Data Services: The Marriage of Data Integration and Application Integration

A Whitepaper

Author:
Rick F. van der Lans
Independent Business Intelligence Analyst
R20/Consultancy

July, 2012

Sponsored by

Copyright © 2012 R20/Consultancy. All rights reserved. The Talend Platform for Data Services, Talend Open Studio, and The Talend Unified Platform are registered trademarks or trademarks of Talend Inc.. Trademarks of other companies referenced in this document are the sole property of their respective owners.

Table of Contents

1	Management Summary	1
2	Integration Is a Necessity	2
3	The Disintegrated Integration Solution	3
4	Data Integration	4
5	Application Integration	6
6	Data Services for Integration	7
7	Unification of Integration – The Integration Layer	9
8	The Talend Platform for Data Services	11
	About the Author Rick F. van der Lans	13
	About Talend Inc.	13

Copyright © 2012 R20/Consultancy, All Rights Reserved.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

1 Management Summary

Integration of systems and data is high on the agenda of most organizations. For new applications, for new forms of reporting and analytics, for compliancy and regulatory reasons, for improved customer support, and for numerous other reasons, IT systems have to work together. Unfortunately, in many organizations today a labyrinth of integration solutions has been developed. Many of them deploy different tools, programming languages, and technologies, ranging from using low-level programming languages to high-end ETL (extract transform load) and ESB (enterprise service bus) tools. The consequence is replication of integration specifications with no to limited re-use. This labyrinth is expensive and difficult to maintain. A need exists for unification and centralization of these integration solutions. This whitepaper describes the importance of unification and centralization of integration and explains the role Talend's open source integration solution called The Talend Platform for Data Services *can play here.*

Many organizations struggle with their plethora of applications, databases, and systems that they developed and bought over the years. The infamous *silos of applications* are a reality and are forming a constraining factor for making their IT systems ready for the coming decade. Integration of all these systems, databases, and applications is increasingly a necessity for organizations. New applications must be able to access data in older systems as if there is one integrated environment. The goal is to have one *unified view* of all the enterprise data and systems.

The goal is to have one unified view of all the enterprise data and systems.

Over the years, most IT departments, have constructed many integration solutions. In most cases, these integration solutions use different technologies, languages, and tools. Some solutions use ETL, replication, and ESB tools, whereas others are developed with programming languages, such as Java, C#, Python, or database languages, such as PL/SQL and Transact-SQL. This has resulted in a labyrinth of integration solutions in which many integration specifications are replicated and are not re-used. In other words, it has resulted in a *disintegrated integration solution*.

The disadvantages of replicating the integration specifications are numerous, including inconsistent integration specifications, inconsistent results, decreased time to market, increased development costs, and increased maintenance costs.

An environment is required that supports the different forms of integration and is capable of centralizing the integration specifications. A more integrated setup of the integration problem is needed where integration specifications are defined only once and are re-used repeatedly.

Organizations need a more centralized integration solution where integration specifications are being re-used: the integration layer.

The Talend Platform for Data Services is an open source integration platform, that supports different technological styles of integration, including ETL and ESB, and allows many integration specifications to be entered once and to be re-used repeatedly. Whether developers want to

Copyright © 2012 R20/Consultancy, All Rights Reserved.

use data integration or application integration, they all use the same development platform. The Talend Platform for Data Services helps to work towards an *integration layer* that centralizes integration specifications and unifies an organization's integration solutions.

2 Integration is a Necessity

In an ideal world, all the applications retrieve data from and manipulate data residing in one data store in which all the data is stored only once. In fact, the reason why database servers were introduced a long time ago, was to get rid of the dispersion and duplication of data in countless files. Those countless files were becoming an obstacle to the further expansion and maturation of IT systems. Back in 1977, Chris J. Date, one of the founders of the relational model, wrote the following in his best-seller *An Introduction to Database Systems*[1]:

> *An enterprise should store its operational data in an integrated database to provide the enterprise with centralized control of its operational data, … This is in sharp contrast to the situation that prevails in most enterprises today [1977], where typically each application has its own private files … so that the … data is widely dispersed.*

Most of those files are gone by now. Currently, enterprise data is predominantly stored in databases. Unfortunately, it's still distributed and duplicated, though not over files anymore, but over numerous databases. Many organizations struggle with their plethora of applications, databases, and systems that they have developed and acquired over the years. The infamous "silos of applications" remain a reality and a constraining factor for making their IT systems ready for the coming decade.

