# Interacting with the Web of Data through a Web of Inter-connected Lenses

Igor Popov[*]
School of Electronics and Computer Science
University of Southampton
SO17 1BJ, Southampton, UK
ip2g09@ecs.soton.ac.uk

m.c. schraefel
School of Electronics and Computer Science
University of Southampton
SO17 1BJ, Southampton, UK
mc@ecs.soton.ac.uk

Gianluca Correndo
School of Electronics and Computer Science
University of Southampton
SO17 1BJ, Southampton, UK
gc3@ecs.soton.ac.uk

Wendy Hall
School of Electronics and Computer Science
University of Southampton
SO17 1BJ, Southampton, UK
wh@ecs.soton.ac.uk

Nigel Shadbolt
School of Electronics and Computer Science
University of Southampton
SO17 1BJ, Southampton, UK
nrs@ecs.soton.ac.uk

## ABSTRACT

As a medium of structured information available on the Web, Linked Data is still hard to access for most end users. Current solutions facilitating end user access to Linked Data are either thought the use of *data-mapping* approaches, which allow configureable interfaces to be quickly deployed over pre-selected aggregations of Linked Data, or enable users themselves to browse the Web of Data through the use of generic data browsers. While the first approach is useful and promotes surfacing and easy repurposing of structured data it does little to promote the use of linkages to other, remote datasets. The second approach is much less useable for end users, however enables them to experience browsing a inter-connected Web of Data. In this paper we present *mash-point*, a framework that aims to provide a middle ground between both approaches. The approach treats data-centric applications as high-level lenses over the data, and allows selections of data to be pivoted between applications thus facilitating navigation. The paper presents an initial prototype and discusses both implications and challenges in terms of interaction and technology.

## Categories and Subject Descriptors

H5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Graphical user interfaces (GUI)*; H5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia—*User issues*

## General Terms

Design, Human Factors.

## Keywords

End-user Interaction, Linked Data, User Interface

[*]Corresponding author.

## 1. INTRODUCTION

End users engage in data-centric activities on the Web on a daily basis. Every time we view our news feeds on a social networking site or browse shopping items on a online commerce site, we are offered tools to browse, filter and find the data we need. Much of these data-centric interactions and tools, however, are limiting in one fundamental way - they confine us to browse and explore the only the data for which the tool was designed, denying the opportunity to refocus and find associated data on other web sites offering related data. One of the advocated advantages of linking data is that links that exist between remote datasets can be leveraged to effortlessly integrate and navigate to associated data in remote datasets. Despite this vision, we yet to experience interfaces where simple interactions allow end users to and navigate and find related data outside the current dataset, in essence denying them to truly experience a *Web* of Data.

Currently, three approaches are adopted to surface and allow casual end users to interact with and explore Linked Data. The first approach is the obvious one - creating a tailored interface over pre-selected portions of Linked Data. The second approach is a generalisation of the first - through the use of *data-mapping* tools that allow developers to easily set up and configure visually rich exploration interfaces over Linked Data without too much programming. For example, Exhibit [7] enables developers to create a powerful data exploration interface without any knowledge of database technology or programming, while mSpace [14] allows installation and configuration of a scalable faceted browser over a SPARQL endpoint though an installation wizard. While both of the afore mentioned approaches promote the use and repurposing of structured data, they rarely promote to users the linkages that exist to various other datasets on the Web of Data. In effect, once deployed they create their own data silo - useful for exploring the data over which the interface acts as a lens, but unable to relate to data in other, remote datasets. The third approach lets users explore and aggregate arbitrary data through the use of generic data browsers, which offer browsing the Web of Data as a analogy of browsing the Web of Documents. For example, the Tabulator [3] allows users to browse graphs of RDF[1], specify arbitrary selections of the data and analyse the selected data through inputing them in a variety of widgets such as charts, maps and timelines. Generic data browsers consume and present the data on demand and do not require any configuration. However, since RDF does not prescribe any representational information, generic data browsers often resort to generic representations - a one size fits all interface for data. Additionally the interaction and navigation in a generic browser is closely associated with the underlying RDF model, which often is too fine grained for the casual user and requires transformations or finding suitable representations before it can be used to solve a particular information need. For such reasons, generic data browsers are often too complex for casual Web users - users that are used to visually rich and custom made interfaces.

A trade off of is evident when accessing Linked Data using data-mapping approaches on the one hand and generic tools on the other. I the first case we sacrifice navigability through Linked Data as a unified, inter-connected resource and be satisfied with islands of applications over limited data sources unable to interact with each other. If we use more generic solutions, however, we risk poor usability and experience will thwart large numbers of users from experiencing and gathering information from a integrated Web of Data. In this paper we present our approach to reconcile these visions by offering a middle ground between both approaches. We propose a framework that allows data-centric interfaces to be linked based on equivalent identifiers in their respective data sources, enable them to express their state based on these identifiers, and allow these identifiers to be passed as input from one application to another as a way of enabling navigation. In effect, our framework views applications as a higher level lenses or views over graphs of data on the Web. The framework potentially unlocks novel interaction possibilities and allows for citizen end users to experience an unbounded Web of Data.

This paper is structured as follows. In the following section we briefly discuss related work and discuss challenges facing existing approaches. In Section 3 we present *mashpoint*[2], a framework for using data-centric applications as lenses over the Web; we discuss how users interact with multiple applications to explore and solve data-intensive needs. In Section 4, we discuss implementation details. Section 5, discusses the implications both from an interaction perspective and socio-technical perspective. Finally we conclude in Section 6 and discuss future work.

