is organised as follows: in Section 2 we introduce our use cases and derive requirements for resource discovery mechanisms on the Web. Then, in Section 3 we introduce the layered *Abstract Discovery Model* and introduce candidate technologies for each layer. In Section 4 we compare selected discovery methods and map them to our Abstract Discovery Model. Eventually, we discuss the results and conclude in Section 5.

¹<http://creativecommons.org/>

2 Use Cases and Requirements

We present six different use-cases which illustrate how applications benefit from information about resources. Further, we derive requirements that a discovery mechanism must meet.

2.1 Use Cases

There are many application areas that can potentially benefit from resource descriptions. Apparently, the list is by far not complete and we do not claim to cover all possible scenarios.

2.1.1 UC1: End-User Content Customisation

Homer got a shopping list from Marge and visits an online shop. The personal shopping agent knows from his profile that he is a Duff beer fan and hence shows him only offers in relation to this sort of beer. The customisation is possible because Homer has declared in his profile that he prefers Duff beer and the online shop has marked up certain sections with “related to Duff beer”.

2.1.2 UC2: Service Provider

Anouk is running a vertical search engine for table soccer (similar to the functionality that is provided by Yahoo!’s BOSS² or Google’s CSE³). Rather than filtering the entire results, she wants to crawl up-front only sites that provide information about table soccer (such as equipment or tournaments). How can she find these sites? How can she make sure to index only relevant material?

2.1.3 UC3: Access Control

Roger is a business interface manager in an international operating firm. To get an overview of the current situation in the companies local operating units he needs to have access to the local business reports, formatted in the Extensible Business Reporting Language (XBRL). Now, first he has to find out where the XBRL documents are located and if he has access to them. In case he doesn’t find them or can’t access them, he wants to know whom of the local operating unit he has to contact to gain access to the documents.

2.1.4 UC4: Optimising Access

Silke has a mobile phone with limited credit. Her subscription includes 100MB download volume per month and she plans not to overdraw this limit. On the other hand she likes to visit sites such as YouTube and find out about what is going on in her home city. Given that she consumes a certain amount of movies, how can she exploit the download limit in an optimal way regarding resolution, topics, bandwidth, etc.?

²<http://developer.yahoo.com/search/boss/>

³<http://www.google.com/cse>

2.1.5 UC5: Selective Browsing

Jim likes to preview outgoing links such as provided by `snap.com`. However, beside the sheer content, he does not learn much about the target of the link. He would like to see for example if the content is suited for his needs, or if the content is restricted.

2.1.6 UC6: Federated SPARQL queries

Axel, a researcher, wants to test his optimisations for a SPARQL query engine. He tested his optimisation with local SPARQL endpoints containing information about a research community, such as publications, authors, conferences and institutes. For his evaluation he wants to show that his optimisations gain the same performance increase if he uses public available SPARQL endpoints. He needs a way to discover online SPARQL endpoints and also what kind of information he can query.

2.2 Requirements

Based on the use cases presented in the previous section, and through the experiences we have gathered in the specification process [HL09b, ASP09, ACHZ09] we require a discovery mechanism that it **MUST** be:

- **Self declarative**, that is to allow resources to declare the availability of descriptor information and its location (UC1-UC6);
- **Direct accessible** without interacting with the resource itself. Before the content of a resource is accessed, the client should have a way to obtain the resource descriptor without accessing the resource itself (UC2, UC4, UC5);
- **Compliant with the Web Architecture**, that is, the discovery mechanism is based on URIs and HTTP (UC1-UC6);
- **Scale** to the size of the Web, that is, processing a descriptor should work no matter how many new resources are added to the RDs (esp. UC2);
- **Extensible** with new elements or properties to describe resources in more detail (esp. UC1, UC3, UC6);
- **Granular**, that is, the descriptor must allow the description of a single resource or a group of resources, identified by patterns (UC1-UC6).