IT is becoming more and more a crucial instrument for organizations to become more agile, more customer-oriented, more efficient and effective, and so on. In a 2008 survey of over 1,500 CIO's, analyst organization Gartner[2] concludes that business leaders expect IT to make the difference rather than deliver generic solutions. IT should not be seen purely as a cost factor anymore. Peter Hinssen describes it as follows in his book The New Normal[3]: *Information is the new oil*. Information is becoming crucial for an organization to survive, compete, excel, and expand.

To steer IT into the next era, current systems, databases, and applications must be integrated to form a coordinated system.

But in most cases, to steer IT into the next era, current systems, databases, and applications, although they have been designed to function independently, they must be integrated to form a coordinated system. New applications and new modules must be able to retrieve data in older systems as if there is one integrated environment. The goal is to have one *unified view* of all the enterprise data, despite the fact that data is stored in a heterogeneous set of systems and databases. Integration of all these systems and data stores is

[1] C.J. Date, An Introduction to Database Systems, second edition, Addison-Wesley Publishing Company, 1977.
[2] Gartner, Making the Difference: The 2008 CIO Agenda, Gartner Executive Program CIO report, 2008.
[3] P. Hinssen, The New Normal, Mach Media NV, 2010.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

a necessity for enterprises in the coming years.

3 The Disintegrated Integration Solution

Throwing away all the existing applications and rebuilding them, is not an option. To get that unified view, solutions have to be developed that simulate a unified database. Most IT departments have spent large portions of their budgets on integrating systems and databases. For example, data warehouses have been developed to merge data from many data sources to support reporting and analytics, many service-oriented systems have been constructed in which layers of software are added on top of existing applications to form an integrated environment, replication technology is used to copy data from operational databases to develop fallback systems, and even on a smaller scale integration projects have taken place, such as business users who use tools, like Microsoft Excel, to integrate data from multiple systems using copy and paste. These are all examples of integration projects.

So, the current situation is that many integration solutions have been developed independently of each other. Usually, this has resulted in a *labyrinth of integration solutions*; see Figure 1. In most cases, these integration solutions use different technologies, languages, and tools. Some solutions use ETL, replication, and ESB tools, whereas others are developed with programming languages, such as Java, C#, Python, or database languages, such as PL/SQL and Transact-SQL.

The key disadvantage of this approach is that *integration specifications* are replicated and are not re-used. The definition of a source table structure is replicated in many solutions, the way data from a table has to be transformed is replicated, how the data from different systems has to be integrated is replicated, and so on. Evidently, this approach makes it hard to keep all those integration specifications consistent, and the costs of implementing and maintaining them is high. Over and over again, the wheel is reinvented.

The key disadvantage of independent integration solutions is that integration specifications are replicated and are not being re-used.

This solution can be called a *disintegrated integration solution*. What is needed, is a more integrated setup of the integration problem where integration specifications are defined only once and are re-used. Figure 2 represents the ultimate dream.

To support this need for system integration, a wide range of solutions is available that is normally divided into two categories: *data integration* and *application integration*. The first category focuses on integrating data stored in a heterogeneous set of data stores, whereas with application integration applications are integrated. In the next two sections, these two forms of integration are described in more detail.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