## 2. RELATED WORK

As discussed in the introduction of this paper, a number of tools, such as Exhibit and Dido [7, 8], mSpace [13], and /facet [5] already provide ways to quickly deploy interfaces over structured data. Most of these tools, however, offer deploying data exploration interfaces over relatively simple collections of data; for example a collection of set of resources (e.g. a collection of countries) and several facets for which can be visualised and used for filtering. Navigation and exploration of data from complex graphs, however, requires more advance interactions. Browsers such as Parallax [6], Humboldt [9], Explorator [1] and gFacet [4] introduce the notion of pivoting and set-oriented browsing. Set-oriented browsing is a natural generalisation of the Web's one-to-one browsing paradigm to a many-to-many browsing method since manipulating data often requires dealing with multiple items simultaneously and common semantically typed links offer a consistent way of refocusing with multiple items. Figure 1 displays in abstract the concept of set-oriented browsing. The majority of set-oriented browsers offer a user interface to explore and query graph data; additionally browsers such as Parallax and Tabulator allow users to select data from the explored graph and visualise the results as maps, charts, timelines and calendar views.

[1]http://www.w3.org/RDF/
[2]http://www.mashpoint.net/

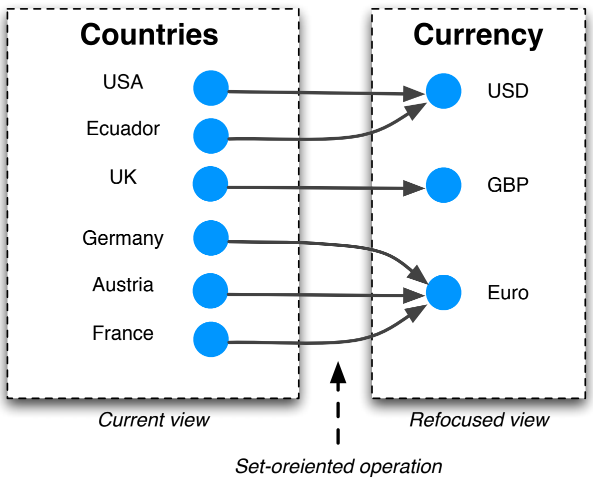


**Figure 1: Abstract representation of a set-oriented operation.**

### 2.1 Issues with Existing Approaches

As we noted in the introduction, data-mapping approaches offer useful and usable interfaces for interacting with data but limit the interaction to the data for which the interfaces are intended for; generic data browsers on the other hand, which are capable of browsing and exploring arbitrary data from graphs, have not as yet been widely adopted. In the following we offer several reasons for their low adoption.

*Dealing with Fine-grained Data.* Often times the data that the data browser exposes is much too granular for end users, often requiring them to do complex transformations or many selections before they can complete a information related task. For example, a simple question of viewing a visualisation of countries GDP/per capita on a map will require multiple steps to complete. Normally, a user would have first to explore the graph and find resources of "Countries" that will probably be associated with properties such as latitude, longitude, GDP and population. Then, a user would have to find and specify which properties will be used to query. Moreover, GDP per capita might not be available, so a user would have to combine the overall GDP and population data before the data can be used in, for example, a chart visualisation widget. Eventually, the user can reach the desired result, however the process can be long and error prone. For the majority of end users, these interactions are often too complex and time consuming. Moreover, browsers rarely capture this transformation from data to usable knowledge done by users and miss the opportunity of offering previous results as a suggested lens to new users in the browser[3].

*Information Overload.* Even with a good exploration tool that abstracts machine-readable data and allows users to perform various queries, graphs of data can still be difficult to explore. They can hold enormous amounts of data, thus frequently requiring users to find and filter to a small portion of the dataset. Additionally, when engaged in an exploratory search - search where users have no concrete information goal but rather engage in exploration - they can find it difficult to figure out which properties would make sense to combine, visualise, or which properties would make good facets for filtering. The authors of BrowseRDF [10] were the first to take note of this issue by trying out an automatic way of detecting useful facets. They acknowledge, however, that automatic approaches are limited and that additional knowledge about the ontology is required. While in the future data-centric browsers could analyse graphs of data and offer recommended views based on established ontologies we believe that such capabilities are not feasible in the foreseeable future.

*Representation.* Unlike the Web, where each page is carefully crafted for human consumption, a Web of RDF data is purposely devoid of any presentational content as a adherence to the principle of separation of content from presentation. Therefore responsibility is transferred to the browser to figure out how to represent data when the data is fetched. Generic browsers currently only base their representations and browsing models on the triple data model of RDF, and this is often reflected in browsers by employing generic representations, using simple heuristics to display data (e.g. searching for `rdfs:label` to display a resource), or providing navigation using the links only between neighbouring resources (those who share a link) in the graph. Additional representational knowledge such as lenses [11] can improve data representation, however crafting lenses without any knowledge of the context in which they will be used in the generic browser is a challenge. Moreover, it is unclear who should bear the effort of providing lenses for generic browsers - the publisher of the data or the browser consuming the data, and what is the immediate benefit of providing representations of the data as lenses as opposed to just building a custom made web site to display a publishers data.

[3]Parallax [6] allows users to export live views of the data and embed them in blogs and web pages.

