A VIRTUAL REPOSITORY FOR LINKED-DATA-BASED DISASTER MANAGEMENT APPLICATIONS

F.-P. YANG, Y.Z. OU, C.W. YU, J. SU, S.-W. BAI, J.-M. HO & J.W.S. LIU
Institute of Information Science, Academia Sinica, Taiwan.

ABSTRACT

Typical state-of-the-art disaster management information systems (DMIS) cannot support responsive discovery and access of data and information needed to handle unforeseen emergencies. Adding semantics and relations to legacy data and transforming them to linked data (LD) can remove this limitation. The virtual repository presented in this article is a development environment for this purpose: It provides application developers with tools for incremental transformation of legacy data and information in the DMIS into LD as needed by the applications. The virtual repository also provides the applications with support for runtime access of LD created and maintained using its tools.
Keywords: Disaster management applications, format translation, linked data.

1 INTRODUCTION

Experiences from past disasters have shown time and again that people's abilities to cope with natural disasters depend critically on the availability of data and information needed to support preparedness and response decisions and operations. Such information not only can help save lives, prevent injuries and reduce damages, but also can make emergency response and rescue operations safer and more efficient. Statistics also show, however, that the positive impact of information deteriorates rapidly with time following a disaster [1]. For making decision support information easily discoverable and accessible by Emergency Operation Center (EOC), responders, victims and general public should be a primary design objective of all disaster management information systems (DMIS).

Today, this objective is met only partially by DMIS of numerous countries and regions in the world, including Taiwan and most of Asia. Typical state-of-the-art DMIS rely mainly on data and information in sources owned by government agencies responsible for disaster management. As a part of standard operating procedures (SOP) in preparation for a disaster, likely emergency scenarios are developed based on knowledge on similar past disasters and experiences in dealing with them. When the disaster becomes imminent, the EOC identifies the data and information needed to deal with the scenarios and has the data retrieved from available sources and cached on devices, computers and display systems and thus makes the data ready for use by decision makers and responders during the emergency. This practice and the DMIS used to support the SOP work sufficiently well for disasters (e.g. typhoons, seasonal downpours and earthquakes of usual severities) that occur frequently in the region.

They fall short, however, in case of unforeseen calamities, especially when the levels of devastations exceed the prediction. Examples include Morakot Typhoon in Taiwan, 2009 Haiti and 2011 Japan earthquakes and hurricanes Katrina and Sandy in USA. Data and information on things such as floor plans/structures of collapsed buildings, available pumps and transports, locations of people needing help, etc. in hands of emergency managers, responders and victims can save lives and properties. A typical DMIS offers little or no support to enable timely discovery, access and use of such data when the data are in sources outside of the official DMIS and across institutional boundaries.

ISSN: 2041-9031 (paper format), ISSN: 2041-904X (online), http://www.witpress.com/journals
DOI: 10.2495/SAFE-V5-N1-1-12

The virtual repository (VR) described in Ou *et al*. [2] is a part of a middleware-level framework for building open information systems that are free of these limitations. Generally speaking, a VR provides client applications (and services) served by it with interface services and interoperability tools to hide the fact that data and information brought to the applications may reside physically in multiple and diverse (information) sources, some of which may be owned by non-government entities (e.g. businesses, institutions, communities and NGOs). An obstacle to responsive flow of information during emergencies arises from policies and mechanisms erected by information sources for the sake of security and privacy protection. The VR provides an accountability-based information brokerage service [3] designed to remove this obstacle for client applications during emergencies on the one hand, while providing its client information sources with confidentiality and privacy protection on the other hand. The next section will present an overview of these VR services.

A way to enable discovery of data and information is to exploit linked data (LD) and related technologies [4–6]. Semantics of data and relations between them provided by LD can ease the discovery of critically needed data during emergencies and enable the design and implementation of new and more effective disaster preparedness and response applications. Research projects on building emergency information systems and management infrastructures on LD and Linked Open Data (LOD) include the ones described in reference [7–9]. Tools provided by projects such as LOD2 [10] and SIMILE [11] can help to reduce the effort and speed up the development of LD-enabled DMIS and disaster management applications.