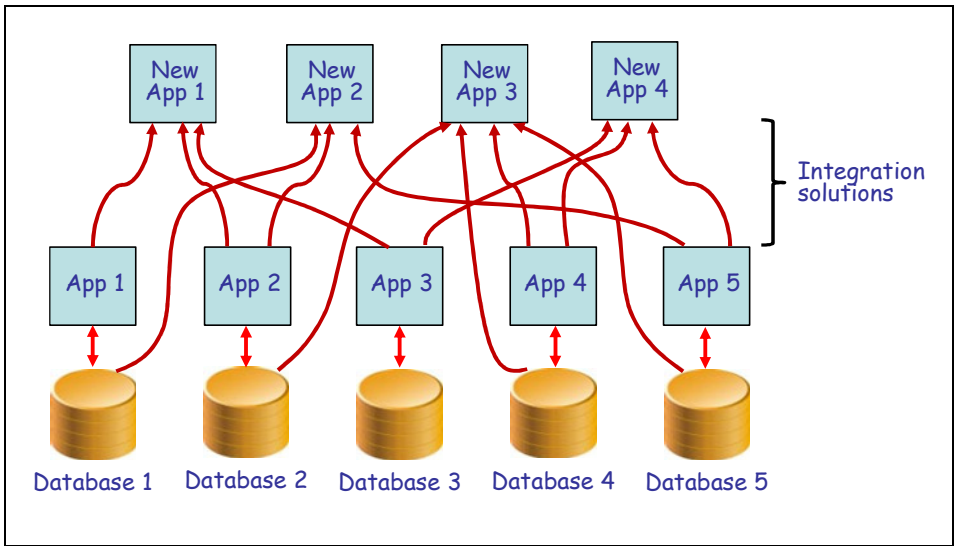

Figure 1 *A labyrinth of integration solutions is used to integrate applications and databases*

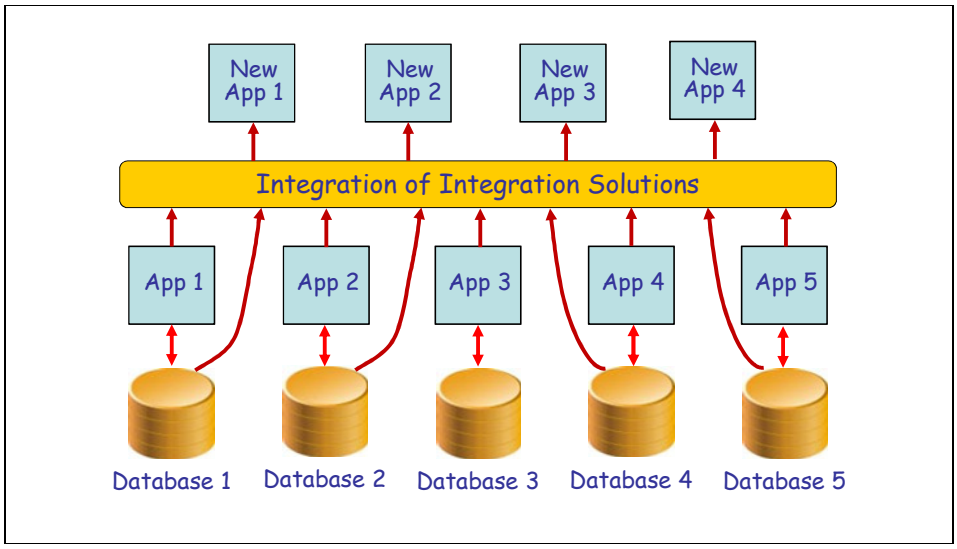

Figure 2 *An integration of the integration solutions, for example through a layer of data services*

4 Data Integration

Data integration implies that integration takes place on the data(base) level itself; see Figure 3. To integrate, data is extracted from two or more data stores and combined by the *integrating application*. Popular technologies for data integration are ETL, ELT, replication, and data virtualization. However, some integrating applications are implemented using code written in classic programming languages.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

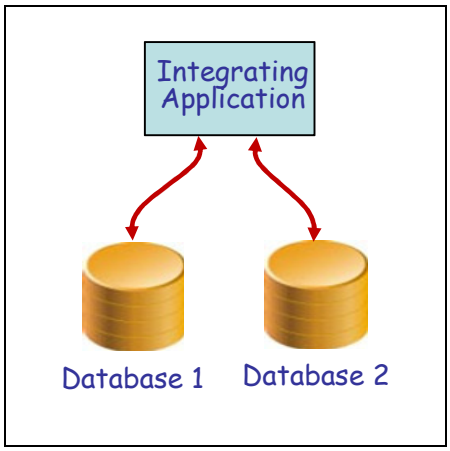

Figure 3 *Data integration implies that systems are integrated on the data level; the integrating application retrieves data from the data stores*

With data integration, the *source applications* (responsible for manipulating the data in the data stores) are bypassed. The consequence is that many important pieces of application logic implemented in those source applications are bypassed as well. Here are some examples of such pieces of application logic:

With data integration various important pieces of application logic implemented in the source applications are bypassed.

