Tampere University

Shubham Keshri

INTEGRATING PRODUCT DATA TO ENTERPRISE DATA WAREHOUSE

Master's Thesis
Information Technology
Examiner: Kari Systä
June 2020

ABSTRACT

Shubham Keshri: Integrating product data to enterprise data warehouse
Master's Thesis
Tampere University
Master's degree in Information Technology
June 2020

Manufacturing industries are gradually moving towards digitalization. This is the demand of the current market. End-user's buying behavior is strongly influenced by digital services offered along with nearly any product. These digital services are mostly driven by data. Data that is gathered during different phase of manufacturing process, sales, customer engagement etc. Data becomes a fuel in not only end-products but even internal business process optimization. Companies have been collecting and nurturing data from software platforms that support these business processes. A common challenge faced by companies is that data from these business processes are stored in different applications and their data is decentralized. Therefore, teams are working in silos.

Data warehousing provides a solution to this challenge. It integrates data from different software applications used across organizations and maintains a central repository of data. Thus, allowing business users to derive meaningful insights from data combined from different applications.

This thesis focuses on one of many business processes, product data management. It tries to study different approaches to integrate product data to enterprise data warehouse. After studying the approaches, it also discusses initial implementation of the selected solution.

Keywords: Product Data, Enterprise Data Warehouse, Integration

PREFACE

The thesis is submitted for a master's degree in information technology at Tampere University of Technology. Research and studies are performed under supervision of university lecturer Timo Aaltonen from the department of software systems. Practical experiments and development for this thesis was conducted for Wärtsilä, Finland. Implementation part of the thesis was supervised by Harri Piili (solution architect, Wärtsilä Finland). Theoretical part of the thesis was guided by Inka Vilpola (former director of digitalization, Wärtsilä Finland).

I would like to take this opportunity to thank each of them for their contribution, faith and patience in me that made it possible to bring the best work to my knowledge. In the entire process of writing and implementation, I continuously learned something new and improved my understanding on the subject. Last but not the least, many thanks to my family for their motivation and support.

Espoo, 16th June 2020

Shubham Keshri

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

CC license	Creative Commons license
PLM	Product Lifecycle Management
PDM	Product Data Management
EDW	Enterprise Data Warehouse
ITK	Integration Toolkit (Teamcenter)
API	Application Programming Interface
UML	Unified Modeling Language
SQL	Structured Query Language
T4EA	Teamcenter for Enterprise Applications
ETL	Extract Transform Load
POC	Proof of Concept
TC	Teamcenter
TUT	Tampere University of Technology
URL	Uniform Resource Locator.

1. INTRODUCTION

Industries are moving towards *digitalizing [1]* their products and services. Firstly, because they want to improve their operational process to enhance productivity. Secondly, they want to increase their revenue by introducing *value added services [2]* around their products. And finally, also because they want to have a competitive edge amongst other players in the market. In practice, for global industries like *Wärtsilä [3]*, it means that their business processes and data must be supported by IT tools specific to their needs. And to support the business process, there are software applications that are used by enterprise. Business processes are different in nature, for example, finance, sales and customer relationships. Therefore, software applications supporting them also are different and so are their data storage and data structure mechanism. This makes it difficult to get a holistic view of data across organizations.

Combining data from these business processes allow business users to answer some key questions related to their business. For example, if someone wants to find out net sales for a particular customer in the last 5 years. Data has to be combined from software applications supporting sales and customer relationships. *Data warehousing [5]* provides the solution to this problem. It defines a centralized storage of data coming from different software applications. This allows to build new relationships by combining data from a variety of applications to get a holistic view of enterprise data. Spanning across enterprise, data warehousing solution is called *enterprise data warehouse (EDW) [6]*. Further, these relationships are used to create business reports which are used by business stakeholders. *Product data management [7]* is one of many business processes at Wärtsilä. This thesis discusses different approaches to integrate *product data [8]* with EDW. Studies and implementation related to this thesis was conducted at Wärtsilä Finland under the supervision of *solution architect [9] and enterprise architect [10]*.