Further, a discovery mechanism **SHOULD** be:

- **Simple**, that is, easy to use and straightforward to parse and process;
- **Non-proprietary**, that is, an open standard, either community-agreed or driven by a standardisation body such as OASIS, IETF or W3C;
- **Supported**, that is, have a noticeable community support, software libraries for processing, tutorials and documentation.

3 Abstract Discovery Model

In order to talk technology-agnostic about discovery, we introduce the *Abstract Discovery Model* (ADM) in the following. Figure 1 shows the ADM with its three layers:

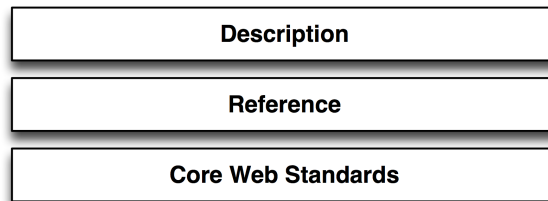


Figure 1: The Abstract Discovery Model.

- The bottom-most layer is taken by the *core Web standards*: Uniform Resource Identifiers (URIs) [BLFM05] for identifying things of interest, and the Hypertext Transfer Protocol (HTTP/1.1) [FGM⁺99] for accessing things and description of things;
- On top of this, the *reference* layer resides. One has to be able, given a resource’s URI, to locate its respective description;
- The top-most layer is the *description* layer, where the terms for describing resources are defined.

We understand that the core Web standards URI and HTTP need no further introduction. Further, we note that our research—based on our requirements stated above—focuses on solutions that are compliant to the REST architectural style [FT02]. In the following we will hence shed a light on the upper two layers of the ADM, the reference and the description layer.

3.1 Reference Layer

The problem of finding descriptions about resources is not just as old as the Web, it reaches back all the way to the very beginnings of any form of knowledge or information management. In pre-electronic times, it was solved with catalogues and look-up systems, which had obvious scalability and categorization problems. With the advent of hypertext/hypermedia, it became apparent that any system for this kind of system needs some way to combine (and thus locate) the content, and the linking overlay for the content. In hypermedia systems and in particular distributed and open hypermedia systems, the idea of *link services* [DWC00] became popular, which in a simplified view accept resource identifiers as input, and reply with links (or pointer to links) as output. It is important to notice that this idea of a link service in no way “solves” the problem of how to discover links (unless there is an assumption that there is a single centralized link service); it simply shifts the problem from the *discovery of links* to the *discovery of link services*. For some applications, this additional level of indirection may be sufficient for the intended application scenario, but in every fully distributed system, it is conceptually impossible to have complete knowledge of “all links” or “all link services” that exist within the scope of the system.

Since the problem of locating (discovering) resources is a problem inherent to many hypermedia scenarios, a considerable amount of previous work can be found in that area. One example is the *XML Linking Language (XLink)* [DMO01], which was an attempt to define a vocabulary for using links in XML documents. Interestingly, XLink also has a format for “out-of-line” links, which means that it is possible to separate content and links. This possibility is a rather direct segway leading to the idea of building services around collections of out-of-line XLinks, and examples for this can be found in specific systems such as *Xspect* [CHB03] and *XLinkProxy* [CFRV02], and in other approaches exploring the general idea of “linkbases” [BE05, LW01, MMG07].

In the context of the Abstract Discovery Model, we have identified **Link-based Resource Descriptor Discovery** (LRDD) [HL09b] as our least common denominator and hence will use it as the core technology. LRDD defines methods to express the location of the resource descriptor as a link relation. The association of a descriptor document with the resource it describes is declared using the **describedby** link relation type as defined and registered in POWDER⁴:

The relationship A “describedby” B asserts that resource B provides a description of resource A. There are no constraints on the format or representation of either A or B, neither are there any further constraints on either resource.