Nevertheless, the adoption of LD within information systems for disaster management has been extremely slow and in some parts of the world, it may not happen. A reason is that the effort required to turn existing DMIS into LD is enormous, and the data providers who do this enhancement may not be benefited directly from the effort. VR aims to speed up the adoption of LD by providing developers of LD-based applications with tools and services for incremental transformation of legacy data in DMIS and other sources into LD on demand as needed by the applications. In this way, VR serves as a development environment of LD, enabling LD consumers to be LD producers. In addition, VR provides its clients with storage for LD created and maintained using its tools and supports runtime access of the data. For this reason, VR can be viewed by client applications with similar data requirements as an extensible repository of links and cached LD.

This article focuses on a system of tools and services collectively called *X2R converter*, or simply X2R. The system is a multi-format converter for transforming legacy data into LD in a Resource Description Format (RDF). X2R builds on existing translation and mapping tools [12–15]. For the sake of concreteness without loss of generality, subsequent discussion assumes that when presented by a developer with a file of application data in a supported format, one of the RDFizers maintained by MIT SIMILE group and W3C [11, 15] is used by X2R to convert the data into RDF. The RDF file produced automatically often contains temporary Universal Resource Identifiers (URIs) generated by the translator. The file may also have blank nodes. Clearly, one cannot search for these types of nodes. Their presence in a file hinders the discovery of the content of the file. For this reason, they are referred to as *bad nodes*. To improve the quality of the RDF file produced by the transformation process, X2R performs a refactoring process: In this process, all the bad nodes in the RDF file produced by the translator are extracted. For each extracted node, X2R searches a specified list of external ontologies to find a URI that meets the developer approval as replacement; failing to find a suitable replacement helps the developer to construct a suitable one, and then replaces each extracted node with the developer-approved URI. X2R also provides a service to manage URIs created by application developers, support their reuse, and thus, build a VR internal ontology incrementally.

Following this introduction, Section 2 presents a brief overview of the architecture and key components of VR to make this article more self-contained. Section 3 presents a pilot study in order to identify complications that a developer may encounter in the process of converting application data in legacy formats to LD in an RDF format. The lessons from this study helped to guide the design of the X2R described in Section 4. Section 5 summaries the article.

2 STRUCTURE AND COMPONENTS OF VR

Figure 1 presents a runtime system view of a VR to illustrate the relationship among the middleware, its client applications (i.e. the applications served by the VR) and client (data) sources (i.e. data sources used by one or more client applications). Typically, the VR is distributed; it runs on diverse computers and smart devices, which by prearrangement are made available pervasively (e.g. at convenience stores, schools, hospitals, etc.) to provide storage, computing and communication resources to the middleware and some of the client sources and applications. These computers and devices are called *points of service* (POS).

2.1 Sample applications

As examples, Fig. 1 shows three client applications. A C3 (Command, Control and Communication) system deployed at an EOC is included here to highlight the possible diversity of applications served by a VR. Application systems exemplified by C3 typically access some of the client sources directly. For the sake of simplicity, the figure omits those access paths. The figure also omits details about VR interface services needed to support models and views of the client applications and hide the fact that data and information used by the applications may reside in independent sources.

Mobile Assistant for Disasters (MAD) [2] is the basis of the case study described in the next section. It is a service designed to make emergency preparedness information and data (EPID), including data on locations and available resources of emergency response facilities (i.e. shelters, hospitals, parks with portable water, etc.) readily available to the general public

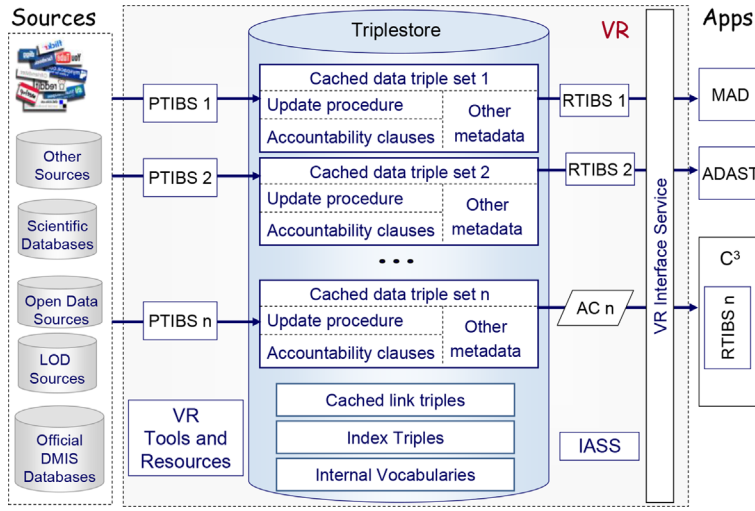

Figure 1: Runtime view of a VR and its clients.

at all times, especially during emergencies. The application has a client-server structure: MAD clients run on mobile devices of the end users and bring to the users EPID in easy-to-use forms. There is at least one Interface Server (IS) and a POS server on each POS used by MAD. The former is the MAD component responsible for retrieving data needed by MAD to service its users from sources containing open data provided by local governments. Today, open data sources are available in numerous cities and counties worldwide, including Taipei; London, England and Yokohama, Japan. The latter are servers that run on POSs. POS servers are shared by the VR and client applications of the VR and provide to them some of the basic VR services, including Internet access and local download support.

