Data Management Requirements for a Knowledge Discovery Platform

Liming Zhu, Len Bass, Xiwei Xu
NICTA, Australia Technology Park, Eveleigh, Australia
School of Computer Science and Engineering, University of New South Wales, Australia
Firstname.Lastname@nicta.com.au

***Abstract*— This paper provides some requirements for the data management portion of a knowledge discovery ecosystem platform. The requirements are functional – what the platform should provide for its clients; quality – how the platform should support modifiability, performance, and availability; and management – how the platform supports operational control to sites that use it. It also provides design guidance that reflects the lack of central management that exists in an ecosystem.**

Keywords- data management, platform, knowledge discovery, software ecosystem, architecture

I. INTRODUCTION

"The purpose of computing is insight, not numbers." This statement is even truer today in the era of "big data" than it was in 1971 when Richard Hamming coined this phrase. In today's environment, massive amounts of data are available from a wide variety of sources, in a wide variety of forms, and with a wide variety of provenances. Analysis and visualization tools are equally variable. The tools used and the individuals involved in the generation of insight from data form an ecosystem or multisided market [1]. Platforms to support the ecosystem of knowledge discovery are certain to emerge, at least for particular vertical industries or domains where some commonality and controlled variation exists.

There are three portions to a platform to support knowledge discovery: data management, data analysis, and data visualization. Designing any of these portions involves significant difficulty. We are initially focusing on the portion of the knowledge discovery platform that manages the data.

The question now becomes "what are the requirements that such a data management platform should satisfy?" That is the subject of this paper. We present data management requirements and design principles derived from our experience in building or designing platforms for several different domains to support knowledge discovery. While we acknowledge that our requirements and design principles are almost certainly incomplete, they do provide a starting point for the development of a platform for data management in support of knowledge discovery.

II. SOURCES

In this section, we describe two experiences with either constructing or designing platforms for the processing of data. The first is for the Australian mortgage lending industry and the second for the integration of data about a particular domain from the different states in Australia into a form uniformly accessible. We also describe what it means to be an ecosystem involving a knowledge discovery platform.

A. The Australian Mortgage Industry

Vertical industries (such as mortgage lending) have been developing e-business standards to improve their business-to-business data management. LIXI (Lending Industry XML Initiative) [2] is an Australian e-business standardization body that serves the consumer loan industry. LIXI e-business standards cover a wide range of business data management and exchange scenarios. Some exchanges are transactional, such as loan application processing. Others are non-transactional such as loan product information dissemination where lenders (e.g., banks) communicate new and updated loan product information to brokers, mortgage house and borrowers. The latter usually requires the unidirectional secure dissemination of large quantities of frequently changed and time-sensitive data from lenders to brokers/borrowers. The lenders own the data and its management. Intermediaries may add value to it by aggregating data from different lenders and performing value-adding analysis and manipulation before republishing in the ecosystem. A wide range of data receiver capabilities has to be considered, in terms of technical sophistication on the organization level, device limitations (e.g. mobile devices for field operators) and basic data management requirements (e.g. sorting, filtering, annotation, refreshing). Costly infrastructure is often not the best solution.

We designed and built (in the form of reference implementations) a data management platform that disseminate the data using ATOM Publishing Protocol (APP) [3]. We used a customized commercial data-schema mapping tool to map lender-specific product schema and other provenance information to LIXI schemas and elements of the APP. We built a tool to then automatically generate product information feed in LIXI and APP compliant fashion. We also built mobile-based and Excel-based LIXI/APP feed consumption components to support the less-IT-sophisticated data consumer in the ecosystem.

B. Integrating Data from Different States

Each state in Australia performs data collection and analyses of particular household information (e.g. transport survey or property transactions). As might be expected, each state collects this information in its own fashion, uses its own schema to define the data, and maintains ownership and governance over its data. Some aspects of a schema are even

bound to state laws and cannot be simply harmonized through a nation-wide standard schema.