LRDD proposes three methods which together satisfy our requirements. We note that the descriptor location (URI) is a function of the resource URI: each of the following three methods is performed by using the resource URI to identify its descriptor:

- The <LINK> element method is limited to resources with an available markup representation supporting the <LINK> element, such as (X)HTML and Atom. This method requires the retrieval of a resource representation. *Example:*

```
<LINK href="http://example.com/resource;about"
      rel="describedby" type="application/powder+xml">
```

- The HTTP Link: header method is limited to resources for which an HTTP GET or HEAD request returns a 2xx, 3xx, or 4xx HTTP response. This method uses the Link header defined in [Not09] and requires the retrieval of a resource representation header. *Example:*

```
Link: <http://example.com/resource;about>; rel="describedby";
      type="application/powder+xml"
```

- The “well-known location” method. This method is available for any resource identified by a URI whose authority supports the well-known location as defined in [NHL09]. It does not require obtaining a representation of the resource, and operates solely using the resource URI. The link relation between the resource URI and the descriptor URI is obtained by using a template contained in an XRD document in the `/.well-known/` directory. *Example:*

```
Link-Pattern: <{uri};about">; rel="describedby";
              type="application/powder+xml"
```

⁴<http://www.w3.org/TR/powder-dr/#semLink>

3.2 Description Layer

As described above, the description layer is the place where the terms for describing resources are defined. Depending on the complexity of the use case and the targeted environment (rather a document-based approach vs. a graph based approach), one might prefer different languages to express these terms.

We have identified three candidates for this layer, based on our work in this area. We will introduce these technologies in the following and note that we intentionally did not take other, related approaches such as UDDI⁵ or XRDS 2.0 [WRC⁺08] into account, as they do not fulfil our requirements.

Eventually, we note that in the *W3C Architecture of the World Wide Semantic Web* Task Force⁶ we are currently working on developing HTTP semantics⁷; this might turn out to be a promising addition to the below described formats.

3.2.1 XRD

Extensible Resource Descriptor (XRD) [HL09a]⁸ is a simple XML-based schema for describing resources. The schema has a two-sections architecture where one describes the resource itself and the other describes its relationships to other resources. For example, the self describing section can include information such as which API version the endpoint supports, while the relationships section can indicate where to find the address book of the person associated with the resource.

3.2.2 POWDER

The *Protocol for Web Description Resources* (POWDER) [ASP09] provides a mechanism to describe and discover Web resources and helps users to decide whether a given resource is of interest. All descriptions are explicitly attributed to their author. There are two varieties of POWDER: an RDF/OWL-based, semantically rich, variety called POWDER-S and a simpler, XML-based version, that transports RDF predicates and objects that can be applied to defined groups of resources. The XML-based version is intended as the primary transport mechanism for so called Description Resources and includes a number of syntactic shortcuts designed to ease deployment. POWDER-S can be generated automatically from POWDER.

3.2.3 void

The *Vocabulary of Interlinked Datasets* (void) [ACHZ09] is an RDF-Schema vocabulary and a set of instructions that enables the discovery and usage of linked datasets. A dataset is a collection of data, published and maintained by a single provider, available as RDF, and accessible, for example, through dereferenceable HTTP URIs or a SPARQL endpoint.

⁵http://www.uddi.org/pubs/uddi_v3.htm

⁶<http://esw.w3.org/topic/AwwswHome>

⁷<http://www.w3.org/2001/tag/awwsw/http.owl>

⁸<http://www.hueniverse.com/hueniverse/2009/03/xrd-sneakpeek>

4 Comparison and Mapping

We compare the three resource descriptions methods (XRD, POWDER, voiD) as described above in the contexts of general characteristics and our requirements of the discovery mechanism in the following. Based on the comparison we discuss the suitability of each method for our use cases. Finally, we map the discovery methods to our *Abstract Discovery Model*.

4.1 Comparison

In Table 1 we compare *POWER*, *XRD* and *voiD* regarding our requirements and additionally list some general characteristics.