1.1 Introduction to Product data management at Wärtsilä

Product data management is a term used to describe the process of capturing, processing, storing and analyzing data for a product (engines and power plants in case of Wärtsilä). The PDM team at Wärtsilä is responsible for management of the entire *product life cycle [11]* process of manufacturing, maintenance and spare part management for

engines and power plants. Data is generated, processed, stored and analysed in phases: gathering requirements, solution design, product manufacturing, assembly of individual parts, customization, maintenance, and spare part management.

Taking a typical example for manufacturing an engine. Firstly, in requirement gathering phase information is gathered from customers related to their needs. It usually comprises interview questions, meeting notes, design files etc. Next, in solution designing phase architects and design engineers create 3-D models for different parts of the engine, for example, a *crankshaft [12]*. In the manufacturing phase, these 3-D models are converted into real parts of an engine in factories. Assembly line workers and engineers use data from the holistic *3-D model [13]* of the engine to assemble different parts of the engine together. Engines are not manufactured as one size fits all. Also, engines have to be tailored based on the local guidelines from the customer's geographic location. For example, the color of an engine has to follow certain standards in different countries. Therefore, customizations are made and finally, engines are delivered to the customer. After installation, maintenance is followed for the service period.

In all of the above phases, data is generated, gathered and stored in different forms, for example, *flat files [14]*, pdf design documents, relationships, 3-D models. Another challenge is that manufacturing units, engineers and customers in global companies like Wärtsilä are located in different geographical regions. This requires a centralized application to store and process product data. The software application that helps manage product data at Wärtsilä is discussed in the next section.

1.1.1 Product data management using Teamcenter

Teamcenter [15] a *Siemens [15]* product is a *software platform [16]* that enables centralized management of product data. It is a commercial software and is sold with a license to be used in an enterprise. It allows storage and version control of data at each step of a product life cycle. It stores, for example, process information, design and documents related to the product or service. The main functionalities provided by Teamcenter under PDM scope are product data and lifecycle management, product concept and design management, engineering change management, product configuration and structure management, product portfolio management, scope of supply management, and manufacturing process management.

Again, using the same, example for, manufacturing of an engine. Product data in this context refers to data that defines the engine, manufacturing process and delivery. En-

gine after installation also generates operational data which is collected, stored and analyzed for improvements and repair. Product lifecycle or in this case engine lifecycle starts from gathering requirements and lasts until the engine has been dismissed from operation. Product design and concept management deals with design of the conceptual design of an engine and its components. Change management refers to how well the application is able to adapt to changes made, for example, in one part of the design. There are processes defined how a change is captured, implemented and propagated to different components that are eventually affected by that change. Similar processes are followed in spare part management. All these features are embedded in different models of the application that works in harmony. This provides the users visibility and access to different phases, for example, design, manufacturing and maintenance of the product.

At Wärtsilä, Teamcenter is one of the main applications used with more than 3000 users globally [17]. It is widely being used in manufacturing marine solution, energy solution and service business units in supporting product data management. Although, having only product manufacturing data is not enough to bring insights and business value. Product data combined with sales, cost and other business processes allow business analytics to bring useful insights to grow business. One typical use case is to predict the quantity of purchase for engine parts in the upcoming year. For accurate prediction, combined data from product sales history and parts lists from product data can help. One way could be to integrate data from the sales database in *Salesforce* [4] to Teamcenter database. This approach is not very practical due to following reasons:

- Teamcenter application is not suitable for handling bulk data access.
- Data storage mechanism of Teamcenter is too complex.
- Teamcenter application contains work in progress data. It might not be accurate for historical analysis.
- It has data relevant only for PDM processes.
- Teamcenter application database is not meant for complex analytical queries to support business reporting.

