1. Introduction
   1. Simple, Agile, Data Warehousing

The Data Integration Framework is a complete and proven solution for data management using a fast, consistent, robust and easy to use agile approach.

The ETL Framework provides a software and methodology independent, structured approach to data integration. The framework is designed to facilitate a flexible and affordable development cycle. By fit for purpose pre-defined documents, templates, design decisions, built-in error handling, auditability and process control the framework provides the consistency and structure for future-proof data integration and Data Warehouse design and development on any platform.

It is not a one-size-fits all solution; everything is defined in a modular way and different elements can be applied to suit the needs of each individual project. The ETL Framework has a variety of content components which can be used in conjunction with each other or as stand-alone addition to existing management information solutions.

The fundamental principle of the ETL Framework is to design for change by decoupling 'warehouse logic' and 'business logic' and ensuring every ETL process can run and recover at any point in time (parallel) without impacting dependencies to other processes. The framework provides a standard to manage all data as an *asset* for the organisation and ensures that any new or changed requirements in reporting and information delivery can be met without re-engineering the underlying foundations. By decoupling the business logic it is possible to add new sources of information in a straightforward and consistent way without impacting existing reporting or analysis initiatives

.

The ETL Framework does not break with established approaches and schools of thought but has defined the optimal combination of well-known and understood techniques to simplify data and information delivery.

* 1. Why need an ETL Framework?

‘If we want better performance we can buy better hardware, unfortunately we cannot buy a more maintainable or more reliable system’.

The design and implementation of the ETL data flow is largely a labour-intensive activity and typically consumes large fractions of the effort in Data Warehouse and data integration projects. Over time, when requirements change and enterprises become more data-driven, the BI architecture faces challenges in the complexity, consistency and flexibility in the design (and maintenance) of the data integration flows.

These changes can include changes in latency, the bigger variety of sources or the introduction of more parallel processing to a previously rigid serial pipeline. All of this occurs when Data Warehousing and BI become more and more mission critical and its information is integrated into the operational decision making process.

Using a flexible ETL approach will meet these challenges by providing structure, flexibility and scalability for the design of data integration flows.

Today’s BI architecture is typically designed to store structured data for strategic decision making where a small number of (expert) users analyse (historical) data and reports. Data is typically periodically extracted, cleansed, integrated and transformed in a Data Warehouse from a heterogeneous set of sources. The focus for ETL has been on ‘correct functionality’ and ‘adequate performance’ but this focus misses key elements that are equally important for success. These elements, such as the consistency, degree of atomicity, ability to rerun, scalability and robustness are addressed by using the ETL Framework.

Future data solutions should for example be able to cater for sending back cleansed or interpreted data to the operational systems. They also should be able to cope with unstructured data next to the structured data and must be able to quickly respond to changes in (business) requirements. Lastly, it will need to support a ‘feedback loop’ to incorporate changes made by (authorised) end-users in the front-end environments.

To be ready for future changes the next generation data integration and ETL designs must support a methodology which provides the foundation for a flexible approach. Without this structured approach to data integration design the solution will ultimately risk becoming the ‘spaghetti of code and rules’ that it was initially meant to replace. That is why we need an ETL Framework.

The ETL Framework provides a structured approach to data integration design for an easy, flexible and affordable development cycle. By providing architecture documents and mapping templates, design decisions and built-in error handling and process control the ETL Framework provides the consistency and structure for future-proof ETL on any platform.

* 1. Key benefits
* Respond quickly to changing business requirements; the industry standard layered architecture and its related defined ETL solution (taxonomy) allow agility when it comes to information delivery
* Limited or no need to re-engineer; by decoupling data management (warehouse) and business (transformation) logic a solid foundation to manage all data within the enterprise is established. Changes or additions can be applied without the need to 'go back' and change the underlying foundations and technical improvements can be executed without impacting the reporting environments
* Reduction of time to value; defined and proven ETL patterns, an out of the box process metadata model and ETL automation support drastically reduce development times
* Built-in handling of (information) complexities, integration and dependencies; pre-defined solutions for error handling, parallelism, reconciliation, recovery and handling of business rules provide a data platform that requires no re-engineering
* High level of maintainability and support; built-in resilience in ETL, archiving, combined with a maintenance Graphical User Interface (GUI) and a strict set of conventions remove maintenance complexities often associated with Data Warehousing
* Model driven design; define the information model, and expand your solution gradually and consistently from there. ETL is automatically generated using the model specifications
* ETL quality and consistency; template driven ETL automation based on a conceptual framework provides a repeatable and dynamic development process which reduces the need for extensive documentation and delivers deterministic and high quality ETL logic
* A documented and sound foundation for the Data Warehouse; the highly structure and complete documentation of all framework components provide a full picture from the high level concepts all the way down to the technical implementation for a large variety of ETL platforms
* The ETL Framework provides the rules; only the focus on the necessary data (input) and the reporting (output) is required

