

# Next Generation Application Integration: Challenges and New Approaches

Ian Gorton, Dave Thurman, Judi Thomson

*Information Sciences and Engineering,  
Pacific Northwest National Laboratory, Richland, WA 99352, USA  
Ian.Gorton@pnl.gov*

## Abstract

*Integrating multiple heterogeneous data sources into applications is a time-consuming, costly and error-prone engineering task. Relatively mature technologies exist that make integration tractable from an engineering perspective. These technologies, however, have many limitations, and hence present opportunities for breakthrough research. This paper briefly describes some of these limitations, and enumerates a subset of the general open research problems. It then describes the Data Concierge research project and prototype that is attempting to provide solutions to some of these problems.*

## 1. Introduction

In the current and foreseeable heterogeneous world of information technology, complex application integration will remain an inevitability. Industry estimates vary, but as much as 70% of information technology spending may be devoted to integration-related activities. Consequently, it is no surprise that the technology world is awash with software solutions aimed at making integration a more tractable task.

Over the last decade, a whole industry sector has emerged, typically known as enterprise application integration (EAI) middleware [1]. In essence, these technologies provide software infrastructures and associated tools that attempt to make it easier to build new, overarching applications from legacy systems that were never designed to work together. More recently, as the Internet has fuelled electronic business-to-business (B2B) transactions, EAI technology has broadened to accommodate B2B application integration. Overall, this creates a complex landscape of competing and overlapping technologies, many of which are proprietary and will not interoperate!

The recent emergence of XML-based Web Services promises to provide a set of standards for application integration. At the time of writing, most vendors have Web Services-enabled their EAI and B2B technologies. While Web Services are not fully mature or proven, they seem likely to become the *lingua franca* of EAI and B2B. This will make integration simpler through common

transports, data formats and associated services (security, transactions, etc).

However, more widespread standards adoption will only alleviate some of the *accidental* [2] complexities of EAI. Fundamental problems concerning aspects such as semantics, automatic data discovery and integration, and scalability remain as challenges for the research community.

This paper will briefly describe the current state-of-the-art in EAI technologies, and then outline some of the myriad of research problems that exist. We then describe some of our work in attempting to provide solutions for flexible, adaptive integration technologies.

## 2. State-of-the-Art in EAI

Figure 1 depicts the major technological components in a generic EAI solution. A brief description of the roles and responsibilities of each is below.

**User Applications:** Like most IT applications, EAI is concerned with creating a capability for users wishing to view and manipulate data. What characterizes the need for EAI is that the required data is spread across multiple heterogeneous data sources, such as databases, ERP applications, web sites, file systems, and so on. The user application requirements can only be satisfied through a suitable integration solution. These applications, typically custom GUIs or browser-based, call services available in the Orchestration Layer to carry out their requests.

**Data Sources:** The integrated solution must read and write data from existing data sources. In almost all cases, the solution must access the data sources 'as is', because other existing stovepipe applications already rely on their defined schema. It is often necessary to create new, temporary data sources to act as staging areas between systems for data consolidation and access. These temporary data sources are simply artifacts of the integration process.

**Orchestration Layer:** The Orchestration Layer accepts user requests and manages the fulfillment of these requests. A single input request may trigger a complex

sequence of events that access multiple data sources, transform message formats, and handle exceptions and conditional events, and so on. Orchestration involves defining and subsequently managing the process state and state transitions needed to accomplish a given business activity instance for a user application.