- **Data Integrity:** All the application logic that is implemented in the source applications to verify whether new data is according to all the integrity rules, is bypassed.
- **Data-Related Logic:** All the application logic that "understands" the data and that is hardcoded in the applications, is bypassed. For example, if the meaning of certain codes is hardcoded in the source applications, that meaning is not available when doing data integration. Another example is that sometimes the meaning of certain column values is dependent on the values in another column, of which the integrating application is not aware.
- **Data Security:** All the application logic related to data rules implemented in the source applications, such as which user is allowed to do what with which data elements, is also bypassed.

Despite the fact that all that application logic is bypassed, in many projects, data integration is the preferred approach. For example, in data warehouse projects this form of data integration is almost always the default solution. And for creating standby and fallback copies of operational databases, replication technology is often used, which is also a form of data integration.

Any data integration solution should at least be able to solve the following issues:

- **Data Federation:** A data integration solution should be able to seamlessly combine data residing in multiple data stores, even if a heterogeneous set of data stores is accessed.
- **Data Structure Transformation:** Data in a data store always has a particular structure. A data

Copyright © 2012 R20/Consultancy, All Rights Reserved.

integration solution should be able to transform that structure into an alternative structure. For example, it should be able to transform tables with a normalized structure into tables with a denormalized structure, or vice versa, and it should be able to transform a table with detailed data into a table containing aggregated data.

- **Data Value Transformation:** A data integration solution should be able to transform individual data values. This may be needed for standardizing values.

- **Data Cleansing:** A data integration solution should support features for correcting incorrect data, such as false, missing, and misspelled data, to increase the data quality level.

- **Interface Handling:** Data may reside in different types of data stores, requiring different APIs and database languages. A data integration solution has to support many of those APIs and languages.

In this whitepaper, all the specifications for data federation, data structure transformation, data cleansing, and so on, are called *integration specifications*. In most cases, when data integration is used, all these integration specifications, which are part of the source applications, are bypassed.

5 Application Integration

With application integration the source applications are accessed and thus not bypassed.

The main difference between data integration and application integration is that with the latter the integrity rules, data security rules, and data-related logic are *not* bypassed. When data is requested or updated, it's done by invoking the proper module within the source applications; see Figure 4. What's special about application integration is that the applications communicate directly with one another. Usually, this is technically more difficult than doing data integration.

Many technologies have been available and are available for application integration, such as DEC (distributed computing environment), message queuing, CORBA (common object request broker architecture), Java EE, MOM (message oriented middleware), and RSS (really simple syndication) feeds. For the last few years, the service oriented architecture (SOA) with the enterprise service bus (ESB) as the enabling technology, is the most popular. All these technologies allow one application to, somehow, invoke application modules in other applications, or to send data to other applications.

Although this form of integration is called application integration, in most cases it comes down to data integration as well, because the invoked application returns data. Therefore, all the same integration aspects listed in the previous section apply here as well: data structure transformation, data value transformation, data federation, data cleansing, and interface handling.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

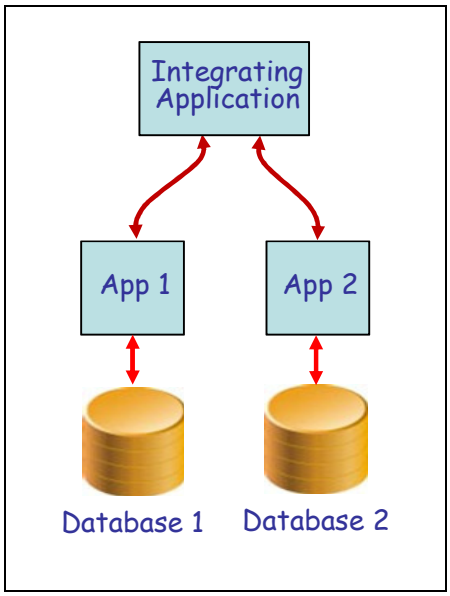

Figure 4 *Application integration implies that systems are integrated on the application level; the integrating application retrieves data via the source applications*