The goal of the platform we are designing is to allow access to an integrated version of the data collected by all of the states. We are doing this by harmonizing the data to generate a schema for user access to the data and then converting requests for data, and potentially analysis, into a form appropriate for the sources of the data. We expect this will also involve tools that are able to decompose certain analysis and generate analysis results incrementally so that an analysis can be partially done on certain state data sets. The provenance of the data is also important. Some data might be derived; other data might be raw. Data might change as a result of the correction of errors and analyses that were performed with erroneous data may need to be repeated.

C. *Ecosystems*

Software ecosystem is a new conceptualization of large-scale software development. A Software Ecosystem consists of the set of software solutions that enable, support and automate the activities and transactions by the actors in the associated social or business ecosystem and the organizations that provide these solutions [4]. An ecosystem adds some unique challenges [5]:

- **Decentralization**: Data, development, evolution and operational control are all decentralized. For example, LIXI is a non-profit organization with no standard enforcement power. Its membership is voluntary.
- **Inherently conflicting requirements**: Most parties want complexity to reside in others' parts of the overall system and want information to be shared, but do not want to share their own information. Technical solution companies provide and favour intermediary gateways and custom-built applications, while smaller players typically want commoditized applications and no intermediaries.
- **Continuous evolution with heterogeneous elements**: The whole ecosystem cannot be stopped and re-engineered. Day-to-day data management activities have to go on, and horizontal interactions with the larger systems also exert constant influence.
- **No clear people/system boundary**: The scale of the involved organisations varies widely. Some companies have sophisticated systems that can automate most tasks, while others still rely on fax and manual processing.

In an ecosystem based on a knowledge discovery platform, data may be available through a set of external value-added-resellers, and a community of users building and sharing customizations [6]. Every component in such an ecosystem has its own reason for existence and its own management. This means there is no possibility of overall direction. For example, the use of APP as a Web-friendly API for publishing loan product information essentially opens up the competitive value-adding aggregator market where aggregators consume through APP, perform analysis and add value, and republishes in APP often transparently. Another example is the publishing of mappings between schemas in the state data integration example to effectively allow more innovative and smart applications combing two different data sources in the ecosystem.

III. MOTIVATING USE CASES

In this section we present some of the motivating use cases that led to our set of requirements. The use cases represent situations that are to be expected when utilizing a platform for the data management portion of knowledge discovery. Clearly, this list is incomplete but it is indicative of the types of situations that might arise.

U1. A data analyst wishes to combine unformatted, tabular, and graph based data in order to perform an analysis. One set of data comes from a survey of households, another set represents the search queries emanating from that same set of households, and a third set consists of the social networks of the members of that set of households. Furthermore, the form of the tabular data has changed from one year to the next.

U2. One of the services provided in the infrastructure is updated. Those responsible for services that depend on the updated service should be notified with information about what has changed.

U3. One of the services provided in the platform fails. The service falls back to a degraded service mode. The data produced by this service is annotated to indicate it was produced by a service operating in degraded mode.

U4. The OCR that scanned the data from a survey is found to have an error that results in misreading the survey answers. Consumers of the data are informed, in case their analyses are affected.

U5. One company located in the United States and another located in Germany establish a joint venture to develop a new product in the insurance space including health information for the insurers. The product must adhere to relevant data privacy laws for health data.

IV. REQUIREMENTS

We use our list of use cases to motivate more specific requirements for a platform. As with the use cases, we do not expect these requirements are exhaustive but they are indicative. We omit some obvious requirements such as authentication and authorization. We divide our requirements into those that apply to the platform's services or qualities those that apply to the design of the platform.

A. *Services and qualities of the platform*

R1. Allow clients to read/write data according to client specified formats. Data for knowledge discovery exists in a wide variety of structured, semi-structured, or unstructured forms. Tabular data, graph-oriented data, document centric data, textual data are all popular forms of data storage. The platform should have the ability to serve data to a client according to the client's preferences. This requirement suggests the need for a wide variety of converters both syntactic and semantic.