	XRD	POWDER	voiD
Requirements			
Self-Declarative	yes	yes	yes
Direct Access	partially	partially	partially
Web-Compliance	yes	yes	yes
Scale	yes	yes	yes
Extensible	possible	possible	possible
Granularity	single/group	single/group	entire datasets
Parser/Processor	available	available	available
Non-proprietary	yes (OASIS)	yes (W3C)	yes (open community draft)
Support	yes	yes	yes
Characteristics			
Application Area	general purpose	general purpose	RDF datasets
Format	XML	XML/RDF	RDF
Adoption (1)	few	some	many
Instances	some examples	< 50 sites	~200 docs
Expressiveness (2)	10	21	16
Statistics (3)	no	no	yes
Consumer (4)	M	M+H	M

(1) ... number of implementations/supporters

(2) ... number of terms in the core vocabulary

(3) ... support for expressing statistics about items

(4) ... M is for machine, H is for human

Table 1: Comparison summary of Web discovery methods.

In the following we will discuss the details of our comparison summarised in Table 1.

Self-Declarative All candidates allow to declare the availability of descriptor information and its location.

Direct Access By design, XRD and POWDER can be discovered using LRDD. Hence, at least when the well-known location method is used, a client can obtain the location of the

descriptor directly. In contrast, void is currently specified⁹ to be either discoverable via `robots.txt` (\rightarrow `sitemap.xml` \rightarrow `void.ttl`) or via the `dcterms:isPartOf` property from a single resource. We are working on aligning void to play together with LRDD.

Compliant with the Web Architecture All candidates are based on HTTP and URIs (in case of void, additionally RDF).

Scale None of our candidates requires a central repository, hence we assume they are scalable to the size of the Web.

Extensible From our study we can confirm that all three candidates are extensible. XRD uses standard XML namespaces to extend XRD core schema with new elements at every level, and new attributes in every elements. Also, POWDER provides the possibility to add user specified attributes and relations. Eventually, since void is RDF-based, one can easily extend it by mixing-in own vocabularies or by extending the existing vocabulary.

Granularity XRD provides the “URITemplate” element to describe groups of URIs (e.g. by adding the suffix “photo” to refer to all URIs ending with “photo”). It is possible to refer to groups of URIs with regular expressions in POWDER. In case of void only entire datasets are on-target, hence the description of single resources is not easily possible.

Parser/Processor We were able to find parser implementations for all three methods. For POWDER¹⁰ and XRD¹¹ their exist XML parser with XPath expressions and for void¹² respective RDF parser are available. The same holds true for processors, though most of them are in an early stage.

Non-proprietary Since XRD is a OASIS working draft, POWDER is a W3C Proposed Recommendation, and void is an open community effort, all three fulfil the requirement to be group-reviewed, non-proprietary technologies.

Support Comparing all three proposals we found detailed documentation for POWDER, followed by void and XRD. One might see it as an advantage of void that already descriptors and many examples are available in the Linked Data cloud¹³, which can be used as a good starting point for adopters.

Further, we provide some information about other characteristics of our selected discovery methods:

Application Area The application area of void is to describe RDF data sets and how they are interlinked within each other. POWDER and XRD provide machine and human readable information to serve various purposes for individuals or organizations. Reaching from the description about the topics of the resource to complex scenarios with authentication of clients, access restriction and license issues.

⁹http://rdfs.org/ns/void-guide#sec_5_Consuming_void

¹⁰<http://i-sieve.com/powdervalidator/index.shtml>

¹¹<http://github.com/willnorris/php-xrd/tree/master>

¹²<http://semanticweb.org/wiki/Void>

¹³<http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData>

Format All three candidates use machine readable formats. Since `voiD` is for describing RDF data sets, it is self-evident that RDF is used as a model. Because XRD and POWDER are supposed to server a general purpose their format of choice is XML, though POWDER sort of lives in both worlds (POWDER-S).