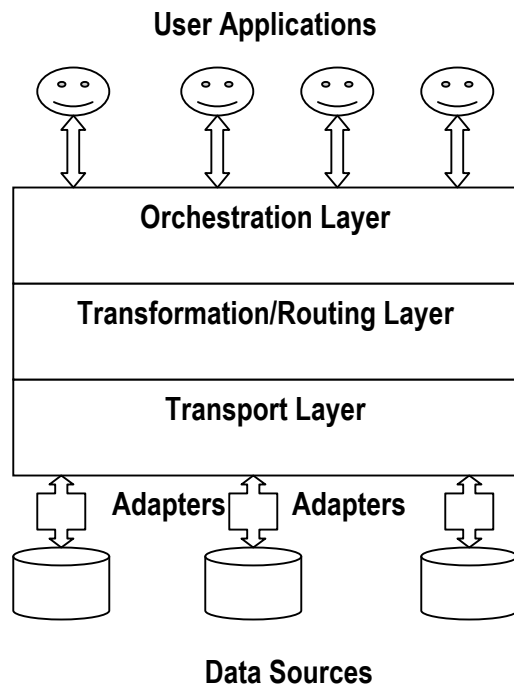


Figure 1 Generic EAI Architecture

**Transformation/Routing Layer:** As different data sources will have often radically different data formats, moving data from one to another requires data transformation. Transformations may be simple mappings from one schema to another, or be complex functions requiring multiple operations and calculations. In addition, a message may need to be sent to a given data source based on the content of some specified field. Hence the Transformation/Routing layer must contain technologies to define, manage and rapidly execute transformations from one data format to another in support of orchestrated business processes.

**Transport Layer:** The transport layer is responsible for moving data between data sources with a defined quality of service (QoS). For the majority of applications, preventing data loss is the most important QoS issue, as well as the ability to transactionally deliver batches of

messages. However, sometimes data loss can be accepted or compensated for to increase performance. Support for asynchronous, publish-subscribe messaging is also a key feature of an effective transport layer.

**Adapters:** Adapters encapsulate the specific access mechanisms for a data source. They typically offer a service-based API that abstracts the actual low-level calls necessary to read and write data. Some adapter architectures support the deployment of transformation services within the adapter. The latter are often somewhat grandiosely known as intelligent adapters.

Multiple vendors provide relatively mature products that support some or all of these layers of technology. The major products can be broadly characterized in to a number of families, namely:

#### Proprietary

Only since approximately 2001 have important standards emerged for various components of EAI solutions. Hence most products are based on proprietary innovations. They use proprietary development tools, languages and transformation engines, and run over proprietary transport mechanisms. Adapter architectures also vary, meaning that an adapter from one EAI product cannot be used with another EAI product. Overall, this has led to the 'integrate the EAI technologies' problem for organizations who have adopted different EAI technologies in different business areas.

#### J2EE-based

Java has been widely adopted as a development language within proprietary EAI products. However, the influence of Java is greater through the impact of the Java 2 Enterprise Edition (J2EE). J2EE has standardized a programming interface for the transport layer, the Java Messaging Service, and adapter architecture, the J2EE Connector Architecture (JCA). Both are seeing significant adoption levels, and promise to provide interoperability at the transport and adapter levels. J2EE-based products typically implement their orchestration and transformation engines as components within a J2EE container environment, providing further potential interoperability and portability. However, development tools vary widely, and there is no standardization as yet for transformation and orchestration definition.

#### .NET

The influence of Microsoft's .NET technology has been profound on the EAI market. The BizTalk Server orchestration technology and transformation engine, along with component-based adapters and asynchronous messaging has created a comprehensive product set for a broad section of the enterprise market. Of course this is

also a proprietary technology, but the pervasiveness of Microsoft technology in the market and BizTalk's penetration level warrant separate categorization.

XML-based Web Services technology is an orthogonal influence on EAI. All the above categories are embracing Web Services to provide a platform neutral integration mechanism. Despite potential performance problems and the current lack of mature accompanying services, the appeal of Web Services is immense. They are transport-independent, provide both synchronous and asynchronous operations, and make it possible through the use of XML documents to create extremely loosely coupled applications. They are also well supported in a multitude of development tools, and their development has essentially been automated.

