

White Paper

Fundamentals of Distributed Data Architecture

Adrian Miley

Synopsis

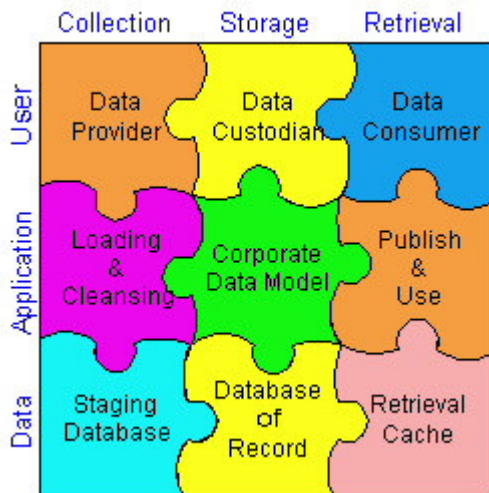
Constructing a high-level Distributed Data Architecture is like solving a jigsaw which can sometimes seem very complex with hundreds of pieces that need to be fitted together to form a seamless whole. This article describes an approach that reduces the significant components to nine main parts as an introduction to the background of a Distributed Data Architecture Framework.

This document contains information proprietary to Miley Watts & Associates Ltd, and may not be reproduced, disclosed or used in whole or part without express written permission of Miley Watts & Associates Ltd.

If you would like to discuss any of the ideas described in this document in more detail then please contact us at enquiries@mileywatts.com and we will be happy to have a conversation.

Constructing a high-level Distributed Data Architecture is like solving a jigsaw which can sometimes seem very complex with hundreds of pieces that need to be fitted together to form a seamless whole.

However luckily enough even the most complex data environment can be reduced down to nine main components that fit together something like this:



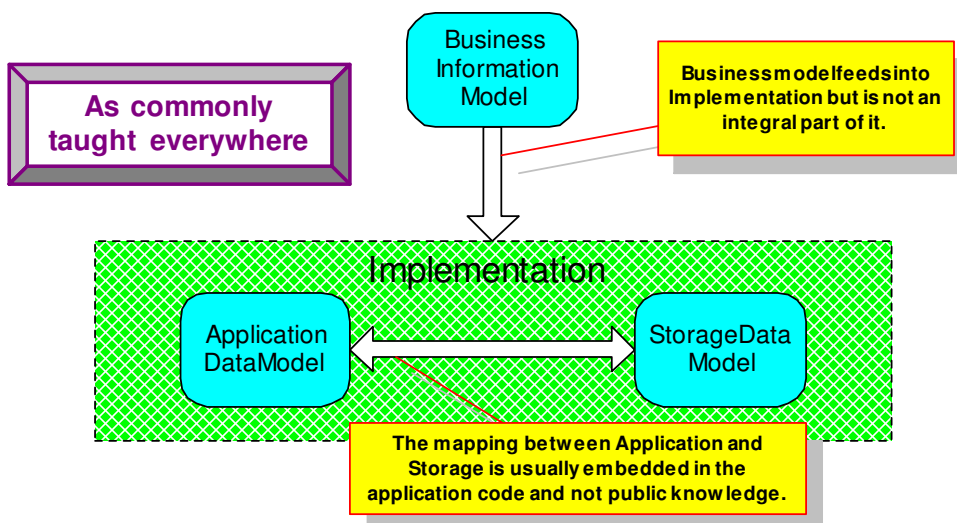
The stacks are the main stages in the life of the data from Collection (initial creation and modification) to Storage (retention and management) to Retrieval.

The layers are the three main components of virtually all data processing applications which are Users Interfaces (the public entry-points into the data domain), the Application layer (the software that defines, applies and enforces the business processes being supported) and the Storage layer where the data may be persistently stored when an Application is not operating on it.

Each piece of the jigsaw should be self-contained and, as with most framework architectures, the physical details of each component (physical location, technology platform, etc) is largely irrelevant - it is establishing the purpose of each piece and the principles and definition of what happens at the boundaries that is important.

The fundamental thing, and hence the start point to solving the jigsaw, is the data content itself and the four views that an organisation has of this data - we call them the Content Forms.

Traditionally most of the academic views, originally developed to describe smaller single-purpose applications, take a three model view consisting of a conceptual or logical business model (the Corporate Data Model), an application view and a database storage view linked together as follows:

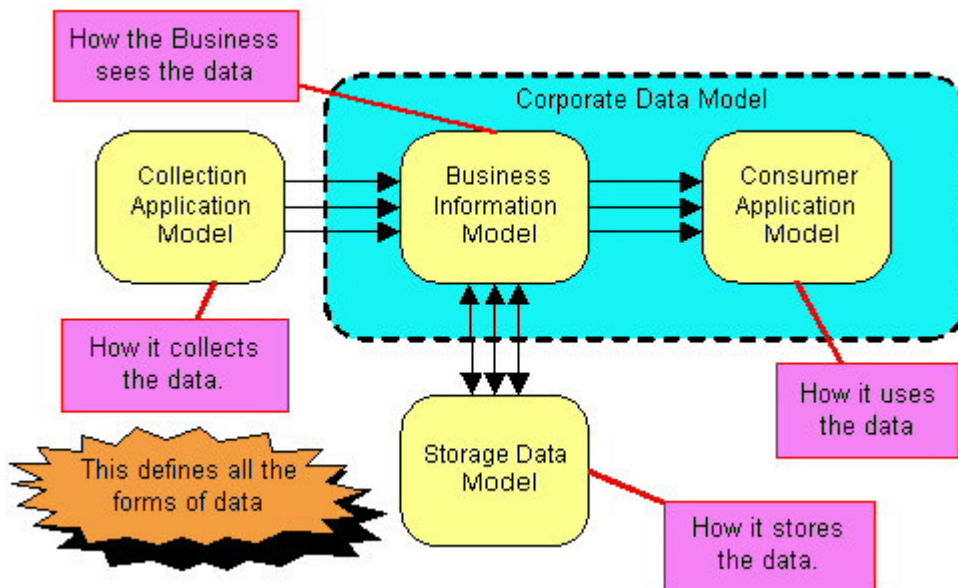


However within large -scale, data-centric organisations the applications that collect or create data and the applications that retrieve or consume data are generally separate, and may be radically different, applications with different behaviour characteristics.

Separating Collection and Retrieval simplifies the overall data architecture in the following ways:

- Only the Collection Data Models need detailed knowledge of the business rules that are to be supported to enforce data consistency.
- Retrieval models can assume that the data they receive is valid so, because they do not have to implement the validation rules, they are much simpler to define. Generally there are far more applications that consume a piece of information than there are applications that create it. Therefore simplifying the rules for defining consuming application models should significantly reduce "time to market" for new consumer applications.

This gives us a 4-model view that looks something like this:



The purpose of each model is:

- The Corporate Data Model (CDM) is the business oriented canonical view of the entire integrated data domain expressing all the information using the business semantics and providing a single, consistent definition and validation rules for all the data that exists within the data domain.
- The Storage Model describes each of the operational databases (relational, object or hierarchical databases) where the operational data is persistently stored and managed.
- The Collection Application Model describes each of the data collection applications that populate and feed information into the Corporate Data Model. In the case where the data is sourced from an external 3rd party data source then the Collection Model will be provided by that external supplier.

- The Retrieval Data Model describes each distinct application that consumes the data and presents it to downstream clients in effect it describes how an organisation uses the data that it collects.

These Content Forms gives us the initial four pieces of the jigsaw i.e. the three pieces of the Application Layer plus the Storage piece.

This however this assumes that only fully defined business data is actually stored and that all applications only interact with the Database of Record for all their data requirements (though there may be multiple data-stores holding that data) but in a large scale data environment this is unlikely to be the case.

On the collection side there may also be pre-production data-stores where data may be temporarily stored whilst it is being cleansed or awaiting integration with other data feeds or archived for auditing or data analysis purposes.

On the retrieval side there may also be multiple data caches defined for storing sub-sets and snapshots of the operational data specifically to support a particular Retrieval Application. These could take many forms including in-memory databases or pre-generated XML documents created for performance purposes.

These additional two pieces are optional but need to be considered in the overall architecture in order to ensure that "straight through" data-flow requirements are logically supportable.

The final parts of the jigsaw are of course the user interfaces that sit on top of the application layer. These can take many forms including interactive screens, batch data-feeds, broadcast events, on-demand services and any other method of either putting data into or taking data out of the business domain.

Generally this layer is only vaguely of interest within the high-level data architecture itself as it is mainly concerned with the "user experience" i.e. presentation and rendering, and the main reason that it needs to be considered is because of the extensive "metadata" usage that normally exists within the User Interfaces.

So that essentially gives us the basic 9-piece framework of the Distributed Data Architecture jigsaw. It is merely the beginning of the process and its purpose, like all frameworks, is simply to help us break down the problem into simpler parts in order to make it both solvable and manageable.

So, when faced with designing a new data environment or analysing an existing one the initial two tasks are then:

- Identify the components, applications and data-stores that exist and which part of the framework they should reside in.
- Identify the rules and principles that need to be enforced at the boundaries to ensure transparent inter-operability and consistent data-flow between pieces. This is critical because once established it is policing the boundaries that should form the nucleus of the IT Governance activity *not* auditing the activities within a component.

In applying the framework to establish a high-level Enterprise Data Architecture it is important to note that the actual technology should neither be a concern nor an issue - it should be irrelevant whether DB2, Oracle or SQL Server underpins a particular data-store or whether an application is implemented in Java or .NET or whether transaction based

applications use service-oriented, events-oriented or good old-fashioned peer-to-peer client-server principles to interact with the operational data and so on.

These technology choices are important decisions to be made but are unimportant in describing the target architecture to the business stakeholders.

In applying this framework we also draw in many aspects of other architectures e.g. once the Corporate Data Model is in place then:

- A Model Driven Approach could then be taken to producing the service interfaces required by the downstream Retrieval Applications because they should be directly derivable from the Corporate Data Model using pre-defined transformation and generation rules.
- A Data Quality & Governance framework can be implemented across multiple storage platforms through enforcing a consistent definition of each data-item.
- An Event Driven Architecture could be created for the data collection components where all of the data processing rules necessary to carry out an operation generated by an event can be directly derived from the Corporate Data Model.

That essentially is it - a Distributed Data Architecture Framework in nine pieces.