Specifically, for each region serviced by MAD, the IS server retrieves from local government source(s) data on emergency response facilities, converts the data into LD in RDF format, partitions the data into subsets and publishes to each POS server the subset(s) containing data on facilities in the neighborhood around the POS. Figure 1 refers to the MAD data stored on POSs as 'Cached data triple set 1.' By thus distributing the data on POSs and making the data downloadable via local connections to diverse mobile devices, MAD makes EPID critically needed during emergency highly available even when Internet and phone connections are disrupted during and after a major disaster. Because data exchanged within MAD components are in RDF format regardless of the data models and formats of data retrieved by the Interface Server, MAD mobile clients work in all geographical regions with the service.

While the RDF model and format(s) are a design choice of internal data for applications such as MAD, they are essential for applications that must be able to discover and access as soon as possible data from sources not contained in the official DMIS. Automatic Disaster Alert System for Tourists (ADAST) [2] is such an application: In response to an alert declared by a responsible authority (e.g. the Central Weather Bureau) warning of an imminent calamitous event (e.g. a severe storm and possible landslides), ADAST proactively notifies people in the affected areas specified by the alert. When the threatened areas include popular national parks, the application must reach not only local residents, but also tourists. The data required for this purpose are likely to be in sources maintained by multiple government agencies and companies (e.g. real-time data on numbers and locations of tourist groups are likely to be in databases maintained by Tourism Bureaus and tour companies). The VR serving this and similar applications should provide at least semantics on and links to data critical for the applications to meet their minimal requirements. In the case of ADAST, a minimal requirement is that the contact persons of all tour companies operating in the park(s) and park ranger stations are notified. Hence, semantics and links to data on these entities, referred to collectively as 'Cached link triples' in Fig. 1, are created during the design and development process of the application and maintained in the triple store of the VR. The values of the data related by these links are released to selected POS servers following policies governing when and to whom such data are to be released during emergencies.

2.2 VR services

Figure 1 shows two VR services: Intelligent Active Storage System (IASS) [16] and Trustworthy emergency Information Brokerage Service (TIBS) [3]. IASS aims to provide a DMIS built from passive sources served by a VR with the capability of automatic event-triggered, push-based delivery of critical data to specified applications and through them to the end users. It does so by allowing client applications to define events or conditions that will trigger push-data operations in terms of values of specified data objects in client sources

and the VR triple store and specify the data sets and recipient applications of each push-data operation.

As an example to illustrate how IASS may be used, we consider a landslide warning application. The application may request IASS to monitor sensor readings from a soil moisture sensor data source and the rainfall forecast by the Central Weather Bureau. It can then specify that its alert message be sent to designated disaster preparedness applications when monitored sensor readings indicate increasing moisture level beyond a specified threshold and the weather bureau forecasts more rain during the day. The version of the VR described in [3] has a more commonly seen variant of monitoring and notification service, called client subscribe and notification service. To use the service, an application first registers with the services and specifies the types of disaster alerts (e.g. typhoon, earthquake, debris flows and downpour) about which it wants to be notified and the maximum allowable delay in notification for each type of alert. These services monitor the authorized sources where the specified types of alerts and warnings are posted (or published) and notify the requesting application when a specified alert is found (or received).

TIBS [3] is a VR service that supports emergency information release and access policies specified by the data sources. The policies are defined by release clauses and accountability clauses. The release clause of a set of data defines the conditions under which the data may be released. (e.g. EPID are released and kept up to date by local open data sources at all time and can be retrieved anytime by applications such as MAD. In contrast, the identities and possible locations of tourists in an area are released only when the area is threatened by floods and landslides.) Release policies are enforced by the prospective part of TIBS, referred to as PTIBS in Fig. 1. The assumption here is that PTIBS processes are executed by the VR as parts of the procedures to refresh the cached data triples or retrieve values of links stored within the VR. This way, no change to the client sources is required to use TIBS.