6 Data Services for Integration

The enterprise service bus (ESB) has become a very popular technology for integrating systems. The key building block of an ESB is a *service*. Boris Lublinsky[4] describes a service as follows:

> *Each service represents a certain amount of functionality. In the simplest case, a service can be defined as a self-contained, independently developed, deployed, managed, and maintained software implementation supporting specific business-relevant functionality for an enterprise as a whole and is "integratable" by design.*

So, a service represents some business logic. A special type of service, one that is responsible for accessing and integrating data, is called the *data service*. This section focuses on data services, because it's a popular integration solution.

A data service is a service responsible for querying and manipulating data.

As their name suggests, data services are used for querying and manipulating data. In most cases, data services access data stores directly and bypass the source applications entirely; see Figure 5. Besides retrieving data from a data store, a data service is also responsible for cleansing and transforming the values to a result that the consumers expect and need. It also assigns a structure to the result of the service.

Data services reside on the boundary between data integration and application integration. The technology used for developing data services is based on application integration, however, they are considered a form of data integration, because they access the data stores directly.

[4] B. Lublinsky, 'Is REST the Future for SOA?,' *www.infoq.com*, August 11, 2011.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

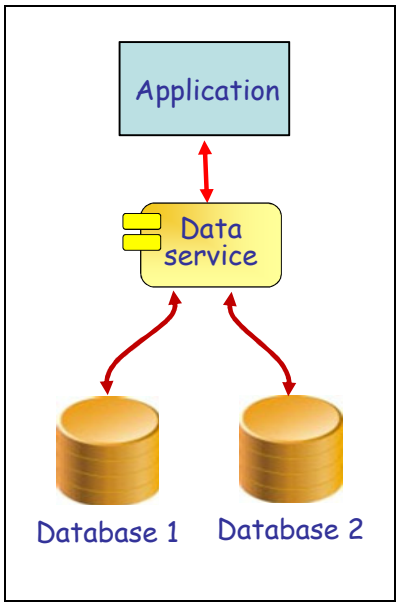

Figure 5 *Data services access data stores directly*

Roughly speaking, data services can be developed in two ways. The most obvious approach is by developing custom-code that performs all the above tasks. For example, a data service can be developed in Java or C# as a component that accesses a SQL database. The drawback of this approach is that most of the integration specifications have to be coded by hand. Some of it, such as parsing the incoming XML document, can be handled by calling functions offered by the languages, and some can be implemented by ESB functionality. Still, no task is done automatically.

Alternatively, a data service can also be developed using a more meta data-driven tool. In this case, data services are developed by entering specifications that define what the data service should do, and not so much how that work has to be done. The advantages of using meta data-driven tools are:

- **Increased Productivity:** Developing a data service with a meta data-driven tool requires a lot less development time than when a custom-made solution is developed. In this case, developers of data services only have to focus on defining the integration specifications. It's a more declarative approach. Developers don't have to deal with many of the technical aspects; compare this to the developers using programming languages who have to spend quite some time on such details. To spend less time on technical details and to focus only on the required integration specifications, improves productivity.

- **Improved Maintenance:** Because there is less code, maintenance is less complex. This feature is not supported when services are hand-coded.

- **Interface Independency:** If code is written by hand, probably only one technical interface is implemented. Supporting an extra interface requires a serious development effort. With a meta data-driven tool, allowing applications to invoke a service using other technical interfaces is purely a matter of publishing another API for that service. In other words,

Copyright © 2012 R20/Consultancy, All Rights Reserved.

the functionality of what a service should do, is developed independently of the technical interfaces for invoking it.

- **Security Independency:** All the specifications related to security are specified in the meta-data-driven tool. They can easily be changed without having to change the service itself.

7 Unification of Integration – The Integration Layer

As indicated, too often organizations deploy multiple solutions and tools for integrating their applications and databases. For example, ETL tools are used for data integration in data warehouse projects, ESB data services are used for integrating applications in real-time, database triggers are used to copy data between databases, and replication is used for creating fallback environments.