* 1. Intent and foundational principles

To accurately and quickly adapt to business needs the intended data solution should decouple ‘warehouse logic’ from ‘business logic’. The basic assumption is that requirements will change over time, and that any solution that specifically is designed for a certain output or requirement will fail over time when adjustments are made. Rather, the framework views requirements from a data perspective and aims to properly integrate and consolidate data before applying a certain view.

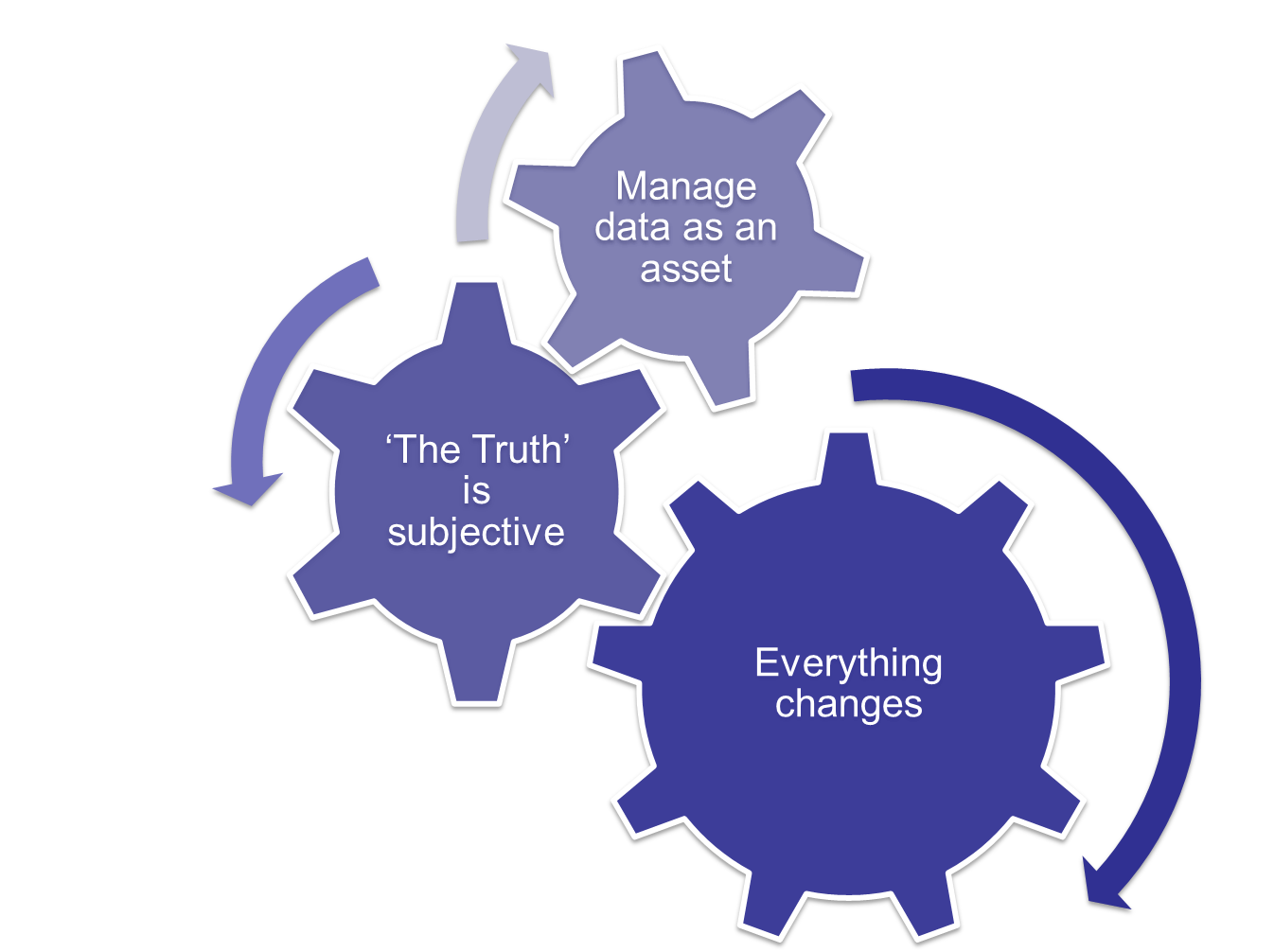


Figure 1: Underlying principles

To this intent, the framework aims to establish a ‘corporate memory’ of data to which new components can be added on a per-project basis. Output can be re-deployed with different views when requirements change.

1. ETL Framework overview
   1. Components

The complete deployment of all components supports an industry standard flexible and agile Data Warehouse application at a minimum of cost and maximum flexibility and ease of use.

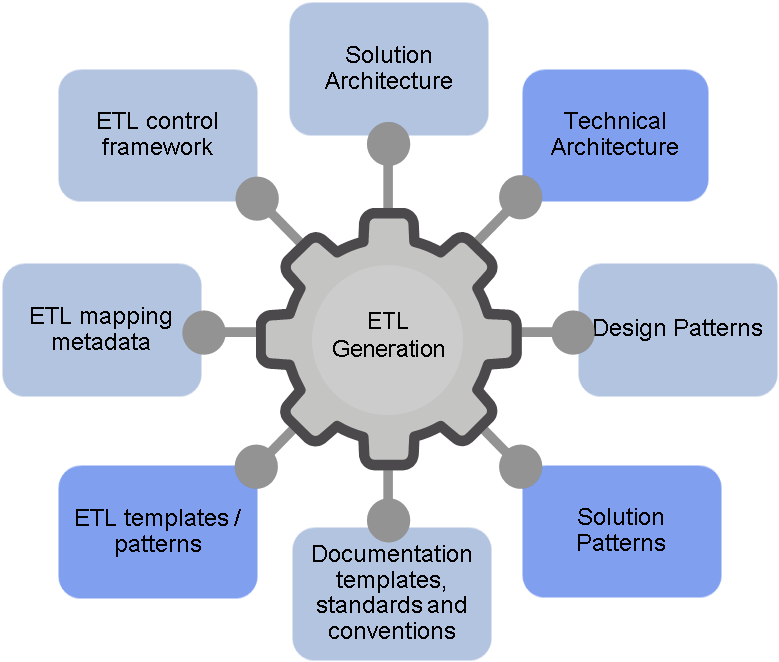


Figure 2: ETL Framework components

* **Reference Solution Architecture**; the default Data Warehouse / Information Hub architecture
* **Reference Technical Architecture**; the most common technical requirements relevant to the solution architecture
* **Design Patterns**; documentation templates for key design decisions. Design Patterns are centrally stored and managed and are publicly available in Confluence
* **Solution Patterns**; technical implementation for key design decisions (design patterns). Implementation Patterns are also publicly available in Confluence
* **Documentation templates, standards and conventions**; modelling and technical conventions
* **ETL Templates**; detailed explanation of the taxonomy of ETL processes
* **ETL mapping metadata**; a method for managing ETL metadata (source to target mappings)
* **Operational Meta Data** (OMD); ETL process control, recovery and traceability. The OMD includes a repository for ETL control and a Graphical User Interface

* 1. ETL Framework documentation breakdown

The ETL Framework consists of the following documents:

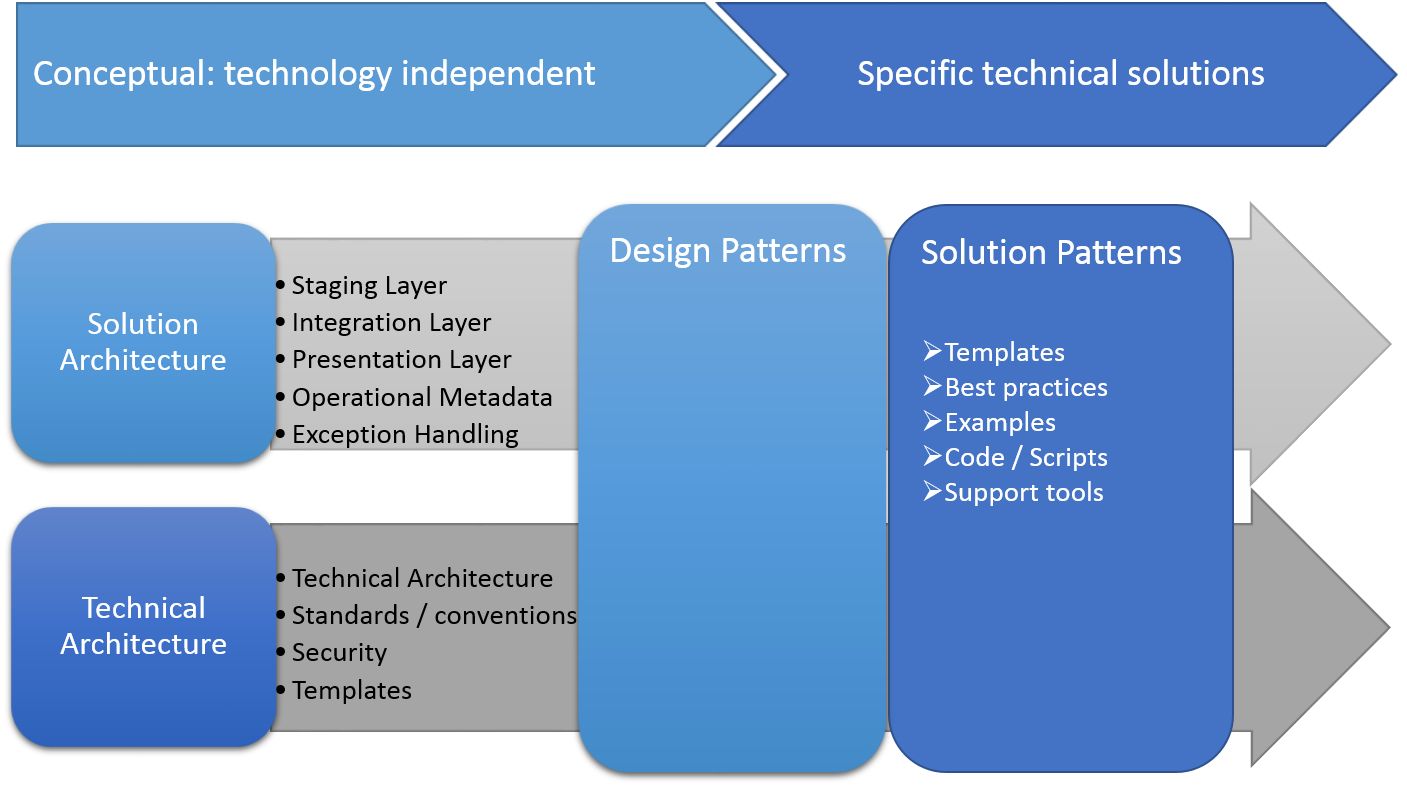


Figure 3: Documentation breakdown

A full overview is provided below:

* The (reference) **Solution Architecture** documentation is composed of the following documents:
  + ETL Framework – 1 – Overview. The current document, providing an overview of ETL Framework components.
  + ETL Framework – 2 – Reference Architecture. The reference architecture describes the elements that comprise the (enterprise) Data Warehouse and Business Intelligence foundations, with the details showing how these elements fit together. It also provides the principles and guidelines to enable the design and development of Business Intelligence applications together with a Data Warehouse foundation that is scaleable, maintainable and flexible to meet business needs. These high level designs and principles greatly influence and direct the technical implementation and components
  + ETL Framework – 3 – Staging Layer. This document covers the specific requirements and design of the Staging Layer. The document specifies how to set up a Staging Area and History Area
  + ETL Framework – 4 – Integration Layer. This document covers the specific requirements and design of the Integration Layer; the core Enterprise Data Warehouse
  + ETL Framework – 5 – Presentation Layer. This document covers the specific requirements and design of the Data Marts in the Presentation Layer which supports the Business Intelligence front-end.
  + ETL Framework – 6 – Metadata Model. This document covers the complete process of controlling the system, which ties in with every step in the architecture. All ETL processes make use of the metadata and this document provides the overview of the entire concept. The model can be deployed as a separate module
  + ETL Framework – 7 – Error handling and recycling. This document covers the complete error and recycling process, which ties in with every step in the architecture. Elements of the error handling and recycling documentation can be used in a variety of situations
  + ETL Framework – 8 – OMD Framework Detailed Design. This document provides detailed process descriptions for the ETL process control (Operational Meta Data model – OMD).
* **Design Patterns**. Detailed backgrounds on design principles: the how-to’s. Design Patterns provide best-practice approaches to typical Data Warehouse challenges. At the same time the Design Patterns provide a template to document future design decisions.
* **Solution Patterns**. Highly detailed implementation documentation for specific software platforms. Typically a single Design Patterns is referred to by multiple Solution Patterns, all of which document how to exactly implement the concept using a specific technology
* The (reference) **Technical Architecture** contains the following elements:
  + (Project) Technical Architecture. The (situation specific) solution architecture document describes the physical implementation of the architecture in detail. This document includes:
    - General ETL director setup or configuration;
    - Database schemas and physical data model;
    - Adopted modelling techniques;
    - Considerations and reasoning;
    - Selected interface types;
    - Every generic technical decision made;
  + Database and Security configuration. A repository for the database setup in terms of security, roles and privileges.
  + Naming Conventions and standards usable for data modelling and ETL implementation are stored in Confluence
  + ETL templates and approach used

The Reference Architecture and the corresponding Technical (Solution) Architecture provide common ground for all design and development activities. Design Patterns relate to (concepts of) the architecture and provide the details of the concepts including considerations and pros and cons. In other words: the Reference Architecture positions the concepts and the Design Patterns detail how the concepts should function.

For instance, the Reference Architecture (Staging Layer component) states that the loading of Flat Files should be broken in different process steps where data type conversions must be performed. It also states that Flat Files should be archived after processing, and why.

In this example the Design Pattern would refer to the ‘AGA ETL Framework - 2 - Staging Layer’ document and related Solution Patterns to define the necessary elements like storing the file creation date, unzipping and moving files, creating file lists and other necessary steps.

1. Adoption
   1. Positioning

The ETL Framework should be viewed as one part of the larger (enterprise) architecture. The purpose is to specify how the ETL and the data model can be configured for an optimal Enterprise Data Warehouse implementation. This is a detailed (albeit significant) component in the Data Warehouse architecture which in itself includes other components such as system landscape, subject areas and the Business Intelligence and Data domain.