The accountability clause associated with a set of released data specifies the minimum user identity and traceability requirements for releasing the data. (As an example, the interior layout of a private building may be released to rescue workers as long as they download the layout using valid cell phone accounts. Later, in the recovery phase, these workers can be tracked down based on their cell phone numbers and will be held accountable for actions enabled by the information provided by the layout.) The clauses are enforced by the retrospective part of TIBS, referred to as RTIBS. In contrast to PTIBS, RTIBS should be added (and can be easily added) to some applications, e.g. C3 and ADAST used by an EOC where accountability can be tracked more effectively. For other applications, such as MAD and many mobile emergency response applications, it makes sense to have RTIBS, i.e. accountability processes, executed by the VR on POSs.

It is worth noting that TIBS has three types of functional modules: administrators, providers and helpers. It has at least one administrator, running on a resource-rich POS server or on a cloud. The administrator maintains definitions of user roles and release and accountability policies and binds them with the data released by PTIBS. It also serves RTIBS processes as an audit center for tracking down information abuses and identifying the abusers. TIBS providers run on POS servers that hold some of the proactively released data. They enforce release and accountability policies during dissemination of the data, and maintain transaction audit records as required by the accountability policies. A provider may also be equipped with a rule/attribute-based access control (AC) subsystem. To illustrate this point, Fig. 1 shows that a C3 system makes use of this capability. Finally, TIBS helpers are components on ultra-lightweight POS servers. Their main function is to keep track of ad hoc transfers of data that have been released. One approach is to use a lightweight RESTful web proxy on

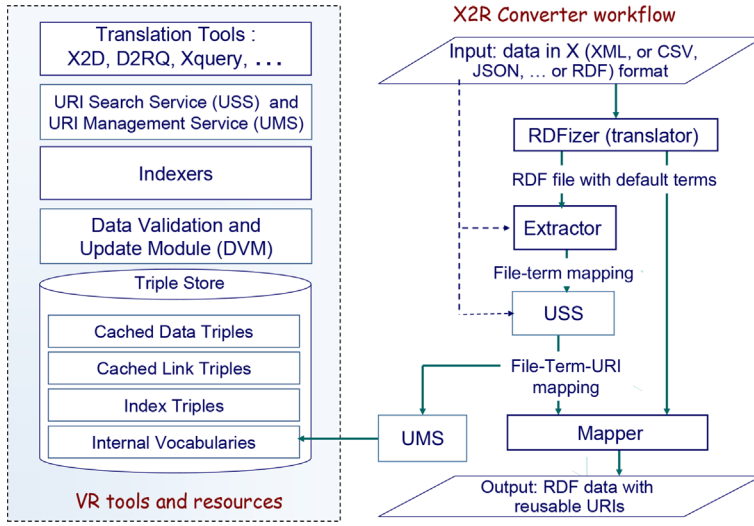

Figure 2: VR tools and resources.

POS servers. The proxy will produce metadata that tags each and every data transfer across user-specified information domains. Later, when the need arises, the metadata tags can be traced to reconstruct the flow of each data set released under the care of the helper.

2.3 VR resources and tools

Figure 2 provides further details on the box labeled VR tools and resources in Fig. 1. Specifically, the left part of Fig. 2 shows some of the important tools provided by VR. The right part of the figure shows the workflow among component tools within the X2R for incremental transformation of legacy data into LD. Section 4 will describe the converter as well as URI search and management services (USS and UMS) used by the converter.

Indexers are tools for building indexes of contents of media files. These files are typically in formats such as JPEG, TIFF, AVI, MP3, etc. that cannot be represented in an RDF format. The indexes built by these tools are in RDF N-triple format and are stored in the triple store of the VR as shown in the figure.

Finally, data validation and update module (DVM) is responsible for maintaining the consistency of data stored within the VR with external data stored in the original sources. Each triple or set of related triples in the VR triple store has a valid interval. The interval is specified by the client application to which the triples belong. A triple is considered expired and hence invalid when time elapses beyond its valid interval. When this happens, the DVM calls the update procedure associated with the triple to update it.