Hence, the spectrum of EAI technologies is incredibly rich, diverse and complex. There appears to be a strong trend to rationalization of products and technologies based around emerging (de facto) standards for integration. It will no doubt take some time for the market to validate this trend and the leading innovators to emerge.

### 3. Challenges of Application Integration

While current EAI technologies are impressive in their scope and capabilities, there are many limitations. These arise from research challenges that remain to be solved, and from the ever evolving nature of applications and their data integration requirements.

Some of these challenges are described below:

**Scale:** The world is awash with digital data sources. New advanced commercial applications in analytics and visualization are beginning to require rapid access to multiple data sources, and the ability to rapidly fuse data from disparate formats. Hence next-generation integration technology must be able to scale to handle large numbers of data sources with flexible, fast and scalable format transformation engines.

**Dynamic configuration:** Existing EAI technologies must be statically configured to interact with an existing data source. This requires an adapter component to be at the minimum, acquired, configured to produce and consume messages, and deployed. When a suitable adapter for a data source doesn't exist, then one must be created. This is a non-trivial programming activity, costly, and it inevitably causes delays in accessing the required data. Next generation integration technologies must significantly reduce the cost and time to integrate new data sources. Ideally, it should be possible to locate a new

source of data, automatically deploy or generate a suitable adapter, and make the data immediately available to applications.

**Semantics:** Many modern programming technologies, such as Java and Web Services, support dynamic discovery of the syntax of a service or object interface. These reflective mechanisms make it possible for clients to connect to a server, retrieve the operation and parameter names/types, and dynamically construct calls. However, these mechanisms are devoid of semantics. They do not convey the meaning of the operation and parameter names. Hence the same name can be used in different interfaces for completely different purposes, and there is no way for a client to dynamically resolve this clash. Consequently there is a need for integration technologies to augment reflective interfaces with semantic descriptions of how the interfaces should be used. This will facilitate true dynamic discovery and integration.

**Finding Relevant Data:** As the scale of integration grows, it will become increasingly problematic to find data that is relevant to a user's or application's needs [3]. Most existing data sources are not accompanied by searchable catalogs and meta-data that can be used to locate data items of interest. This makes it imperative that next generation integration technology incorporates new techniques to automatically extract semantic information from data sources and link relevant data to applications.

Applications areas that require this new range of integration middleware capability abound. For example, organizational mergers and acquisitions cause major application integration problems that could be effectively reduced in magnitude through dynamic, adaptive integration infrastructures. New analytical applications in science, intelligence analysis and commerce have requirements to rapidly discover and use new sources of relevant data. These applications cannot afford the expense or delay of using engineering staff to build and/or deploy complex adapters for newly discovered data sources. And should the Grid [4] be successful in its aims, the ability of integration middleware to scale to handle huge numbers of data sources will become critical.

### 4. The Data Concierge Prototype

The Data Concierge aims to extend the state-of-the art in data integration middleware to address the above problems. We have developed prototype technology to demonstrate the new architectures and algorithms required for a generational breakthrough in data integration middleware, by transforming it to support a dynamic, adaptive mode of deployment and operation.

This will facilitate the rapid discovery, integration and use of new data sources in existing applications. Importantly, this should be achieved with minimal and, in some cases no administrative, software engineering, or user effort.

The core of the prototype is a reflective middleware platform [5] that can dynamically integrate new data sources. It employs knowledge-based techniques to classify the contents of a data source and the user's data interests. Users are then notified when new data that is likely to be of interest to them becomes integrated with the middleware platform. In this manner, users are rapidly connected to new data sources. The high level architecture for the Data Concierge is depicted in Figure 2. The following describes its functionality from two different perspectives, namely the data integration perspective, and the application user's perspective.