With these limitations it is clear that Teamcenter alone is not sufficient to allow business users and analysts to build reports to help them make business decisions. This also helps to understand why having a centralized data storage capable of storing and processing large amounts of data is necessary for business reporting. And, eventually help

in understanding why integration of product data to enterprise data warehouse is required.

1.1.2 Introduction to Enterprise data warehouse

Data Warehousing is a solution that provides large data storage and high processing capabilities. It provides a platform for *business reporting and analytics [18]*. Data Warehouse stores data from sources such as enterprise resource planning (ERP) application [19], customer relationship management (CRM) application [20], IoT devices [21] and product data management application.

Data comes from a variety of source applications and their structure is different. Software application which provides storage and processing for data warehousing allows distributed processing. This enables massive parallel processing of data on large datasets. There are four stages of data in data warehousing: data extraction from source, data ingestion to Data Warehouse, data modelling and data consumption.

In the data extraction phase, data is extracted from the source system. In the next phase of data ingestion, extracted data is loaded to *distributed storage [22]*. Structure of data at this stage is still in the source application model. To create relationships between data coming from multiple source applications, a common model is chosen, and data is modelled accordingly. Each business unit then creates their own *data mart [6]* suitable for business reporting to answer business questions. Finally, these reports are consumed by business users.

A few examples of business reporting solutions consuming data from data marts are Microsoft business reporting [23], Qlik view [24], Microsoft power BI [25] and IBM Cognos [26].

1.1.3 Data extraction and ingestion

Data extraction is the phase where data is extracted in different ways from a software application. Some software applications expose a set of *APIs (Application Interfaces) [27]* for external applications to fetch data. Another common way to extract data from a software application is by creating connections to their back-end database and fire *SQL queries [28]* to fetch data. The next step after extraction is ingestion of data into data warehouse storage. Depending upon type of data storage and frequency of loading data there are many commercial and open source software platforms specializing in ingestion of data. This brings the discussion which leads to the research question and purpose of this thesis.

1.2 Research questions

Need for integrating Teamcenter data with enterprise data warehouse arrived when the legacy reporting application WDMS (Wärtsilä database management system) used by Wärtsilä for business reporting needed to be phased out. These reports were declared outdated and needed more information from other applications to support business. Wärtsilä had already invested in a data warehouse platform running on *Oracle Teradata [29]*. The obvious solution was to extract data from Teamcenter and ingest into Oracle Teradata. This would allow business users to use modern business reporting tools such as *Microsoft SSRS (SQL server reporting services) [30]* to create new reports with data combined with other applications such as Salesforce.

Arriving to the research question: This thesis was organized to find out the best approach to integrate product data to enterprise data warehouse at Wärtsilä.

1.3 Structure of thesis

This thesis is structured into four chapters. First chapter discusses the introduction. It starts with describing the scope of thesis and about Wärsilä and its line of business. Later this chapter discusses two software applications associated with the study. Product data management tool Teamcenter and Enterprise data warehouse. Later, it briefly discusses the data extraction and ingestion approaches and opens up the research question.

Next chapter discusses the theoretical background of product data management and enterprise data warehouse. In this chapter different types of integration strategies are discussed based on level of application integration, data processing timeline, topologies and technical implementations. Following which is the research methodology and material chapter. This chapter focuses on action research methodology. Research is divided into three iterations and each of them are evaluated based on some evaluation criteria. Finally, in the result and analysis chapter pros and cons of each approach is discussed and the best approach is selected for development and deployment. In the end the thesis outcome is summarized in the conclusion chapter.

2. THEORETICAL BACKGROUND

This chapter discusses the background and *technical architecture of Teamcenter [15]*, data warehousing and integration. It also discusses application types, data types, data storage mechanism and other factors that will help determine the possible alternatives. The study will also influence the selection for the right software tool for extraction of data from Teamcenter and ingesting it into enterprise data warehouse.