3 A CASE STUDY

As stated earlier, the X2R converter, described in the next section, is a system of tools for converting data in legacy formats retrieved from existing DMIS into LD in RDF format. Its intended users include developers of LD-based client applications served by a VR. To provide rationales and motivation behind use case development, requirement capture and user interface design of X2R, the development of MAD was used as a case study: MAD developers have

converted open data retrieved from multiple cities in diverse formats into LD in RDF format without the help of the X2R converter. By observing the conversion processes carried out by MAD developers, the steps that are challenging and time consuming for some or all of the developers were identified. The current version of the X2R converter, described in Fig. 2 by the workflow of the conversion process and interfaces between component tools, aims not only to ease developers' efforts by automating challenging steps as much as possible but also to improve the quality of the RDF outputs produced by the converter.

Specifically, the conversion processes of two MAD developers, hereafter referred to as Joe and Bob, were followed. Joe had worked with Taipei City Open Data developers on metadata and application program interface (API) functions. He is familiar with RDF formats and related techniques, as well as commonly used formats, such as JSON-LD and RDF/XML. In contrast, Bob is a novice. Both Joe and Bob are familiar with the raw data, which are a collection of data sets on emergency response facilities retrieved from open data sources of three cities: Taipei, Taiwan; London, England and Yokohama, Japan. The formats of the raw data sets are diverse, including CSV, XML, JSON and YAM. Again, the developers must convert all the raw data to RDF format, the format of data exchanged among MAD components.

Figure 3a shows the workflows within Joe's and Bob's conversion processes. When presented a file of raw data, the first step for each of them was to select an RDFizer among all translators that are capable of parsing and translating some of the raw data. During the case study, Joe's choice was RDF Translator [17] while Bob's choice was OpenRefine [18]. The former is a multi-format translation tool. Its service is triggered by either a URI or text input. The tool can automatically detect several input formats (including RDFa, N3, XML and JSON-LD) and produce output in one of many common formats, including RDF. The latter is a general-purpose tool, for cleaning up messy data, as well as transformation of input data in an even wider spectrum of formats.

Preprocessing of the input data is typically necessary in order to capture the semantics of the input data and make the data conform to the conventions of the selected RDFizer. Joe chose to do so, as shown in Fig. 3a. In contrast, after estimating the effort needed to make input data conform to the complicated conventions of his RDFizer, Bob decided to skip this

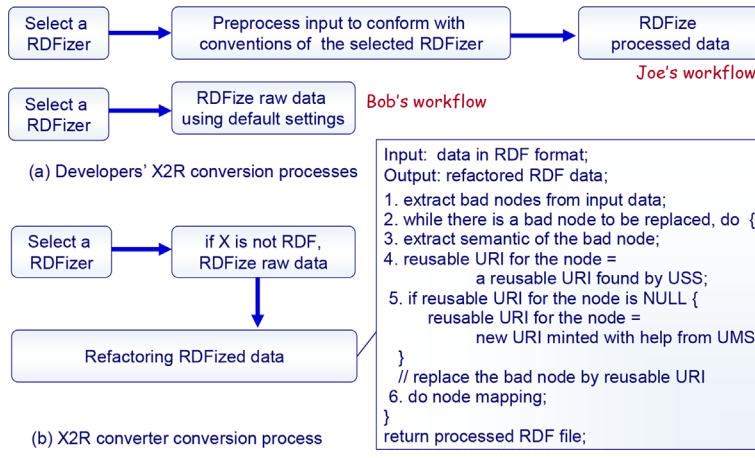

Figure 3: Conversion processes of application developers and X2R converter.

step and to use the default settings. As a result, his RDF output data have correct syntax but suffered semantics loss during the translation.

Both RDF Translator and OpenRefine are excellent RDFizers, but they do not support all the formats of the open data retrieved by the MAD interface server. In fact, no existing RDFizer can convert directly raw data in all the formats encountered by Joe and Bob to produce output of satisfactory quality. The developers chose to overcome this problem by first converting data in a non-supported format into a supported format using other tools or customized scripts. For example, Joe handcrafted a script and used it to inject URIs and needed headers to transform raw data in JSON format to JSON-LD format, which is an acceptable input format of RDF Translator. Bob used an online tool to convert YAM files to CSV files. This kind of solution requires as many scripts or tools as there are input data formats not supported by the selected RDFizer. An alternative is to use multiple RDFizers. One can easily see that the expertise and efforts needed to process input data in multiple formats to conform to multiple RDFizer-specific conventions can seriously discourage application developers from transforming legacy data into LD in RDF format.