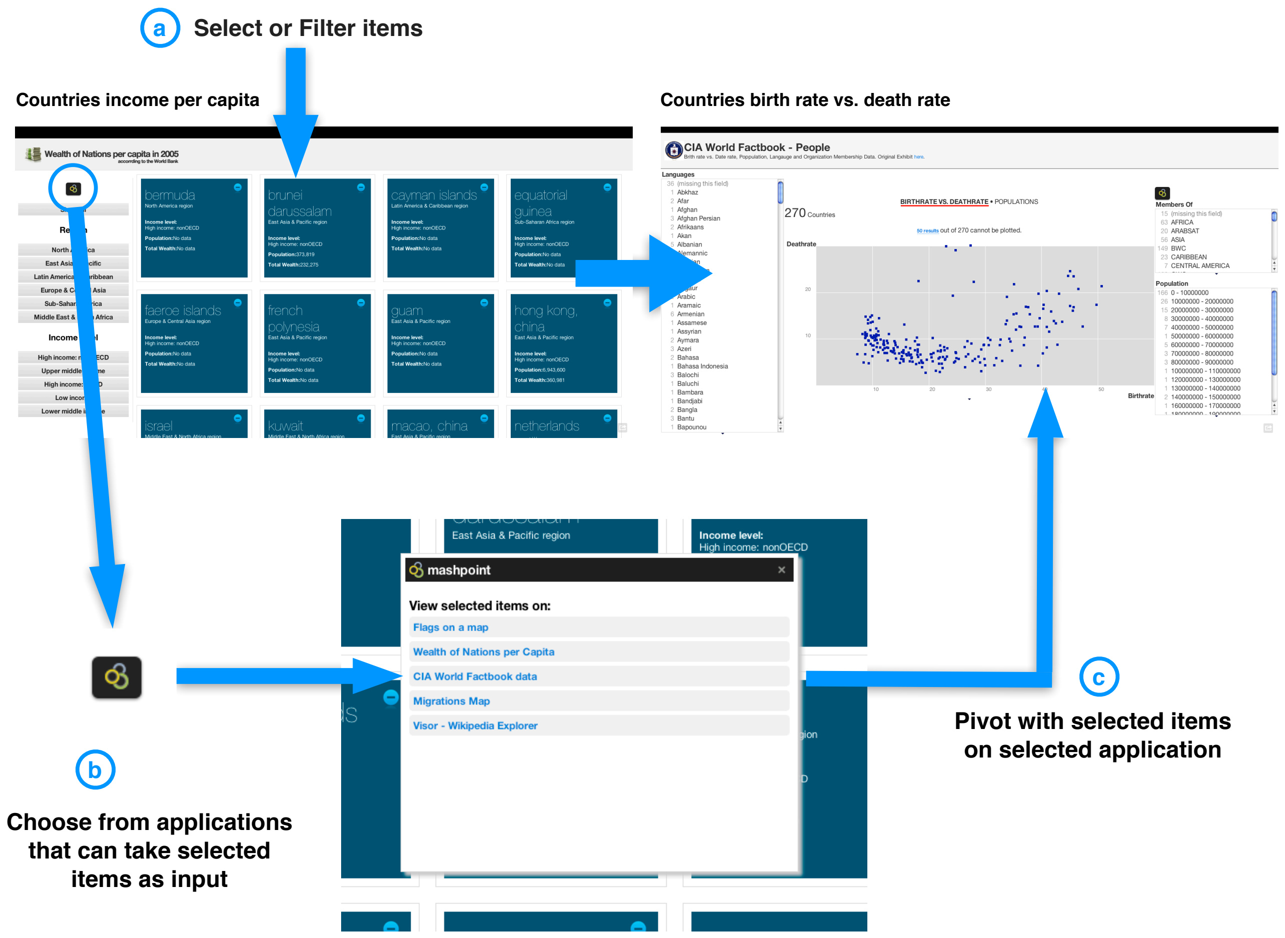
## 3. MASHPOINT: USING INTER-CONNECTED DATA-CENTRIC APPLICATIONS AS LENSES

In this Section we describe *mashpoint*[4] our prototype framework for using and navigating through higher level abstractions or lenses over data. The basic premise of the mashpoint framework is that when data is viewed or interacted with by end users, it should always be represented within a certain context. Data-centric applications are perfect examples of data viewed in context and therefore considered as lens over some data in the mashpoint framework. A data-centric application is any application that is powered by and offers some interaction over some data. For example an Exhibit is a typical data-centric application. With mashpoint our goal is to enable users to select specific data through the interactions offered in one application and pivot (i.e. execute a set-oriented operation with that selection) to another application that can accept that data as input and provide new information corresponding to the selection done in the previous application. In such a way we achieve set-oriented navigation through graph data. Figure 2 depicts this interaction technique between two mashpoint enabled applications. In the example, the first application[5] (Figure 2a) shows a simple data-centric application which allows users to explore data about countries GDP/per capita and population information by allowing the data to be filtered by "Income level" or "Region". For example, selecting "Low income" from in the first application will filter and show low income countries to users. A user can then click on the mashpoint button (Figure 2b), which pops up a window and offers other applications that can take and offer new insights regarding the selected data in the first application. For example, a user may want to view CIA factbook data about the "Low income" countries and choses to open that application[6] (Figure 2c). The CIA factbook application has various data about countries. For example, the user can view information about countries birth-rate vs. death-rate and filter the existing selection on different facets. Note that the items that are shown in the new application reflect the items chosen in the previous application.

As part of the development of this framework, we started adapting and linking existing data-centric applications on the Web. Figure 3, shows three other applications that we adapted and linked up using our framework. The first one[7] (Figure 3a) is a simple Exhibit showing images of world currencies, and the currency code. The second application[8] (Figure 3b) is a simple exploration application which allows users to view and browse countries flags depicted on a map. The third application[9] (Figure 3c) is an existing application we found on the Web[10] that we integrated into our framework.