Extraction of data from one application and ingesting it into another application is manual process. In software development context, this process is also commonly referred to as integration of applications or *EAI (enterprise application integration) [22]*. Integration of two or more applications means automating the process of manual *replication* of the same data across multiple applications. Integration *increases organizational efficiency, reduces duplication of procedures and records and provides an agile system with less redundancy [31]*.

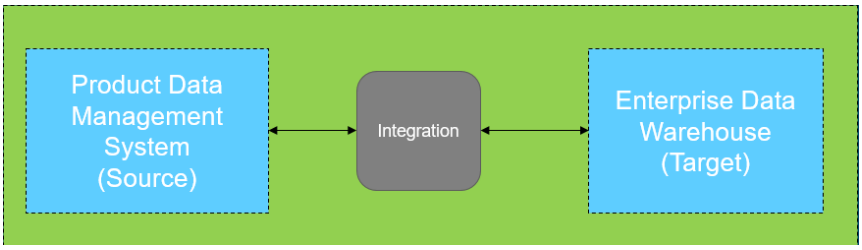

Figure 1. *Integration (high level)*

Figure 1 shows integration in context with two applications: Teamcenter (product data management system) and Oracle Teradata (enterprise data warehouse). The software application from which data is extracted is also known as source application, and the software application to which data is ingested is also known as target application.

2.1 Product data management application (Teamcenter)

Technical architecture behind Teamcenter on a high level has two parts: application and database *(Figure 2)*. Application part consists of the code which enables the user interface, functionality and features of the application. It is designed using *object-oriented design [32]*. Database behind Teamcenter stores data related to the product and technical metadata related to the Teamcenter application. Teamcenter allows users to select from a list of commercial databases that it can support. In case of Wärtsilä, the database behind the Teamcenter application is *Microsoft SQL server [30].*

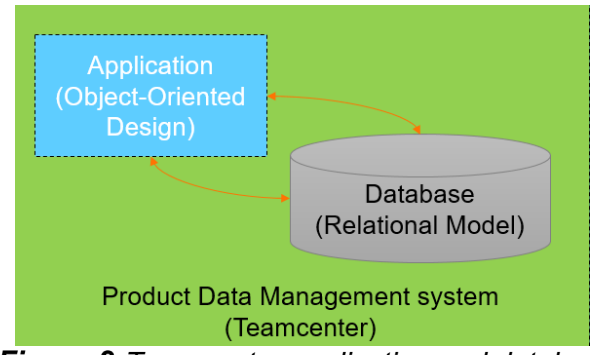

Figure 2. *Teamcenter application and database*

2.1.1 Teamcenter application

Teamcenter application has object-oriented design and is based on *classes* and *objects*. Object-oriented design is used in scenarios where an application wants to persist their data in a relational database. The process of mapping between object-oriented data models with relational databases is called *object-relational mapping [33]* and can be achieved with already available packages that generate the necessary code. The benefit of using object-oriented design is that it provides a structured representation of data.

Teamcenter client application consists of a master object called Workspace. Within the workspace object there are subclasses. The main subclasses within the workspace object class are [34]:

- **Item**: Fundamental physical or conceptual entity to manage information. For example: parts, documents and equipment.
- **Item revision**: The version of an item. Whenever there is a modification in an item, the change history is maintained as a different item revision of the original item.
- **Form**: Form is used to view or modify name/value pairs. These are then stored in a dataset object.
- **Dataset**: It is the actual storage class of the object. An example could be linked to a physical document.
- **BOM view**: Bill of materials (BOM) is the assembly structure of a product. Example can be a BOM of engine assembly.

- **Relation**: It is an object which relates one workspace object to another.