By far, the most tedious and time consuming tasks in the X2R conversion process for both Joe and Bob are searching external ontologies for reusable URIs and creating new URIs that can be reused by others. Joe and Bob relied on general-purpose search engines, which are not effective for searching URIs. Semantics search engines or RDF endpoints are relatively effective, but require their users to learn SPARQL. Based on statistics published by W3C on LOD, there are more than 31 billion RDF triples as of September 2011 and growing. For sufficient coverage, the developers would have to issue queries to multiple semantics search engines and aggregate the returned results. In addition, they must deal with issues such as result ranking and disambiguating and query refinement. Consequently, the developers spent more than 50% of the time on these tasks. An inexperienced developer like Bob is also challenged by the task of injecting reusable URIs into raw data to meet the convention of his selected RDFizer.

4 X2R CONVERTER

The design choice based on observations from the case study was to make the X2R converter work as shown in Fig. 3b. The converter uses multiple RDFizers to support formats of input data. Rather than preprocessing input data in diverse formats according to conventions of multiple RDFizers, the tool post-processes the output RDF files produced by RDFizers. An obvious advantage of post-processing is that the tool needs to manipulate only RDF model and format. The post-processing step is called *refactoring*, a term borrowed from software engineering: Refactoring is a technique to improve the code and to ensure that the changes are safe. Here, RDF refactoring is done to improve the quality, in terms of reusability, and to ensure syntactic correctness of the output RDF data.

4.1 Extract bad nodes and semantics

As stated in Section 1, output data generated by RDFizers often contain bad nodes: They are nodes with RDFizer-generated URIs and blank nodes. RDFizer-generated URIs are temporary and may even be unreachable. Such URIs violate the design principles recommended by Tim Berners-Lee [5]: Use URIs people can easily look up and include links to URIs to enable their discovery. Blank nodes are undesirable because their identification is assigned by RDF parsers. Consequently, the identification of a blank node in an RDF data set changes after each processing, and the presence of blank nodes in a data set prevents the data from being

fully linked. Replacing blank nodes with URI referenced nodes is called *bnode skolemization* [19]. There is yet no automatic skolemization method.

For these reasons, the first step in RDF refactoring is the extraction from the RDF data output of the translator RDFizer-generated URIs and blank nodes with the intent to replace them with reuable URIs. Reusable URIs are either already in use and hence can be found in commonly used ontologies or are created by the developer following the recommended design principles and placed in the VR internal vocabularies to be shared by other client applications.

The extraction work is done by the component tool called extractor, as shown in Fig. 2. To detect bad nodes, the extractor parses the RDF data produced by the RDFizer and extracts all URI references and blank nodes. It then examines each extracted URI reference to determine whether the URI can be looked up through the HTTP protocol. URIs that cannot be thus looked up are treated as bad nodes along with all the blank nodes. Another function of the extractor is sense-making, i.e. to extract semantics of bad nodes to facilitate the search for sense-compatible and reusable URIs, ideally widely used ones, as their replacements. Sense-making relies on two kinds of information about each bad node: (1) context information for making sense of the node and (2) prior knowledge of term usage of commonly used URIs.

Context information is relatively easy to find for RDFizers that generate URIs using representative terms. Consequently, it is relatively easy to make sense of thus-named bad nodes. Some RDFizers generate URIs containing meaningless terms, as often is the case when default settings are used. For a node with a meaningless URI and for each blank node, analysis of the triple in which the node is involved, and sometimes, the whole RDF data set, is necessary to get context information for the node. The current version of the extractor requires the developer's help to complete this task.

Prior information on term usage means developers' knowledge of public URI vocabularies. In this respect, application developers are often at a disadvantage. During the case study described in the previous section, MAD developers had difficulties in selecting representative terms because of insufficient knowledge of widely used URIs and frequently used terms.

The current version of the extractor does sense-making semi-automatically. The developer needs to make the final decision of the extraction. In some cases, the developer needs to do human computation tasks to help the extraction process, especially for blank nodes. For novice developers, this is also a major challenge.

4.2 Search for and mint reusable URI

A design objective of the X2R converter is to automate as much as possible the searches for reusable URIs and creation of new URIs: The converter uses the USS and UMS provided by the VR to automate the tasks of searching and minting reusable URIs. Intervention from the developer is limited to the selection of a reusable URI among all the URIs found by USS from a specified list of ontologies as possible replacements of a bad node (i.e. line 4 in the pseudo-code in Fig. 3b) and creation of a new reusable URI when the developer rejects all the URIs found from existing ontologies (i.e. line 5). The pseudo-code describes an interactive mode of the converter: The developer and the tool collaboratively find/mint a replacement for one bad node at a time. The converter can also work in batch mode, as described by the workflow shown in the right half of Fig. 2. In batch mode, the converter requests USS to find candidate replacement URIs for all bad nodes and presents all the candidates for each of the bad nodes to the developer for selection (or give approval of the tool's selection) after the search.