From an integration perspective, when a new data source becomes available, the *Data Source Discovery Subsystem* can automatically locate and integrate the data source with the reflective middleware platform. This is achieved by deploying a software component in the *Data Source Integration Subsystem* that can interact with the data and has a *rich service-based interface* that can be introspected and dynamically integrated into the platform. The data source's data and meta-data are then sampled by the *Data Source Classification Subsystem* using various algorithms to extract a representative summary of the data contents. The resulting data source summary is used to create a *knowledge signature* that captures the strength of the relationship of the data contents to a range of application needs expressed as domain ontologies. Ontology management and knowledge signature generation is handled within the *Knowledge Signature Subsystem*.<sup>1</sup>

From a user's perspective, data source queries are tracked by a daemon that is configured by the user to observe their application usage. In the *User Data Classification Subsystem*, various algorithms are used to characterize the types of data that an application uses. This classification data is also used to create a knowledge signature against the installed ontologies. A *Data-Use Matching* component then periodically executes and attempts to find new data sources that will be of interest to a user of an application. If a match is found, the user is notified and the middleware facilitates access to the data.

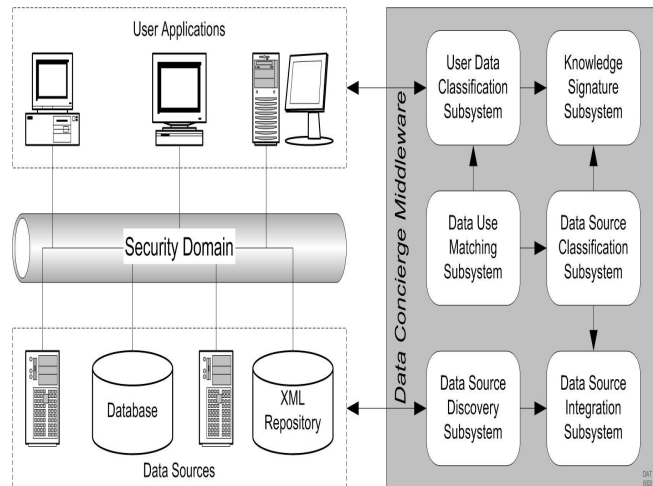


Figure 2 Data Concierge High Level Architecture

## 5. Current Work

There remains much research to be performed to further validate and enhance the prototype solution that exists. Some of the key areas are described below:

### Adapter Builder Tools

The Data Concierge middleware must be able to dynamically attach to and integrate with a new data source of arbitrary type (e.g., relational, non-relational, ftp, file system, etc). This requires the data source adapter component used for integration to describe the syntax and semantics of the access methods (operations, parameters) and the valid order of calls to the access methods for various types of interaction (read, write, etc). A solution needs to extend current work in software component description languages [6], and provide an 'adapter builder' tool to enable the integrator to generate the adapter component without explicitly writing any program code.

In the current prototype, we have designed and hand-coded specific instances of adapter components that have the necessary core syntactic and semantic descriptions for dynamic integration. This demonstrates the feasibility of the approach. The program code for the adapter is however complex. Consequently, the key research challenge that remains is to automate the generation of this code, along with the additional interfaces required for introspection.

The research involves designing a graphical design tool that takes as input a programmatic interface for a data source. The design tool then provides the facilities that the user needs to graphically describe the semantics, safety constraints and associated state machine that can be used by the Data Concierge middleware to access the

<sup>1</sup> Specifically we are working with DAML - <http://www.daml.org/about.html>

adapter. Once the design is complete, the tool will generate all the necessary code and reflective interfaces necessary for dynamic deployment and integration with the Data Concierge.

To constrain the problem, we are working with existing standards-based programmatic interfaces for data sources, such as Web Services and Java. These are inherently reflective, widely used, and make it easy for the design tool to parse them and present the interface details to the user. The targeted output will be an extended Web Service or Java compliant interface, along with XML descriptions for the safety constraints, semantics and state machine.

### Data Source Signature Tool

The data source classification subsystem must be able to extract a useful summary of the contents of given data source. Data source heterogeneity mandates that different types of meta-data and data will need to be utilized. In the current Data Concierge prototype, we have designed some relatively simple default mechanisms for different types of data source, but anticipate that human guidance will be needed in some situations to ensure useful data is included in the summary.