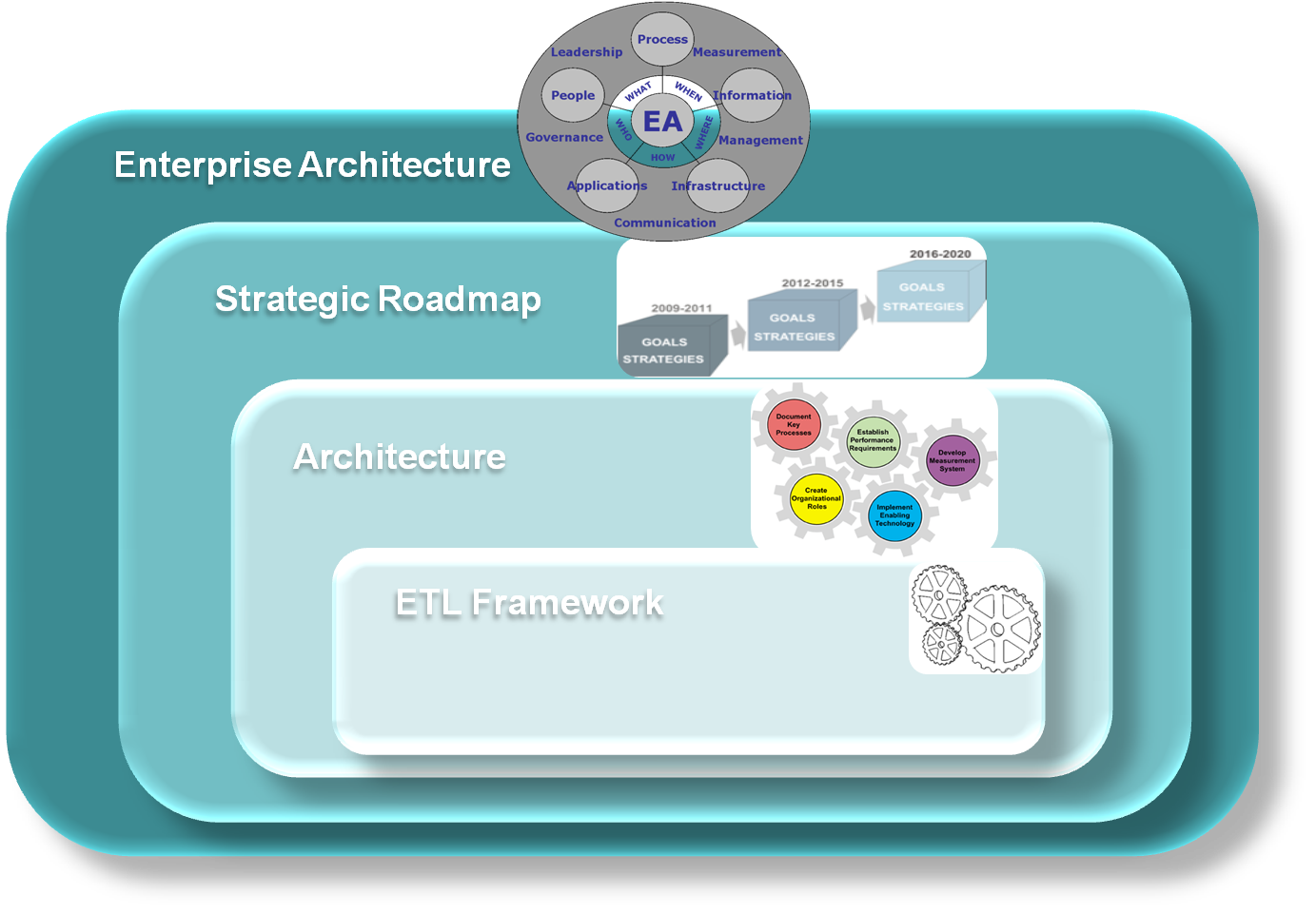


Figure 4: Positioning of the ETL Framework

* 1. Using the ETL Framework

The reference architecture serves as an outline to relate ETL examples and best-practices to. The main purpose is to create a common ground where every developer can use the same approach and background to contribute to a common integrated data repository.

Because of its nature as reference architecture not all components necessarily have to be deployed for an individual project. In some scenarios components are integrated in existing solutions or structures. For this reason the solutions will be as designed to be as modular as possible thus enabling the utilisation of specific components.

The framework Reference Architecture is a standard approach for Enterprise Data Warehouse and Business Intelligence design and implementation. The most important aspect is to understand which ETL and Data Warehousing concepts are used in what Layer of the architecture and the reasoning behind this.

The Reference Architecture also provides the basic structure for the documentation of the ETL Framework. Every component of the design and implementation relates back to this architecture, including tips, tricks and examples of implementation options. The implementation solutions for this architecture are designed to be as generic as possible without losing practical value.

* 1. Executing a project

In principle every project that contributes to the common integrated data model as executed within the AGA ETL Framework follows the same approach. At a high level this is as follows:

* Define Solution Architecture and Technical Architecture based on the framework reference architecture. Effectively this defines how sources are interfaced and integrated into the central model. For instance how to collect data delta (CDC) following the framework principles, where data should be integrated in the structured or unstructured world etc.
* Define Project Scope; this is a breakdown of a requirement in data terms. What data is needed to meet requirements in the broader sense (answer this and similar questions). This scope of data becomes the input for bottom-up planning of data integration
* Execute Model Driven Design. Once the data is identified both in scope and way of interfacing the metadata can be entered to drive ETL generation and (automated) testing. Model changes can be made accordingly, after which the ETL can be re-generated.