In the following we describe several examples how combining and navigating with different selections in the data can

[4]http://www.mashpoint.net
[5]http://mashpoint.net/demoapps/countriesincome/index.html
[6]http://mashpoint.net/demoapps/birthratevsdeathrate/index.html
[7]http://mashpoint.net/demoapps/currencycodes/index.html
[8]http://mashpoint.net/demoapps/flagsonamap/index.html
[9]http://mashpoint.net/demoapps/mapmigrations/index.html
[10]http://migrationsmap.net/



a Select or Filter items
Countries income per capita
Countries birth rate vs. death rate
b Choose from applications that can take selected items as input
c Pivot with selected items on selected application
mashpoint
View selected items on:
Flags on a map
Wealth of Nations per Capita
CIA World Factbook data
Migrations Map
Visor - Wikipedia Explorer

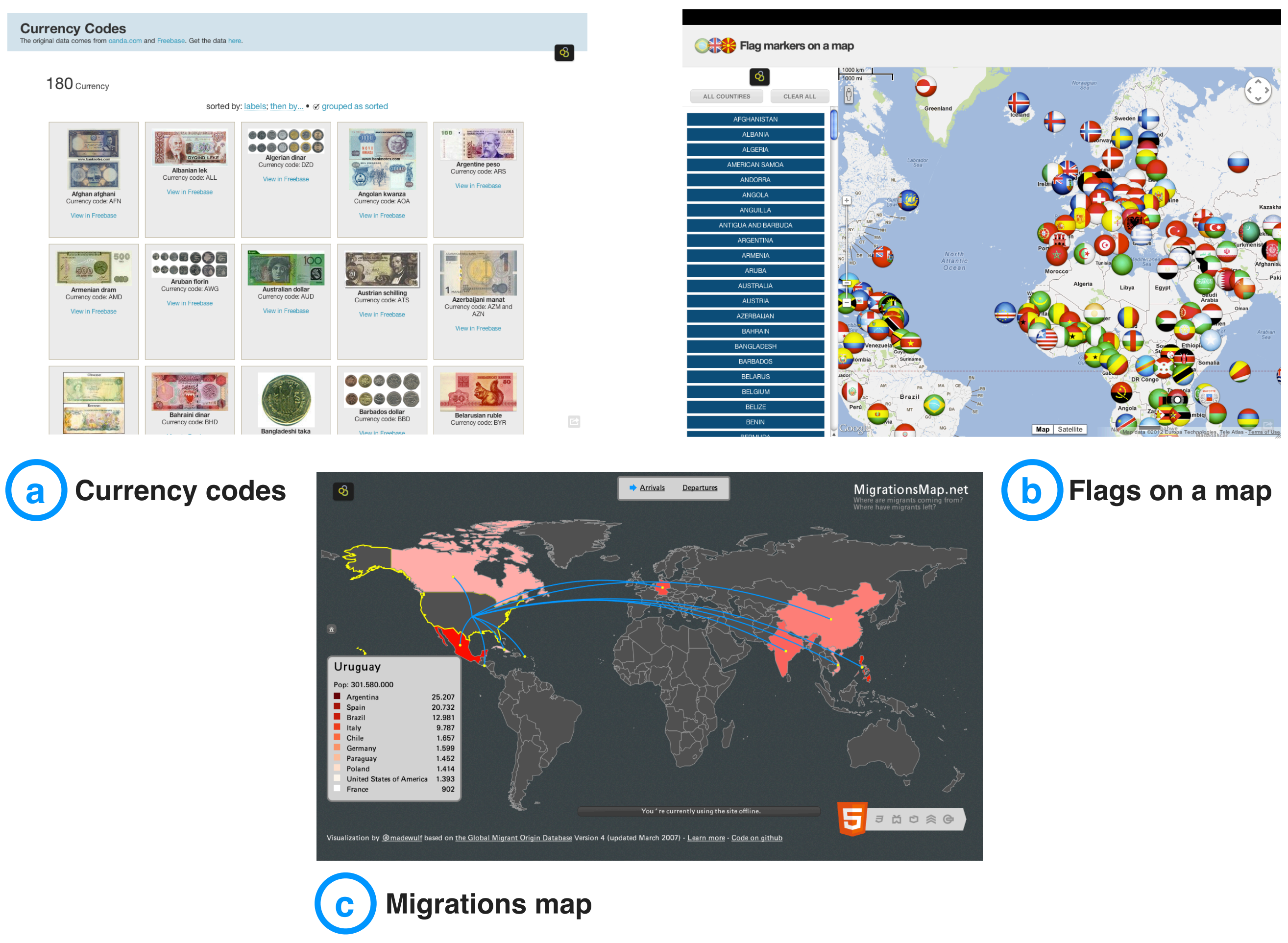
Figure 2: A pivoting operations between two applications in mashpoint.

produce some interesting insights into the data:

- In the previous example we used an application that showed World Bank data about GDP/per capita (Figure 2a). A user can browse the data in that application using the facets that are provided, however, the application provides only a single representation of the data. A user may wish to view countries on a map in order to see how countries of different income groups are distributed geographically (for example, which continents contain "Low income" countries?). Using current tools on the Web, a user would be required to copy and paste each country in another application (e.g. Google Maps) to answer this question. Using mashpoint, however, the user can take any selection of the data and find applications that are able to provide geographic information and representations about the data. For example, after filtering to "Low income" countries the user can open the mashpoint dialog and select the Flags on a map application (Figure 3a), which can display the current selection of countries on a map as little flag markers on a map. Immediately it is revealed that out of the all the low income countries only a single one (Haiti) is in the Americas, while the rest of the low income countries are in sub-saharan Africa, Central and South-east Asia.