Adoption Compared to XRD and POWDER, `voiD` has the most adopters, currently. Especially, the linked data community makes extensive use of `voiD` (for example for federated query processing [CFM⁺09, ACHZ09]). Adopters of POWDER are currently in the enterprise realm (e.g., Deutsche Telekom, one of the leading European ISPs is using it).

Instances Studying the usage of the three candidates, we figured out that there exist around 200 `voiD` descriptions of RDF data sets. For POWDER the number of instances is less then 50 (T-Online is providing descriptor files for their domain and sub-domains¹⁴, which makes in total approximately 20 POWDER files hosted at `*.t-online.de`). We were not able to detect XRD files on the Web so far, other than exemplary files, which is mainly due to the fact that XRD is currently under specification.

Expressiveness Since all descriptor formats can be easily extended, we focus here only on the expressiveness of the candidates without user extensions. XRD is by design very light-weight. Users have only a limited vocabulary to describe resources, mainly terms to describe the author, expire date and of what kind of media types the resources are. In contrary, POWDER and `voiD` have defined vocabularies with around 20 terms, which makes them more expressive as XRD. However, POWDER users use the `descriptorset` element to describe a resource with either RDF vocabularies or plain text or they can use the `tag` element to tag resources with either URIs or plain text. This two terms makes it in our view more expressive then `voiD`.

Statistics So far, only `voiD` supports terms to describe statistical properties of a items (resources, classes, etc.) for (RDF) datasets.

Consumer XRD and `voiD` descriptors are—by design—consumed by machines. POWDER on the other hand was designed to be used by humans and machines (that is, assumes a human in the loop to take a final decision).

	XRD	POWDER	voiD
UC1: User Customisation	++	++	-
UC2: Service Provider	o	o	o
UC3: Access Control	++	+	-
UC4: Optimising Access	+	++	o
UC5: Selective Browsing	+	++	-
UC6: SPARQL Optimisation	o	-	++

Table 2: Applicability of the method regarding the use cases.

Finally, we analyse the three candidates against our use cases outlined above (Table 2).

¹⁴<http://dawsec.dicom.uninsubria.it/andrea/ppp/>

UC1: End-User Content Customisation To provide the “Duff beer” related information to Homer while he is doing the shopping for Marge the online shop either tag single resources or a group of resources with “related to Duff beer”. Based on the user profile of the customer the shop could either display just the group of resource URIs or disable the display of unrelated URIs. For either approaches POWDER and XRD are suitable. With POWDER the shop owner could use the “tag” element and with XRD he could use the “TYPE” element with a URI that represent the “related to Duff beer” preference.

UC2: Service Provider Anouk’s vertical search engine crawler may benefit from POWDER or XRD descriptions. A efficient way to crawl only sites that contain information about the topic table-soccer is to discover for each domain the global descriptor file and then extract only the related URLs. In the case that the descriptor file uses regular expressions, the crawler could utilise this patterns to extract only matching links. In case of RDF content, void seems most appropriate.

UC3: Access Control Our business interface manager Roger, who needs to gain access to the business reports described in XRBL needs a descriptor format that has the ability to describe the format and the contact person of the resources. Further, he requires information if he is authorised to access the resources. The most suitable candidate for this use case is XRD, which enables users to use OAuth for the authentication, the “MediaType” element to describe the format of the resources and the “URI” element to refer to the contact person. Rogers requirements could also be met with POWDER, but this would require to add some additional terms.