R2. Allow clients to dynamically add or delete data sources. Discovery of data sources is not a service we anticipate to be within the data management platform since discovery may involve contractual obligations but once a data source has been discovered and access has been negotiated, clients should have the ability to dynamically access the source without the necessity for re-configuration.

R3. Data shall be accompanied by meta-data that includes the history, validity, or uncertainty of the data. This requirement suggests the need to be able to identify and, possibly furnish, missing data or inferred data. It also suggests the need to track usage of the data since the clients of data that is modified subsequent to the usage may need to be alerted to the fact that the data used in their analyses has changed.

R4. The platform shall support collaboration among different analysts over the data. This collaboration will be either synchronous or asynchronous. This suggests that consistency is important for data being accessed by multiple clients.

R5. Clients shall be able to access the data according to a variety of different patterns. The client can subscribe to data that is being created, the data can be moved to the client, or analysis can be moved to the data. In addition, Data shall be available for batch as well as interactive analysis.

R6. All tools should be accessible from a wide variety of hardware and operating systems. Heterogeneity is a given in today's computing environments. Data can be stored, analyzed, or visualized on hardware platforms ranging from supercomputers to servers to desktops to mobile devices. Although some tools may be restricted to specific platforms, supporting access to these tools through a web frontend or a virtualized environment is necessary.

R7. The platform shall be able to accommodate change in data management tools. The methods for data management are continually evolving and domain specific. New tools are continually arriving.

R8. The platform shall support the volume of data available from data sources. Data sources may be on the order of peta-bytes. The platform should provide tools to manage this amount of data in a cost effective fashion. For example, efficient data query API, streaming or moving the analysis to the data rather than moving the data to the analysis are possible techniques to support large data volumes.

R9. The platform shall be aware of the dependencies among the clients and the services within the ecosystem. When one service is dependent on another; the platform should provide that information and inform system administrators when a service anywhere in the dependency chain is modified.

R10. A failure in one service shall not affect the availability of a client. Failure of instances in modern computing environments should be expected. Each service should detect and take corrective action when a failure occurs. Furthermore, each service should provide a real time window into its current availability [7]

R11. The platform shall be aware of relevant privacy/location regulations and raise alerts if a violation of a regulation occurs. It may not be feasible to move data to remote analysis sites either due to privacy/security concerns.

B. Design requirements

In an ecosystem, there is no central control. Data consumers and providers have to cooperate. We observe the following requirements for design drawn from the LIXI experience [8].

D1. Use principles and rules to influence design rather than prescribe structures. An architectural rule may be satisfied by several potential structural architectures.

D2. Influence but do not control others. Decentralization is one of the main characteristics of an ecosystem. Additionally, many ecosystems are collaborative rather than hierarchical. In such settings, influence instead of control is the main mechanism to achieve data management interoperability and improve overall system quality. This set of rules encourages the use of influence through micro-format proposals and optional design alternatives.

D3. Use Minimal service interface between data consumers and data providers. Ecosystems should use message-centric (rather than operation-centric) interfaces in data sharing. That is, service interfaces should not expose abstractions in the form of remote procedures. Essentially, we advocate the use of a single operation on a service but allow more complicated interfaces to exist. This rule encourages maximum flexibility in the face of constant evolution. Ever-changing shared contexts are carried within messages.

D4. Share Metadata and Context. Metadata is usually described in data sharing contract. Contexts are more instance-specific. We encourage metadata and contexts to be shared in all possible ways. Such metadata can be related to policies (e.g. security requirements or encryption capabilities), quality of service characteristics (e.g. required response time), and semantic descriptions. Through the sharing of metadata and context, interoperability can be achieved at both design time and run-time with little top-down prescriptive planning.