- Continuing from the previous example, once viewed geographically, a user can chose to view additional data about the selected "Low income" countries by pivoting to the CIA Factbook application (Figure 3b) and explore data about birth rates and death rates for the selected, low income countries. The user decides to compare these with high income countries so he/she repeats the same navigation, only this time starting with a high income countries in the first application. The user can then conclude that there is great diversity in both birth rates and death rates in low income countries as opposed to high income countries where death rates are fairly consistent, and birth-rates experience small variations.

- Similarly to the previous example, a user might chose to view migration patterns and instead of grouping countries by income levels he/she might be interested in a particular geographic region. For example, pivoting from Middle-eastern countries in (Figure 2a) to the Map Migrations app (Figure 3c) can reveal to the user that people from those countries typically migrate to countries in the same region and to countries of the Western Europe and Northern America.



Uruguay

Pop: 301.580.000

| | |
|---|---|
| Argentina | 25.207 |
| Spain | 20.732 |
| Brazil | 12.981 |
| Italy | 9.787 |
| Chile | 1.657 |
| Germany | 1.599 |
| Paraguay | 1.452 |
| Poland | 1.414 |
| United States of America | 1.393 |
| France | 902 |

**Figure 3: Example applications linked with the mashpoint framework.**

- A user is planing a trip across Europe, traveling to multiple European countries. The user is aware that some European countries share a single currency however needs information about each of the countries he/she is traveling to. Deciding that the best way to quickly select the countries of interest is to use a map, the user selects the countries of interest on the Flags on a map application (Figure 3a). By selecting the Currency codes application (Figure 3b) the user is able to pivot with the current selection of countries, obtaining the corresponding currencies of each country. This saves the user time since the alternative would be to look up each country and integrate the information manually.

If we examine carefully the above examples, an interesting observation can be made from each example. Each application by itself offers very limited capabilities to interact over that data. By enabling selections of data to be pivoted or shifted to other applications, we not only aid in the discovery of new information, which is one of the advocated uses of Linked Data, but additionally allow users to interact with the newly found information in a way that is tailored for the specific data to be displayed.

## 4. IMPLEMENTATION

This Section describes the implementation details of mashpoint. Our investigation into designing mashpoint began with the simple interaction challenge by asking "Why are generic data browsers unusable?" and "How do we solve these problems, and where can we gain in usability without sacrificing browsing and navigating capability?". While the implementation of mashpoint were guided by these principles, we tried to implement mashpoint so that the barrier for entry for linking application to the framework would be minimal.

The implementation consists of three parts: (1) the applications themselves, which need to be data-centric in nature and be built with certain requirements, (2) a discovery service that allows applications to look up other applications so that the user can pivot between them and (3) a means of communication between the applications and discovery service. In the following we discuss each part in detail.

### 4.1 Applications

In order to enable pivoting between applications, they need to be designed according to some specifications and rules. Our choice of specifications was motivated by a desire to make the integration of new and existing data-centric applications as painless as possible i.e. not to impose any unnecessary learning curves or restrict developers to use any particular technology. Thus to link an application to mashpoint, it needs to have the following properties:

- **Offer Data-centric features.** Each application in mashpoint needs allow interaction over data with identifiable resources. An application can hold multiple collections of identifiable resources - for example be about People, Countries, Events etc. An Exhibit like the one in Figure 2c is a typical example of a data powered application that offers browsing over data about countries. In the data of this particular Exhibit, each country is an identifiable real-world object.

- **Use of URIs.** While the data underlying the application does not necessarily need to be in RDF, an URI needs to be present for each identifiable resource of the data. In our examples, since the data is about countries, each Country needs to be be associated with an URI. The use of URIs is also needed in order to be able to save the state of the application. We discuss how to preserve the applications state in the next two points.

- **Be able to select multiple resources.** An application in this framework should typically enable selection of resources in order to be able to pivot with arbitrary selections of data. Selections of the data can can be provided in multiple ways. For example, items can be selected thought filtering by providing various facets over the data and/or allow arbitrary items to be selected. This selection of items will then be passed on as input to another application. Whenever an application changes its focus, the state of the application should be made explicit in the URL of the application. In mashpoint we require each application to list the current resources in view through a `mashpoint` parameter in the URL. Figure 4 depicts the saving of state in each application. Figure 4a for example depicts an application showing a single resource and a `mashpoint` that denotes this state. Similarly, Figure 4b shows the interface on a state with two resources. The state can also group a list of resources (Figure 4c) in order to reflect certain collections of resources e.g. a interface that displays data about both "Countries" and "Currencies".

- **Be able to represent multiple resources on input.** Application should be able to take any arbitrary selection of URL identifiers that represent the resources of the application and be able to show some representation of that data that reflects the selected items i.e. be able to arbitrarily retrieve any state of the application.

The choice of URIs is also an important factor in the current implementation of the framework. In order to enable pivoting between applications we need identical identifiers across all mashpoint enabled applications. In our current

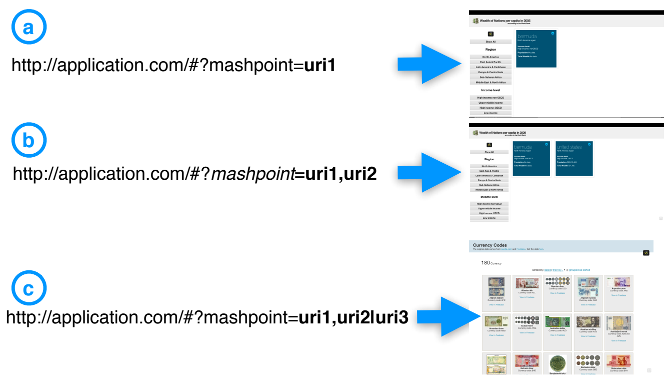

a
http://application.com/#?mashpoint=uri1
b
http://application.com/#?mashpoint=uri1,uri2
c
http://application.com/#?mashpoint=uri1,uri2|uri3