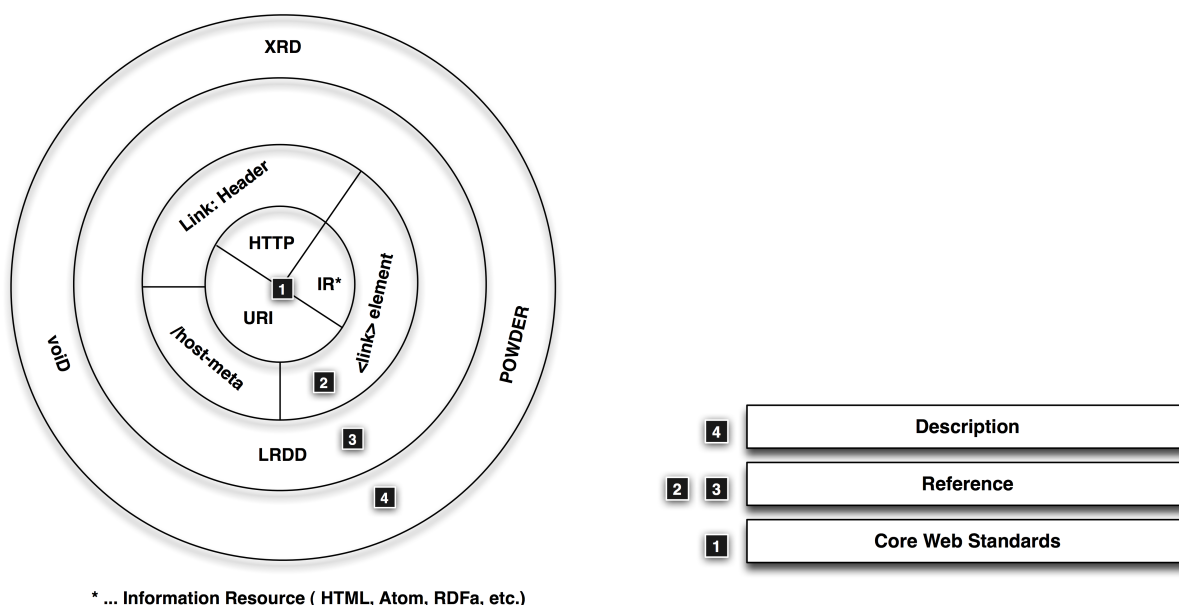
UC4: Optimising Access Silke wants to save time and money while surfing with her mobile the video site YouTube. For this task POWDER would be the best solution as defined in the W3C MobileWeb Best Practice guide [RM08].

UC5: Selective Browsing We can make Jim happy with a browser extension that discovers the resource descriptions of outgoing links in a HTML document and display additional information. Assuming that Jim likes to have human readable information and not URIs, we reckon POWDER is the most appropriate solution for this use case. The browser extension could utilise the free-text element to support Jim.

UC6: Federated SPARQL Queries Our assumption for Axel is to use void. Based on a SPARQL query against Semantic Web search engines to discover the void files he can find information about SPARQL endpoints and also about used RDF properties and classes. To some extent XRD could be also suitable, because of the possibility to describe service endpoints for resources, such as public APIs or, in this use case, SPARQL endpoints. Further, XRD can also describe the type of the resources, which could be the RDF classes or properties. However, since Axel uses already RDF and SPARQL queries void is more suitable to fulfil his requirements.

4.2 Mapping to the ADM

So far, we compared and discussed the characteristics and applicability of the three resource descriptor proposals. The last missing piece is the mapping between the discovery methods and our *Abstract Discovery Model*.



(a) Common Discovery Methods on the Web.

(b) Mapping of Discovery Methods on the Web to the *Abstract Discovery Model*.

Figure 2: Mapping of Discovery Methods on the Web to the *Abstract Discovery Model*.

Figure 2a depicts common discovery methods on the Web and their relation in terms of an onion ring diagram and Figure 2b how exactly this four layers map to the *Abstract Discovery Model*.

The centre of the diagram is divided into the core Web standards, the URI, the HTTP access protocol and information resources (IR).

The next outer layer of the diagram is divided into the three core reference methods defined by LRDD. As we can see in Figure 2a the URI is shared among the three mechanism, whereas, the HTTP `Link: header` requires the HTTP protocol for the discovery and the `<link> element` an information resource such as (X)HTML, Atom etc.