The consequence of deploying multiple solutions is that particular integration specifications are replicated.

The consequence of deploying multiple solutions is that particular integration specifications, such as data value transformation, data cleansing operations, data structure transformations, and data federation specifications, are replicated. Also, in each solution, these specifications are re-developed in another language; a specific data value transformation implemented in an ETL tool can't be re-used in a self-made Java program.

Numerous disadvantages exist of replicating integration specifications:

- **Inconsistent Integration Specifications:** Because the integration specifications are distributed over many integration solutions, it's difficult to guarantee that rules in different solutions that integrate the same data, are implemented the same way.

- **Inconsistent Results:** If different sets of integration specifications are applied, the results from different integration solutions may be inconsistent. In addition, this inconsistency will reduce the trust in data and the supporting systems.

- **Decreased Time to Market:** Because the integration specifications are replicated, changing them enterprise-wide in all relevant solutions, is time-consuming. This slows down the implementation and thus the time-to-market of new systems.

- **Increased Development Costs:** When the same systems are integrated by different integrating solutions, the same integration specifications have to be implemented multiple times. This increases development costs.

- **Increased Maintenance Costs:** Changing integration specifications in multiple different solutions implies changing them in many different tools and programming languages and requires different development skills. This raises the costs for changing integration specifications considerably.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

What is desperately needed is the *unification* of all the integration solutions and a *centralization* of all the integration specifications (see Figure 6), especially if we consider that the amount of integration specifications is abundant, the need to rapidly change those specifications is considerable, and that IT departments need to be able to integrate systems more quickly to support new business users' needs. It's important that integration becomes a separate layer or module in every IT architecture. Technically, such an integration layer can be implemented in different ways, for example using a *data services layer* or some kind of *information bus*.

Unification of all the integration solutions and centralization of all the integration specifications is needed.

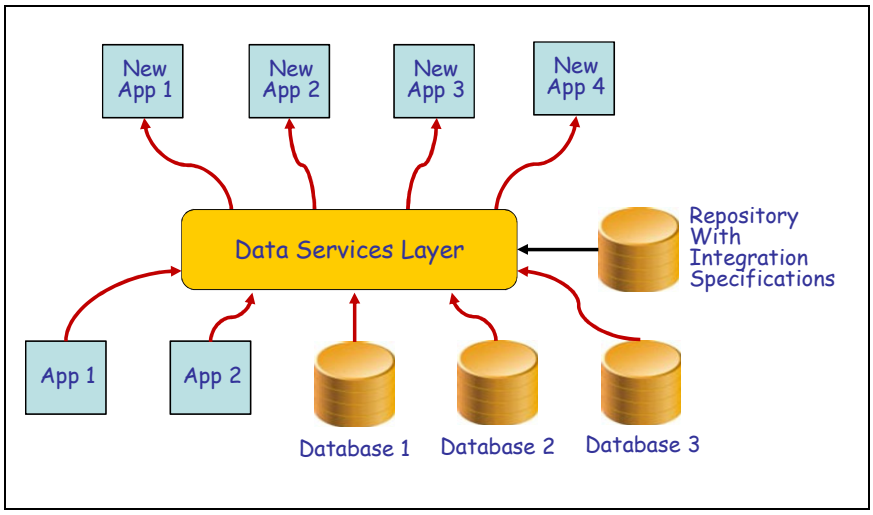

Figure 6 *In this diagram, as an example, a data services layer is used to unify all the integration solutions, a repository is used to store all the integration specifications*

Tools are needed with which these integration specifications can be introduced once, and re-used in many different solutions with the least amount of replication.

Using just one integration tool for all the integration work is not a realistic approach. For example, using data integration tools for every form of integration, or ESBs for every integration challenge, is not realistic, because most of these tools are developed and optimized for just one type of integration. Data integration excels at copying and integrating data periodically in a batch-oriented style, which is not good if the integration has to be done real-time. Likewise, ESBs are good at integrating data real-time, but if a large batch of thousands of records has to be merged with an existing database, data integration is the preferred solution.