**Figure 4: Preserving the state in a mashpoint-linked application.**

instantiation of mashpoint we rely on Freebase[11] as a service to which data used in applications need to be reconciled. Note that we have chosen Freebase for convenience reasons - Freebase and the support offered in Google Refine offers tools to quickly reconcile[12] arbitrary data with Freebase concepts. While the data in the applications need to be reconciled against Freebase, it does not preclude using other data sources that already use established URIs. For example applications consuming Open Linked Data can use resources such as `sameas.org`[13] to either reconcile their data to Freebase or even use the service in real time (although the former is probably the preferred solution because of optimisation issues). In essence, it does not particularly matter which URIs we offer as reconciliation, since the framework requires just reconciliation of identifiers. Moreover, the architecture could also be redesigned in a different way - it could allow applications to use whatever URIs they see fit and try to reconcile them and do discovery in real time through the use services such as sameAs.org. To illustrate this we have already connected Visor[14] [12] a end-user tool for exploring DBPedia [2] data to the mashpoint framework. At this point of time, however, a priori reconciliation provides a more optimised solution to the co-reference problem in our case.

## 4.2 Discovery Service

In order to be able to find applications which can be used to pivot from the current application we implemented a discovery service for mashpoint-enabled applications. The discovery service is a repository that simply keeps a record about which URI identifiers can be represented in which applications. Applications therefore need to register themselves in the discovery service and "subscribe" their URI identifiers. Registering with a set of URIs means that an application can represent and show data about any subset of the identifiers it is subscribed to. Once registered, each application can communicate with the discovery service to find other applications that can take the current selection (represented through the URIs in its state) as input. Figure 5 depicts

[11]http://www.freebase.com
[12]http://code.google.com/p/google-refine/wiki/ReconciliationServiceApi
[13]http://sameas.org/
[14]http://visor.psi.enakting.org/

this architecture. For clarity, URI identifiers are represented with dots, squares and triangles to denote different groups of URIs found across different applications. For example Application 1 is registered with the dot identifiers which means it can take any subset of these identifiers as input. Application 2 can either take the any subset of dot identifiers but it can also take any subset of square identifiers as input. Similarly Application 3 can take subsets of square and triangle identifiers. These groups of URI identifiers are assigned by the application registering to the discovery service.

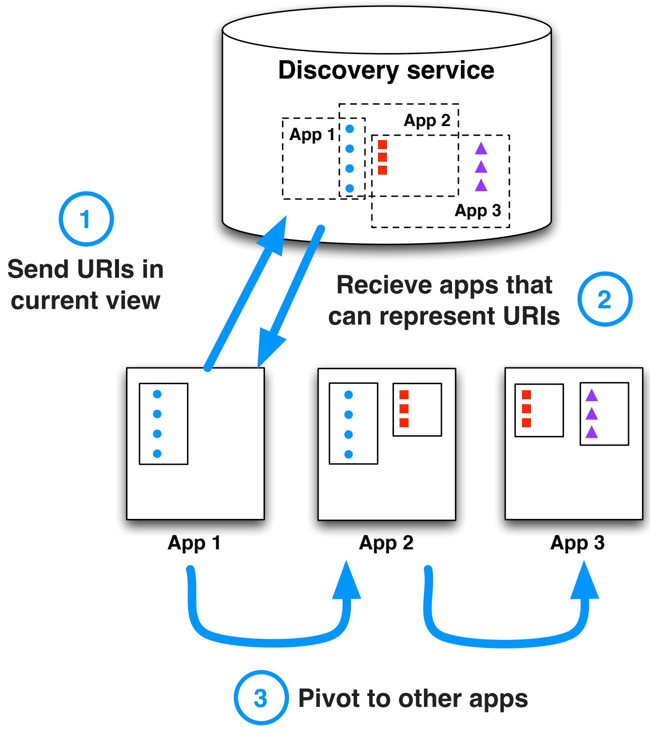


**Figure 5: Architecture of the mashpoint framework.**

## 4.3 Pivoting Across Applications

In order to enable pivoting across applications, they need to communicate and request information based on the current state of the application. Each application therefore communicates its state to the discovery service i.e. it sends the URIs that currently represent the data which is viewed in the application, and retrieves back a list of applications that are able to receive those URIs as input.

In order to facilitate this communication, each application in mashpoint incorporates a small JavaScript widget that is able to parse the URL for the URI identifiers and send them to the mashpoint discovery service (Figure 5-1) The discovery service then retrieves which applications can take the URIs as input and sends them as a response with their states reflecting the identifiers in the request (Figure 5-2). The widget in each application is a third party code that adds a mashpoint button, facilities the communication with the discovery service and pops up the dialog that suggests appropriate applications to users. We note that the discovery mechanism and widget may be omitted from an application. For example, cases may exist where a publisher of an application may want to offer users pivoting to only a certain, predefined set of applications. Therefore the publisher of the application can discover those applications once, and include them as regular links in the application. This removes the need for a third party discovery service, however it is now up to the publisher to keep the links to the other applications consistent with the current state of the application.

## 5. DISCUSSION

In this Section we discuss open issues, challenges and implications in adopting mashpoint as a framework.

### 5.1 Interaction Challenges

A number of interaction challenges need to be investigated and addressed in order to mitigate any usability issues. First, unlike a generic browser where data is viewed, browsed and manipulated within the context of a single application, mashpoint proposes an approach to data browsing where views of the data are provided by distributed applications that can be contributed by many publishers. Evaluating how well users can combine and pivot data between different applications in order to solve data-centric tasks remains to be explored and evaluated.