On top of the discovery mechanism is the linked-based resource discovery description protocol (LRDD). Both described layer map to the reference layer of the *Abstract Discovery Model*. The three candidates use LRDD for their discovery and thus, are placed at the most outer layer of the onion. Since they describe the resources the mapping is straightforward the the top layer of the ADM, the descriptions.

5 Discussion and Outlook

In this report we have motivated the importance of discovery and outlined use cases for it. We have proposed an Abstract Discovery Model (ADM) and reviewed three promising proposals, namely XRD, POWDER and void. We have provided a comprehensive comparison of the three candidates against our requirements and use cases and further, we explained how the discovery methods map to the ADM. Depending on the use cases we found that different discovery methods are best-suited. The findings of this comparison supports our hypothesis that the benefits of resource descriptions are not obvious so far and require further promotion and research.

We are following another reference alternative, as well: *Link Discovery in XML* (LDX). Here the main idea is that links are identified by XPath-based selectors (XSLT's pattern subset might be a good match here) that selected links are associated with roles, and that this mechanism can be applied to any XML-based media type. What would be the advantage of such a mechanism? A simple way to describe the links and link semantics of XML media types, the ability to build generic tools that can extract links from any XML-based media type, and the ability to better separate general REST functionality (what links can I follow in this resource?) from content-specific functionality (what does the content mean, and what does it mean to follow the links?). Generally, such a mechanism could become part of a REST toolbox, and there are advantages for both service providers and consumers.

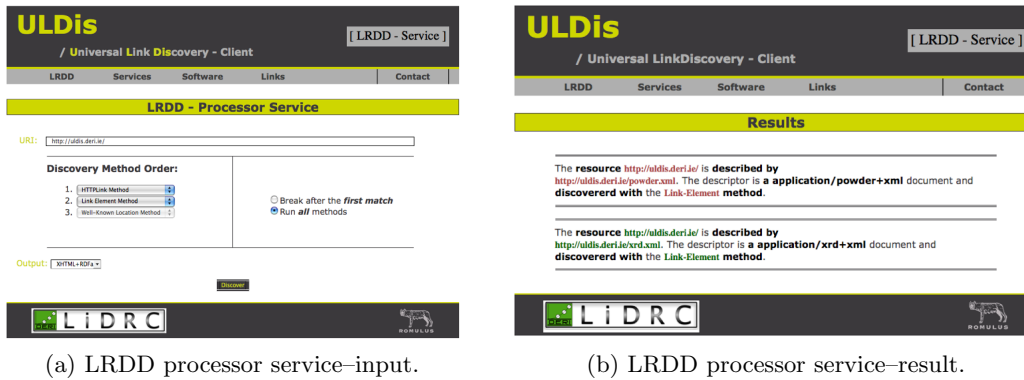


Figure 3: The LRDD processor, part of ULDis.

We are currently working on the *Universal Link DIScovery* client (ULDis)¹⁵, a LRDD-based discovery service, which allows—based on a mapping of the vocabulary terms¹⁶—to perform discovery independent of the underlying descriptor (XRD, POWDER, or void). That is, with ULDis one is able to consume all LRDD-based discovery technologies; the aim is to provide a unified and generic query on top of this service. In Fig. 3a the input form of the LRDD processor (part of ULDis) is shown, in Fig. 3b the respective result is depicted, available in different formats (currently in XHTML+RDFa and RDF/N3).

We note, that eventually only recently a discussion around extending LRDD to support the linked data case of 303 redirect has emerged¹⁷, which may well influence our future work.