A solution is needed that supports different forms of integration and is still capable of centralizing the integration specifications.

A solution is needed that supports different forms of integration and is still capable of centralizing the integration specifications. In this case, data virtualization tools are an option, but other solutions exist as well. Figure 7 represents such a type of product. Using the same set of integration specifications, different solutions can be generated, thus, re-using integration specifications. *The Talend Platform for Data Services* is such a solution and is described in the next section.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

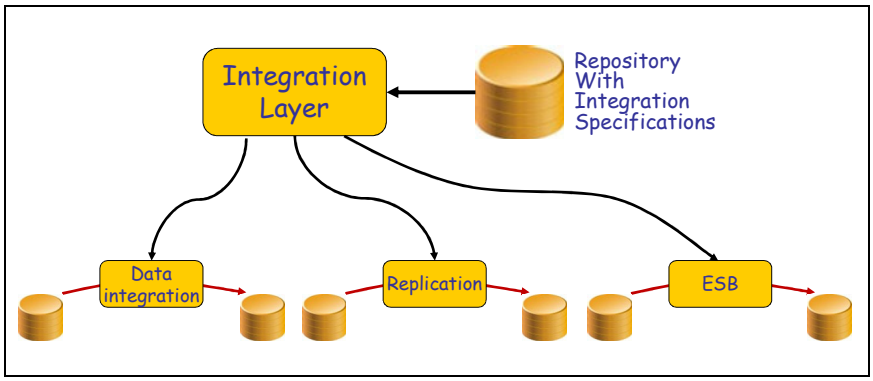

Figure 7 *An integration layer, such as a data services layer, should be capable of generating or deploying different integration solutions (black arrows pointing down) from the same set of integration specifications*

Note: Besides unifying the integration solutions, it's recommended to put integration explicitly on the agenda of each IT department as well, and maybe even introduce it as a separate departmental item: *the integration department.* Integration is so important that it justifies a dedicated department, just like there is a database department, a test department, and a network department. The integration department would be responsible for all aspects related to the integration of systems and data.

8 The Talend Platform for Data Services

Talend Inc. was founded in 2005. They were the first open source vendor of data integration software. In November 2006 they released their first product, the ETL tool called *Talend Open Studio*. In November 2011, Gartner rated Talend a visionary in their well-known magic quadrant for data integration tools.

On November 10, 2010, Talend acquired Sopera and with that they got access to a successful, high-end, open source ESB product for application integration. With this, Talend had the products, the know-how, and technology in the two main integration areas: data integration and application integration.

Since the acquisition, they have worked hard to unify the two integration solutions. The result is the solution called *The Talend Platform for Data Services* which fully supports data integration and application integration. This approach seriously minimizes the proliferation of integration specifications. It makes the goal of a unified view real. Developers trained in data integration solutions can now re-use their skills when switching to other types of integration solutions. The product can be categorized as an integration layer.

Some of the unique features of The Talend Platform for Data Services and Talend Open Studio are listed here:

- Designers and developers can use one and the same development environment, called *Talend Open Studio*, to enter and maintain integration specifications and develop solutions. This module is based on the popular Eclipse extensible integrated development environment.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

- The Talend Platform for Data Services supports many modules, including data integration data quality, master data management, ESB, and business process management. Customers don't have to acquire the entire module set. Based on their own needs, they can pick and choose. Later on they can plug in other modules.

- Integration specifications, such as connectors to data sources and schema definitions, need to be defined only once, and can be shared by different integration solutions; as represented in Figure 7.

- Talend Open Studio supports all the well-known styles of integration, including synchronization of databases, replication of data between databases, ETL, ELT, and data migration.

- The business modeler allows for IT specialists in cooperation with business analysts to define specifications. The business models are non-technical and business-oriented. This allows for supporting the data integration process using business-friendly diagrams.

- Integration specifications are defined using a high-level graphical language improving productivity and maintenance.

- All the meta data (integration specifications) is stored in one integrated repository managed by the metadata manager.

- The Talend Platform for Data Services supports a wide range of connectors to extract data from and import data into data stores, such as SQL databases, files, SAP, Salesforce, web services, and XML documents.