Another issue is the implementation of the set-oriented operations in mashpoint. In the current implementation, when the discovery service is queried with a set of URIs, every application that can represent some subset of these URIs is retrieved. For example, if an applications state is currently focused on three countries e.g. France, Germany, and Brazil, and another application can show data only about European countries, that application will also be retrieved when applications will be requested for those three countries, even though it will only be able to show information on two out of the three countries. This means that in some cases not all URIs will be able to be represented in the application to which the user will pivot. Such information needs to be surfaced to the user, or ideally, enable users to filter and browse particular types of applications according to the content or size of the subset they can take as input.

Another problem that might hinder usability is the lack of context between pivoting steps in applications. Sometimes the data in two linked applications will follow a one-to-one mapping e.g. two applications both showing data about countries. This corresponds as navigation through resources that are linked through a `owl:sameas` relation. As we've seen in our examples, however, pivoting can take place between applications that contain data on diverse topics, for example pivoting from an application about "Countries" to an application about "Currencies". This corresponds to navigating through resources with arbitrary links between them and often times the relationships between them will be many-to-many. In our example, the application showing currency data might not explicitly state which countries are using that currency. Thus a user would find it difficult figuring out which country shown in the first application corresponds to which currency in the second one. Several solutions to this problem are possible. First, either publishing practices will compound users to normally include labels or representations of the items of input, or second, the state of an application encoded in the URL can include additional contextual information that will be sent alongside the URIs. At present, however, this remains future work.

## 5.2 Social Contribution Factor

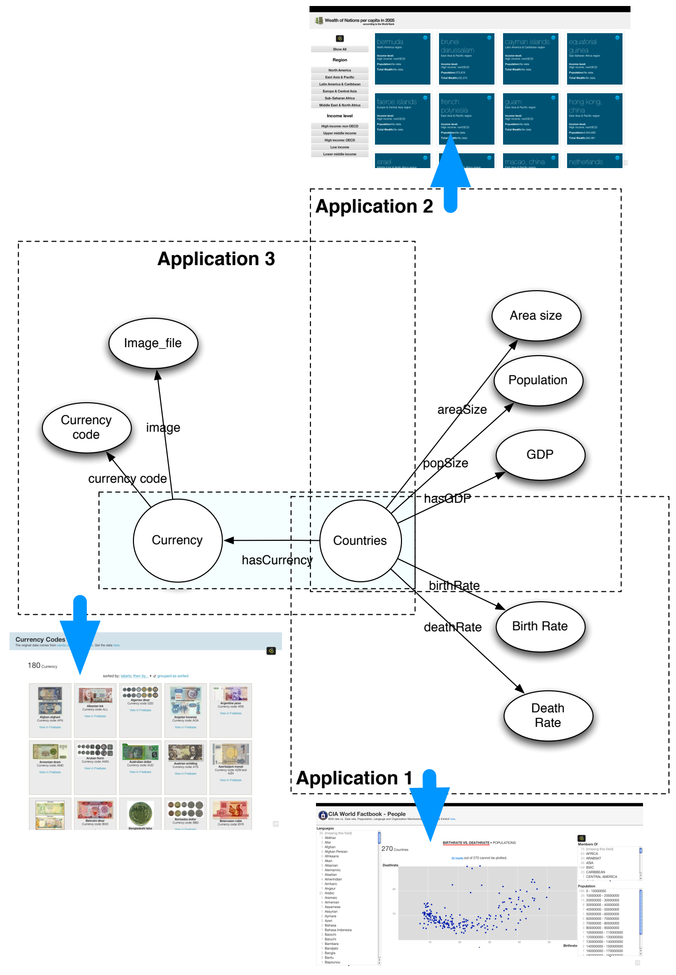


**Figure 6: Model of data and it's relationships encapsualted .**

By design, mashpoint sets forward a paradigm that has inherently a social factor of contribution, which is similar to the social nature of publishing and linking in the original Web. The reason why applying data mapping tools such as Exhibit [7] have seen much wider acceptance than generic browsers such as Tabulator [3] is because a publisher of an Exhibit [7] can control the look and feel of the data and immediately see value of providing a rich data-centric interface over data. The original Web followed a similar pattern; a published Web site offered a custom made document and a presence on the Web - linking to other web sites only improved the quality of the web site by providing connivence of finding relating information. For example a web page about events in Southampton is by itself a useful contribution to the Web, and the publisher can increase the value of information by providing links to other pages (e.g. the Wikipedia page for Southampton or other related web pages offering events information about Southampton). However it is important to note that even without the links, the web site is useful by itself. As a publishing recommendation mashpoint acts in a very similar way. Applications can be viewed as contributions which are useful by themselves - they allow some value over the data they were initially designed for. By linking them up, and enabling pivoting to other data-centric applications, the original application can only increase the value of the original application. In fact, this attribute may provide incentive for publishers of data-centric applications on the Web to link their data using frameworks such as mashpoint.

## 5.3 Low Barrier for Entry

Although none of the applications we have presented in this paper directly operates over live Linked Data, or directly use RDF data directly (although we extracted some of the data from Linked Data sources), we believe this fact to be an added strength to the framework, since it only lowers the barrier for linking new applications - it does not mandate or impose any particular data model on the user. This is not to say that the applications using this framework cannot use standard data-models such as RDF. In fact, as the title our our paper suggests, we view the applications as high-level lenses over graphs of data, as depicted in Figure 6. In Figure 6 we can see the connections between the data items used by the mashpoint-enabled applications earlier. The applications encapsulate views over the data, and the relationships between the data are just hidden within the individual applications. Thus applications are can chose to use either RDF or any other data model and pivoting takes place where these lenses overlap.