USS provides a Graphical User Interface (GUI) and an API function to support both interactive and batch searches. Once USS receives a searched term via the GUI or the API function, the service connects to each of the specified ontologies in turn, queries each of them for URIs based on the search term and integrates URIs returned by the searches into a list of reusable URIs as the result of the search. In this way, USS enables the user to treat all external ontologies as a single source and thus avoid the need to search them individually manually.

After USS returns a URI list, a URI from the list is then selected to represent the term for which the search was performed. The X2R converter allows the developer to make the selection or if the developer so chooses, makes the selection automatically and presents the selection for developer's approval.

Following a commonly adopted principle [20], a new URI is created only when a search of the specified list of ontologies fails to find a suitable URI for the search term. The X2R converter relies on UMS to assist the developer to create and manage new URIs. UMS is built on Neologism, a vocabulary publishing platform for the Web of Data [21]. Every new URI created by UMS is stored in the VR Internal Vocabularies to be shared by client applications of the VR. UMS allows both manual and automatic addition of new URIs. Specifically, the GUI of UMS lets the developer to manually classify and store new URIs offline. UMS also provides API functions using which a client application can request automatic addition of one or more URIs into VR Internal Vocabularies.

4.3 URI-node mapping

The last step in RDF refactoring is node-URI mapping. In this step, the reusable URI found by USS or minted by the developer and UMS for each bad node in the RDF data returned by the RDFizer is used to replace the translator-generated URI of the node or to name the blank node. Mapper is the X2R component responsible for this task as shown by the workflow diagram in Fig. 2 and line 6 in the pseudo-code in Fig. 3. The mapper uses as input a node-URI mapping file that contains an association of (bad node, replacement URI) of each bad node extracted by the extractor and the replacement URI. The tool automatically does the replacement and skolemization.

The process of node-URI mapping and the generation of the final RDF file is semi-automatic. The developer selects a reusable URI for each extracted node, and the tool generates the RDF file based on the developer's selections.

5 SUMMARY AND FUTURE WORK

This article first presented an overview of a middleware-level system of resources, services and tools called VR [2]. A VR facilitates the access and use of data by its client applications from independent sources and provides the applications with interface services to support their own models and views of the data. It is a key element of a framework for building open information systems for disaster preparedness and response from independent data sources for independent applications.

The VR is also a development environment for incremental transformation of data in legacy formats into LD in RDF format. The targeted users of the system of tools called the X2R converter are developers of LD-based applications. Rather than having to wait for owners of data sources to provide them with LD, the X2R converter enables application developers to be producers of LD on demand as needed by their applications.

When presented with a data set in a legacy format as input, the X2R converter first uses an existing translator (e.g. one of the RDFizers supported by the MIT Simile Project) that

supports the input format to translate the data into an RDF format without first pre-processing the data to meet the conventions of the translator. The RDF data thus produced by the translator often contain bad nodes, i.e. blank nodes and nodes identified by unreachable URIs. The presence of bad nodes prevents the data to be fully linked and hinders the discovery of the data. In a post-processing step, the X2R converter works collaboratively with the developer to eliminate these nodes, naming them with reusable URIs found from existing ontologies, or created to be reused in cases when no suitable URIs can be found.

The previous sections presented rationales and motivations behind major design decisions of the X2R converter. These decisions were made after observing the processes carried out by two MAD developers to convert data on emergency preparedness facilities retrieved from open data sources of multiple cities. Even for a relative simple application like MAD, developers have to deal with many input data formats, which no single RDFizer can translate. The efforts in pre-processing input data to conform to conventions of individual translators grow with the product of the number of input formats and the number of translators. The fact motivated the post-processing step of the RDF data produced by the translator, rather than pre-processing the input data of the translator. The observation that searching for reusable URIs consumes a significant part of the developer's time during their conversion processes motivated the automation of the search of multiple ontologies for reusable URIs and thus, enabled the search of all the specified external ontologies as if they were a single source.