- Dependency analysis is supported to determine dependencies between integration modules which eases development and maintenance.

- The Talend Platform for Data Services is also a part of other high-level modules delivered in The Talend Unified Platform that provides capabilities that focus on the quality of the data, such as master data management, data profiling, and data quality checks.

To summarize, given its features, The Talend Platform for Data Services can act as an integration layer. Designers and developers work with one and the same development environment for, for example, developing ETL-based integration solutions, and service-based solutions. One environment for storing all the integration specifications, and one environment for deploying these specifications in multiple systems.

The Talend Platform for Data Services supports unification of integration solutions, allowing integration specifications to be re-used, and can be used for developing an integration layer.

Copyright © 2012 R20/Consultancy, All Rights Reserved.

About the Author Rick F. van der Lans

Rick F. van der Lans is an independent analyst, consultant, author, and lecturer specializing in data warehousing, business intelligence, service oriented architectures, and database technology. He works for R20/Consultancy (www.r20.nl), a consultancy company he founded in 1987.

Rick is chairman of the annual European Data Warehouse and Business Intelligence Conference (organized in London) and chairman of the annual BI event[5] in The Netherlands. He writes for B-eye-Network.com[6]. He introduced the business intelligence architecture called the *Data Delivery Platform* in 2009 in a number of articles[7] all published at BeyeNetwork.com.

He has written several books on SQL. His popular *Introduction to SQL*[8], published in 1987, was the first English book on the market devoted entirely to SQL. After more than twenty five years, this book is still being sold and has been translated in several languages, including Chinese, German, and Italian.

The last years he has focused on applying data virtualization in business intelligence system resulting in his upcoming book entitled *Data Virtualization for Business Intelligence Systems* which will be released in the summer of 2012.

For more information please visit www.r20.nl, or email to rick@r20.nl. You can also get in touch with him via LinkedIn (http://www.linkedin.com/pub/rick-van-der-lans/9/207/223) or via Twitter (http://twitter.com/Rick_vanderlans).

About Talend Inc.

Talend is one of the largest pure play vendors of open source software, offering a breadth of middleware solutions that address both data management and application integration needs.

Since the emergence of data integration and data quality tools in the 1990s, and the more recent appearance of Master Data Management solutions, the data management market has been dominated by a small—and quickly consolidating—number of traditional vendors offering proprietary, closed solutions, which only the largest and wealthiest organizations can afford. The situation in the application integration space is quite similar, with significant consolidation occurring as well. As a result, only a minority of organizations use commercial solutions to meet their data management and application integration needs. Indeed, these solutions not only demand a steep initial investment, but they also often require significant resources to manage implementation and ongoing operation.

---

[5] See http://www.bi-event.nl/59857
[6] See http://www.b-eye-network.com/channels/5087/articles/
[7] See http://www.b-eye-network.com/channels/5087/view/12495
[8] See http://www.amazon.com/Introduction-SQL-Mastering-Relational-Database/dp/0321305965/ref=sr_1_1?ie=UTF8&s=books&qid=1268730173&sr=8-1

Copyright © 2012 R20/Consultancy, All Rights Reserved.

Furthermore, companies are faced with exponential growth in the volume and heterogeneity of the data and applications they need to manage and control. A key challenge that IT departments face today is ensuring the consistency of their data and processes by using modeling tools, workflow management and storage, the foundations of data governance in any company today. This challenge is actually faced by organizations of all sizes - not only the largest corporations.

In just a few years, Talend has become the recognized market leader in open source data management. The acquisition in 2010 of Sopera, a leader in open source application integration, has reinforced Talend's market coverage, creating a global leader in open source middleware. Many large organizations around the globe use Talend's products and services to optimize the costs of data integration, data quality, Master Data Management (MDM) and application integration. With an ever growing number of product downloads and paying customers, Talend offers the most widely used and deployed data management solutions in the world.

Talend offers a completely new vision, reflected in the way it utilizes technology, as well as in its business model. The company shatters the traditional proprietary model by supplying open, innovative and powerful software solutions with the flexibility to meet the data management and application integration needs of all types of organizations.

Copyright © 2012 R20/Consultancy, All Rights Reserved.