¹⁵<http://uldis.der.i.e/>

¹⁶<http://rdfs.org/ns/aardv>

¹⁷<http://lists.w3.org/Archives/Public/www-tag/2009Jun/0102.html>

References

- [ACHZ09] K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets - On the Design and Usage of void, the 'Vocabulary of Interlinked Datasets'. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009.
- [ASP09] P. Archer, K. Smith, and A. Perego. Protocol for Web Description Resources (POWDER): Description Resources. W3C Proposed Recommendation 04 June 2009, POWDER Working Group, 2009.
- [BE05] François Bry and Michael Eckert. Processing link structures and linkbases on the web. In *Poster Proceedings of the 14th International World Wide Web Conference*, pages 1030–1031, Chiba, Japan, May 2005. ACM Press.
- [BLFM05] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic Syntax. Request for Comments: 3986, January 2005, IETF Network Working Group, 2005.
- [CFM⁺09] K. Cheung, H. Frost, M. Marshall, E. Prud'hommeaux, M. Samwald, J. Zhao, and A. Paschke. A Journey to Semantic Web Query Federation in Life Sciences. Technical report, 2009.
- [CFRV02] Paolo Ciancarini, Federico Folli, Davide Rossi, and Fabio Vitali. Xlinkproxy: External linkbases with xlink. In Ethan V. Munson, Richard Furuta, and Jonathan I. Maletic, editors, *Proceedings of the 2002 ACM Symposium on Document Engineering*, pages 57–65, McLean, Virginia, November 2002. ACM Press.
- [CHB03] Bent Guldbjerg Christensen, Frank Allan Hansen, and Niels Olof Bouvin. Xspect: Bridging open hypermedia and xlink. In *Proceedings of the Twelfth International World Wide Web Conference*, pages 490–499, Budapest, Hungary, May 2003. ACM Press.
- [DMO01] Steven J. DeRose, Eve Maler, and David Orchard. Xml linking language (xlink) version 1.0. World Wide Web Consortium, Recommendation REC-xlink-20010627, June 2001.
- [DWC00] David C. De Roure, Nigel G. Walker, and Leslie A. Carr. Investigating link service infrastructures. In *Proceedings of the 11th ACM Conference on Hypertext and Hypermedia*, pages 151–160, San Antonio, Texas, May 2000. ACM Press.
- [FGM⁺99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Request for Comments: 2616, June 1999, IETF Network Working Group, 1999.
- [FT02] R. Fielding and R. Taylor. Principled design of the modern Web architecture. *ACM Trans. Internet Technol.*, 2(2):115–150, 2002.
- [HL09a] E. Hammer-Lahav. Extensible Resource Descriptor (XRD) Version 1.0. Working Draft 01, 09 May 2009, OASIS XRI Technical Committee, 2009.
- [HL09b] E. Hammer-Lahav. Link-based Resource Descriptor Discovery (work in progress). Internet-Draft, 23 March 2009, IETF Network Working Group, 2009.

- [LW01] David Lowe and Erik Wilde. Improving web linking using xlink. In *Proceedings of Open Publish 2001*, Sydney, Australia, July 2001.
- [MMG07] Johannes Meinecke, Frederic Majer, and Martin Gaedke. Construction by linking: The linkbase method. In *Poster Proceedings of the 16th International World Wide Web Conference*, pages 1293–1294, Banff, Alberta, May 2007. ACM Press.
- [NHL09] M. Nottingham and E. Hammer-Lahav. Host metadata for the Web (work in progress). Internet-Draft, 10 February 2009, IETF Network Working Group, 2009.
- [Not09] M. Nottingham. Web Linking (work in progress). Internet-Draft, 17 April 2009, IETF Network Working Group, 2009.
- [RM08] J. Rabin and C. McCathieNevile. Mobile Web Best Practices 1.0. W3C Interest Group Note 31 March 2008, W3C Semantic Web Education and Outreach Interest Group, 2008.
- [WRC⁺08] G. Wachob, D. Reed, L. Chasen, W. Tan, and S. Churchill. Extensible Resource Identifier (XRI) Resolution Version 2.0. Committee Draft 14 April 2008, OASIS XRI Technical Committee, 2008.