To this end, there is a need for a user-driven design tool that allows a data source administrator to specify the components of the data source that should be sampled and utilized in the data source signature. For example, an ftp site may have meaningless numeric filenames, but meaningful textual file content in one of more directories. For a database, certain tables may simply contain only key-related data, and hence should be excluded from the signature. Also, certain table and field names may or may not be relevant. Similar recent work is reported in [7]

The research involves creating a generalized tool that can view both the data and meta-data of a data source connected to the Data Concierge. It must then allow the data source administrator to graphically illustrate the information that should be used to produce the data source signature. The tool will then generate the queries and interface calls necessary for the data source signature production. The generated queries must be persisted in some form of script, so that the Data Concierge can periodically update the data source signature to capture new content.

In addition, the summarization process requires the design of new techniques to effectively characterize large data collections in a space- and time-efficient manner. We envisage building some 'intelligence' in to the data source signature generator that takes in to account the size of the data source and the cost of creating the

sample, and dynamically optimizes the generation process.

### Evaluating Matching Capability

One of the key elements that define the utility of the Data Concierge is its ability to reliably inform users of new data sources that are relevant to their work. In this component of the research, we are experimenting with a variety of algorithms for creating data and user signatures for efficient comparison.

The core matching functionality relies upon a technology we are defining known as *knowledge signatures*. A knowledge signature is a compressed mathematical representation of the meaning of some data in the context of one or more domain ontologies. Knowledge signatures can be compared using mathematical operations, and form an underpinning technology for the *Data-Use Matching Subsystem* of the middleware. Developing the algorithms to flexibly create and perform advanced reasoning on knowledge signatures is an open research problem [8] that is being addressed in this and a related PNNL project.

This Data Concierge is leveraging the knowledge signature effort, and devising algorithms and heuristics that attempt to ensure highly reliable performance in terms of signature matching from the knowledge signature subsystem. A number of experiments are being designed using different types of data sources, ontologies and optimizations. These will be evaluated and progressively refined in order to demonstrate the viability of this approach.

## 6. Conclusions

The Data Concierge prototype demonstrates that breakthroughs in integration technology are likely to require collaborations across a number sub-disciplines of computer science. The project is bringing together new techniques in software engineering, middleware, data analysis and artificial intelligence. Coupled with the necessity for constant consideration of legacy and new developments in data management technologies, effective research in application integration is likely to remain a highly technical, multidisciplinary undertaking.

## References

1. D. Linthicum, *Enterprise Application Integration*, Addison-Wesley 1999
2. F. Brooks, *The Mythical Man Month*, Reading, MA: Addison-Wesley, 1975
3. Iannella, R. *Metadata Repositories Using PICS* Proceedings of the First European Digital Libraries Conference, Pisa, Italy, 1-3 Sept 1997.

4. I. Foster, C. Kesselman, J. Nick, S. Tuecke, *Grid Services for Distributed System Integration..* Computer, 35(6), 2002.
5. F. Kon et al, *The Case for Reflective Middleware*, Comms ACM, vol 45, no 6, June 2002, pp 33-38
6. C. Szyperski, *Component Technology - What, Where, and How?*, Procs. Int. Conf. on Software Engineering (ICSE), May 2003, Portland, IEEE, pp 684-693
7. LLuis Gravano, Panagiotis Ipeirotis , Mehran Sahami, *QProber: A system for automatic classification of hidden-Web databases*, ACM Transactions on Information Systems, Volume 21 , Issue 1 (January 2003), Pages: 1 - 41
8. M. Rodr y-guez, Max Egenhofer, *Determining Semantic Similarity among Entity Classes from Different Ontologies*, IEEE Trans. On Knowledge and Data Engineering, (15), (2) 2003, pp 442-456