D5. Avoid Explicit Intermediaries. Do not introduce the role of an intermediary explicitly in ecosystem data management platform reference architecture. However, allow such intermediaries to organically appear in the overall ecosystem. This is very different from existing e-business me-ta-standards such as ebXML [9], which have an explicit concept of central registry and repositories through which companies post business processes, capability profiles and collaboration protocol agreements. Technically, this is appealing and simplifies some business scenarios. However, it may be very difficult to introduce such a structure within an ecosystem because of complex business issues such as who the intermediaries should be, legal issues such as

confidentiality concerns, and practical issues such as the difficulty of semi-automated agreement negotiation.

V. Types of tools envisioned

In this section, we enumerate some of the types of tools that we envision being a portion of the platform.

T1. Basic data management tools for large data such as those found in Hadoop [10].

T2. Data mapping and model transformation tools for data formats or data schemas such as [11], [12], [13] and [14]. A vocabulary management tool [12] will allow semantic mapping between different state schemas or between state schemas and the harmonized schema without forcing a conversion. Such mapping could be published and used for various purposes in both data management and analysis.

T3. Collaboration tools such as those presented in [15].

VI. Missing requirements

We have enumerated a set of requirements and design principles taken from our experience and the literature. This list is almost certainly not complete. The important question is not so much "what requirements are missing?" as "how do we find the missing requirements?" Normal requirements elicitation involves collecting requirements from stakeholders including the developers. The stakeholders in an ecosystem are not known *a priori*. Finding the correct process for determining requirements for an ecosystem is one of the open questions with which we are concerned.

VII. Acknowledgment

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.

References

[1] D. S. Evans, A. Hagiu, and R. Schmalensee, Invisible Engines: How Software Platforms Drive Innovation and Transform Industries, 1st ed.: The MIT Press, 2006.

[2] LIXI. Lending Industry XML Initiative. Available: http://www.lixi.org.au

[3] L. Zhu and B. Thomas, "LIXI Visible Loans: Reference Architecture and Implementation Guide," Lending Industry XML Initiative (LIXI)2007.

[4] J. Bosch, "From software product lines to software ecosystems," presented at the Proceedings of the 13th International Software Product Line Conference, San Francisco, California, 2009.

[5] L. Northrop, R. Kazman, M. Klein, D. Schmidt, K. Wallnau, and K. Sullivan, Ultra-Large Scale Systems: The Software Challenge of the Future. Pittsburgh: SEI, 2006.

[6] S. Jansen, A. Finkelstein, and S. Brinkkemper, "A sense of community: A research agenda for software ecosystems," in Software Engineering - Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on, 2009, pp. 187-190.

[7] B. Christensen. (2012). Fault Tolerance in a High Volume, Distributed System. Available: http://techblog.netflix.com/.

[8] L. Zhu, M. Staples, and V. Tosic, "On Creating Industry-Wide Reference Architectures," in The 12th IEEE International EDOC Conference (EDOC'08), Munich, Germany, 2008.

[9] ebXML. (2012). Electronic Business using eXtensible Markup Language (ebXML). Available: http://www.ebxml.org/.

[10] Apache. (2012). Hadoop. Available: http://hadoop.apache.org/.

[11] . Stylus Studio X12. Available: http://www.stylusstudio.com/.

[12] NICTA. (2012). Vocabulary Manager (VM). Available: http://vocab.ext.nicta.com.au/vocab/.

[13] I. Gorton, C. Sivaramakrishnan, G. Black, S. White, S. Purohit, M. Madison, and K. Schuchardt, "Velo: riding the knowledge management wave for simulation and modeling," in 4th international workshop on Software engineering for computational science and engineering (SECSE '11), 2011.

[14] J. C. Grundy, J. G. Hosking, R. W. Amor, W. B. Mugridge, and Y. Li, "Domain-specific visual languages for specifying and generating data mapping system," Journal of Visual Languages and Computing, vol. 15, pp. 243–263, 2004.

[15] Wikipedia. (2012). Collaborative Software. Available: http://en.wikipedia.org/wiki/Collaborative_software.

National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.