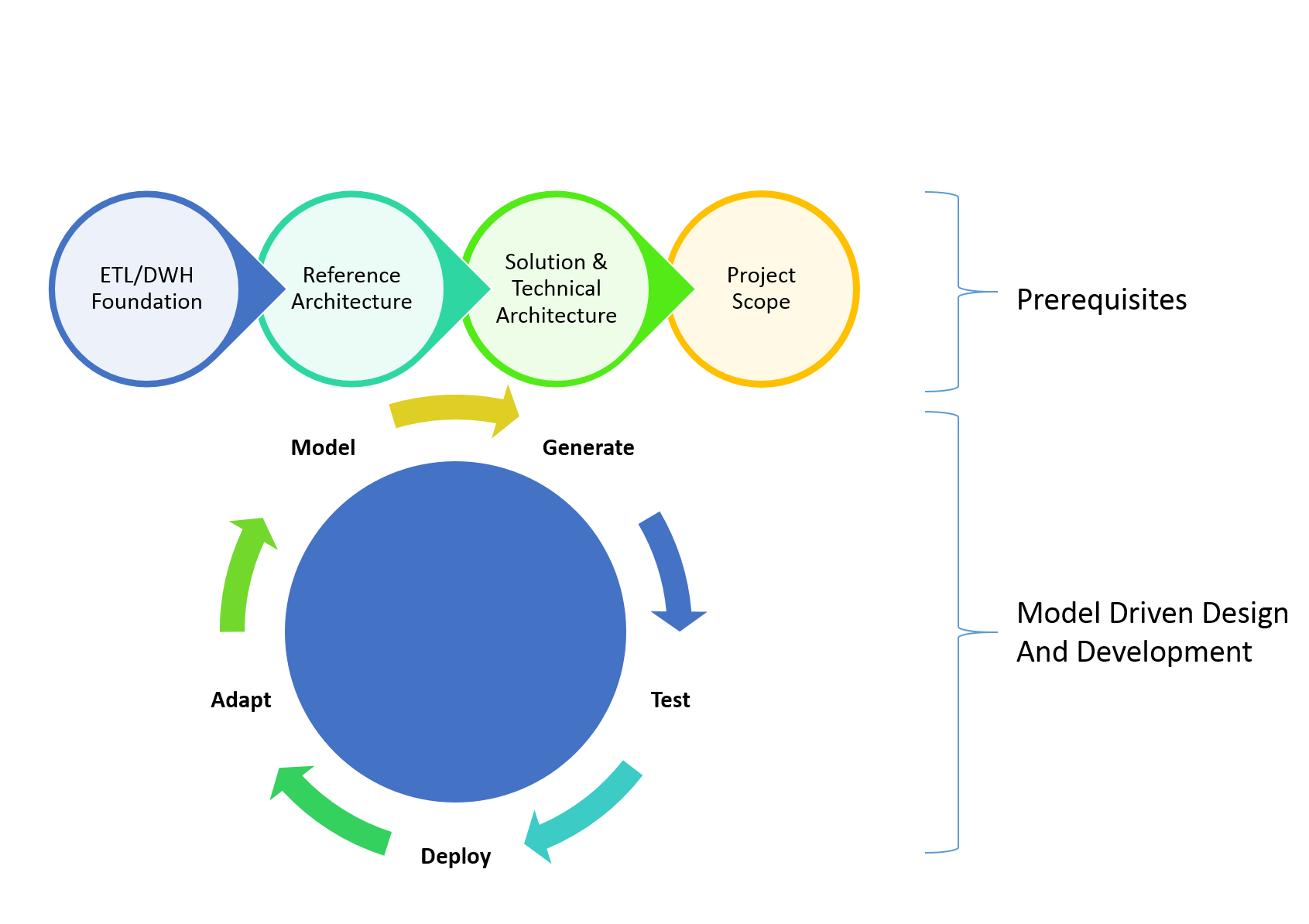
In this approach it is highly recommended to limit the scope to a short-term iteration; ideally bringing the cycle down to 2-3 weeks.

Figure 5: Project approach