A version of the X2R converter is functional. Its code is being restructured to improve its testability and maintainability and will be released under GNU General Public License license when this phase of its development completes. The tool will be used on an experimental basis by developers of MAD and other LD-based applications in a more extensive case study in the future to determine the extent the tool meets its design goal of significantly reducing the levels of expertise and effort required to convert diverse legacy data into linked RDF data.

ACKNOWLEDGEMENTS

The authors would like to thank S. H. Tsai, Y. A. Lai, J. C. T. Hsiao, E. T.-H. Chu, K. J. Lin and J. K. Zao for their contributions to the design and implementation of the VR presented here. This work was supported by the Academia Sinica thematic project OpenISDM.

REFERENCES

[1] Murphy, R.R., A National Initiative in Emergency Informatics. Computing Community Consortium, 2010.

[2] Ou, Y.Z., Tsai, S.H., Lai, Y.A., Su, J., Yu, C.W., Hsiao, C.T., Chu, E.T.H., Lin, K.J., Ho, J.M. & Liu, J.W.S., A linked-data based virtual repository for disaster management tools and applications. *WIT Transactions on the Built Environment*, **133**, pp. 161–173, 2013. doi: http://dx.doi.org/10.2495/dman130171

[3] Zao, J.K., Nguyen, K.T., Wang, Y.H., Lin, A.C.H., Wang B.W. & Liu, J.W.S., Trustworthy emergency information brokerage service (TIBS). *WIT Transactions on the Built Environment*, **133**, pp. 216–227, 2013. doi: http://dx.doi.org/10.2495/dman130221

[4] Bizer, C., Heath, T. & Berners-Lee, T., Linked data – the story so far. *International Journal on Semantic Web and Information Systems*, **5(3)**, pp. 1–22, 2009. doi: http://dx.doi.org/10.4018/jswis.2009081901

[5] Berners-Lee, T., Linked Data, available at http://www.w3.org/DesignIssues/LinkedData.html. doi: http://dx.doi.org/10.4018/978-1-60960-593-3.ch008

[6] Bauer, F. & Kaltenbock, M., Linked Open Data: the Essentials, available at http://www.semantic-web.at/LOD-TheEssentials.pdf
[7] Borges, M.R.S., de Faria Cordeiro, K., Campos, M.L.M. & go Marino, T., Linked open data and the design of information infrastructure for emergency management systems. *Proceedings of International Conference on Information Systems for Crisis Response and Management*, 2011.
[8] Silva, T., Wuwongse, V. & Sharma, H.N., Disaster mitigation and preparedness using linked open data. *Journal of Ambient Intelligence and Humanized Computing*, **4(4)**, pp. 591–602, 2013. doi: http://dx.doi.org/10.1007/s12652-012-0128-9
[9] Schulz, A., Doweling, S. & Probst, F., Integrating process modeling and linked open data to improve decision making in disaster management. *Proceedings of the CSCW Workshop on Collaboration and Crisis Informatics*, Vol. 9, pp. 16–22, 2012.
[10] LOD2, Project, available at http://lod2.eu/WikiArticle/Project.html
[11] SIMILE, Project, available at http://simile.mit.edu/
[12] Converter to RDF, available at http://www.w3.org/wiki/ConverterToRdf
[13] Stolz, A., Rodriguez-Castro, B. & Hepp, M., RDF translator: a RESTful multi-format data converter for the semantic web. Technical Report TR-2013-1, E-Business and Web Science Research Group, 2013.
[14] Generic XML to RDF converter, an open source project hosted in SourceForge, available at http://sourceforge.net/projects/xmltordf/
[15] RDFizer Concept, available at http://wiki.opensemanticframework.org/index.php/RDFizer_Concept
[16] Lee, C.R., Ou, Y.Z., Yu, C.W., Tsai, S.H., Chern, F.R. & Liu, J.W.S., Intelligent active storage service. Presented in Work-in-Progress Session of International Workshop on Resilient ICT for Management of Mega Disasters (RITMAN 2012), December 2012.
[17] RDF Translator, an online service, available at http://rdf-translator.appspot.com/
[18] OpenRefine, an open source translation tool, available at http://openrefine.org/
[19] BnodeSkolemization, available at http://www.w3.org/wiki/BnodeSkolemization
[20] Heath, T. & Bizer, C., *Linked Data Evolving the Web into a Global Data Space*, Morgan & Claypool, 2011. doi: http://dx.doi.org/10.2200/s00334ed1v01y201102wbe001
[21] Neologism, a linked-data vocabulary publishing platform, available at http://neologism.deri.ie/