## 5.4 Incentives for Publishing and Linking

During the last 2 years, the Linked Data community has been advocating data publishing using Linked Data standards, and has promoted the use of these standards as a quality indicator for data available on the Web[15]. However, the benefits of publishing Linked Data and linking to other remote data sources remain elusive for most data publishers and consumers outside the community. Often the results are data repositories that are rarely used and provide sparse linkages to other remote datasets. This lack of immediate value for the effort of converting ones data as Linked Data can thwart many potential adopters of these technologies. With mashpoint we hope to target this problem, particularly in providing incentives for linking data. As we already mentioned, publishers of data-centric applications would only increase the value of their applications by allowing users to find useful, related data without changing the original application. By requiring publishers to reconcile their data we already promote the use of URIs in their datasets, while showing the immediate benefit of being able to pivot and suggest related data to the users of the application.

## 6. CONCLUSION

In this paper we presented mashpoint, a framework which aims to promote the value of a Web of Linked Data by enabling interactions that take advantage of links between remote datasets while remaining usable and familiar as browsing the Web itself. As a publishing framework, we view mashpoint as an extension to the current landscape of tools

[15]http://inkdroid.org/journal/2010/06/04/the-5-stars-of-open-linked-data/

providing end user access access to the Web of Data (Figure 7). While we've communicated the overall idea of having data-centric applications serve as lenses over Linked Data, the work presented here is still work in progress and many open challenges remain. Our future work include evaluating the initial prototype with users allowing them to complete data-intensive task by navigating thorough a Web of tens of mashpoint-linked applications. We believe that the study will provide many insights into the overall design and implementation of the framework. If mashpoint achieves wide adoption, our long term plan will be to study the ecology of inter-linked applications it will create.

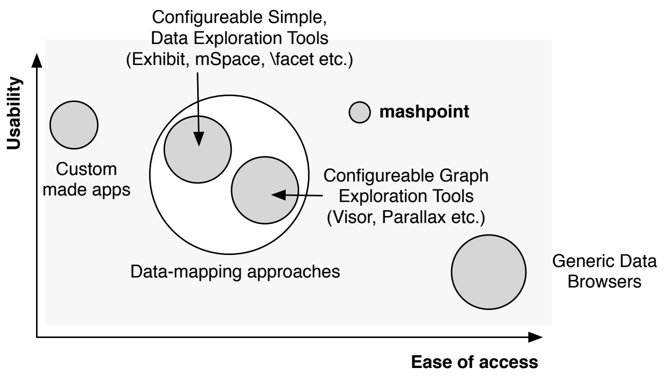


**Figure 7: Landscape of data-centric tools and applications. The Figure shows the tools usability agains the ease of access to the Web of Data. Ease of access relates to how scalable is a tool with respect to accessing diverse datasets from the Web of Data.**

## 7. ACKNOWLEDGMENTS

This work was supported by the EnAKTing project, funded by EPSRC project number EI/G008493/1. Many thanks to Manuel Salvadores for providing useful feedback for this work.

## 8. REFERENCES

[1] S. Araujo, D. Schwabe, and S. Barbosa. Experimenting with explorator: a direct manipulation generic rdf browser and querying tool. In *Workshop on Visual Interfaces to the Social and the Semantic Web (VISSW2009)*, February 2009.

[2] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. Dbpedia: a nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[3] T. Berners-lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *In Procedings of the 3rd International Semantic Web User Interaction Workshop (SWUI06*, page 06, 2006.

[4] P. Heim, T. Ertl, and J. Ziegler. Facet graphs: Complex semantic querying made easy. In *Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010)*, volume 6088 of *LNCS*, pages 288–302, Berlin/Heidelberg, 2010. Springer.

[5] M. Hildebrand, J. van Ossenbruggen, and L. Hardman. /facet: A browser for heterogeneous semantic web repositories. In *International Semantic Web Conference*, pages 272–285, 2006.

[6] D. Huynh and D. Karger. Parallax and Companion: Set-based Browsing for the Data Web. 2009.

[7] D. F. Huynh, D. R. Karger, and R. C. Miller. Exhibit: lightweight structured data publishing. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 737–746, New York, NY, USA, 2007. ACM.

[8] D. R. Karger, S. Ostler, and R. Lee. The web page as a wysiwyg end-user customizable database-backed information management application. In *UIST '09*, pages 257–260, New York, NY, USA, 2009. ACM.

[9] G. Kobilarov and I. Dickinson. Humboldt: Exploring linked data. In *Linked Data on the Web (LDOW2008)*, 2008.

[10] E. Oren, R. Delbru, and S. Decker. Extending faceted navigation for rdf data. In *ISWC*, pages 559–572, 2006.

[11] E. Pietriga, C. Bizer, D. Karger, and R. Lee. Fresnel - a browser-independent presentation vocabulary for rdf. In *In: Proceedings of the Second International Workshop on Interaction Design and the Semantic Web*, pages 158–171. Springer, 2006.

[12] I. O. Popov, M. C. Schraefel, W. Hall, and N. Shadbolt. Connecting the dots: a multi-pivot approach to data exploration. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 553–568, Berlin, Heidelberg, 2011. Springer-Verlag.

[13] m. c. schraefel, D. A. Smith, A. Owens, A. Russell, C. Harris, and M. Wilson. The evolving mspace platform: leveraging the semantic web on the trail of the memex. In *HYPERTEXT '05: Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*, pages 174–183, New York, NY, USA, 2005. ACM.

[14] D. A. Smith, I. Popov, and mc schraefel. Data picking linked data: Enabling users to create faceted browsers. In *Web Science Conference 2010*, March 2010.