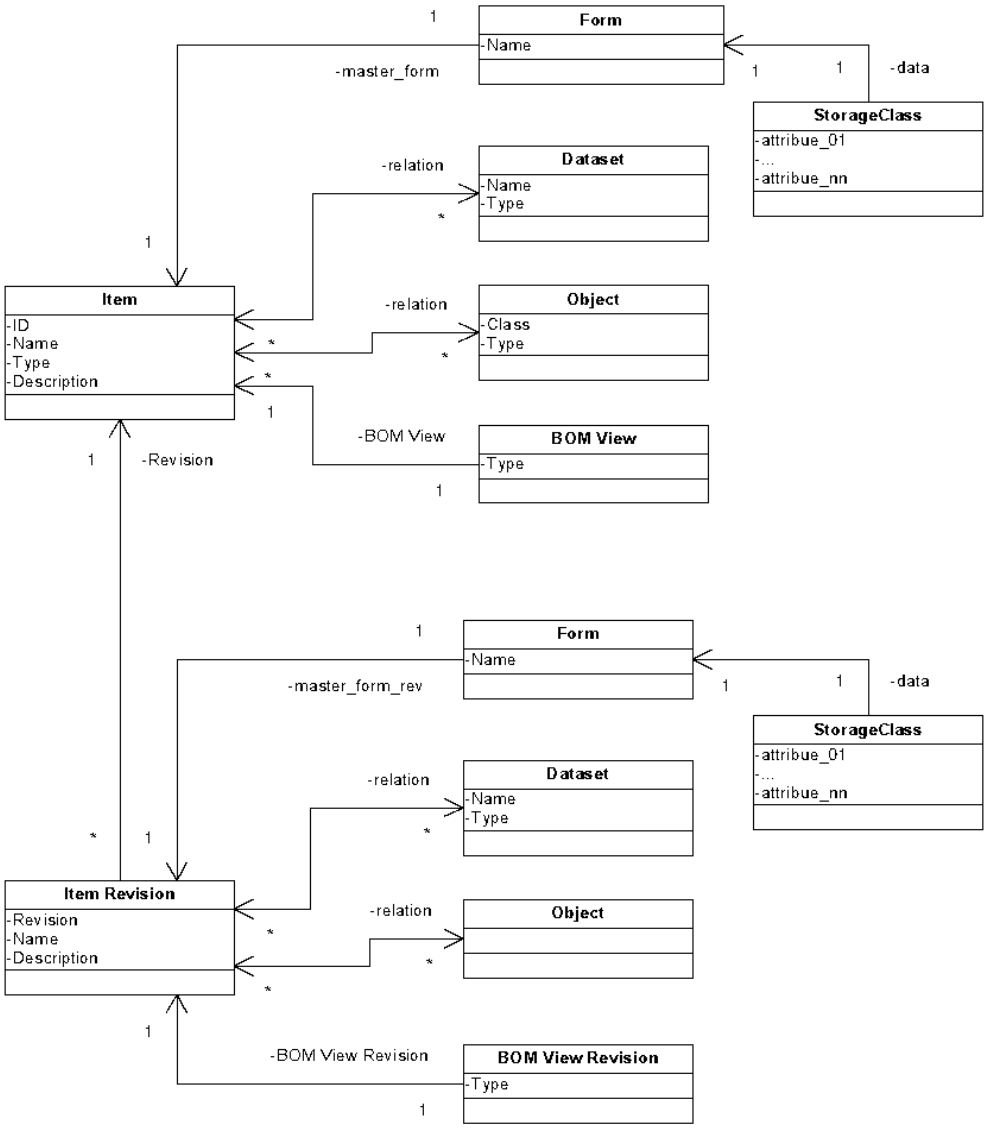

Figure 3. *Teamcenter objects and relationship design*

As an example of object-oriented design, *Figure 3* shows the Teamcenter *out of the box [15]* entity relationship diagram. Out of the box here means that this is the default design provided by Teamcenter application. The design is often customized to meet the business needs of the customer (Wärtsilä in this case).

2.1.2 Teamcenter database

The relational database used by Teamcenter at Wärtsilä is Microsoft SQL server which is a *relational database management system [35]*. The object-oriented design of the application is mapped with the relational database using *object-relational mapping [33]*. An example of a relationship between item revision table and item table is shown in *figure 4*.

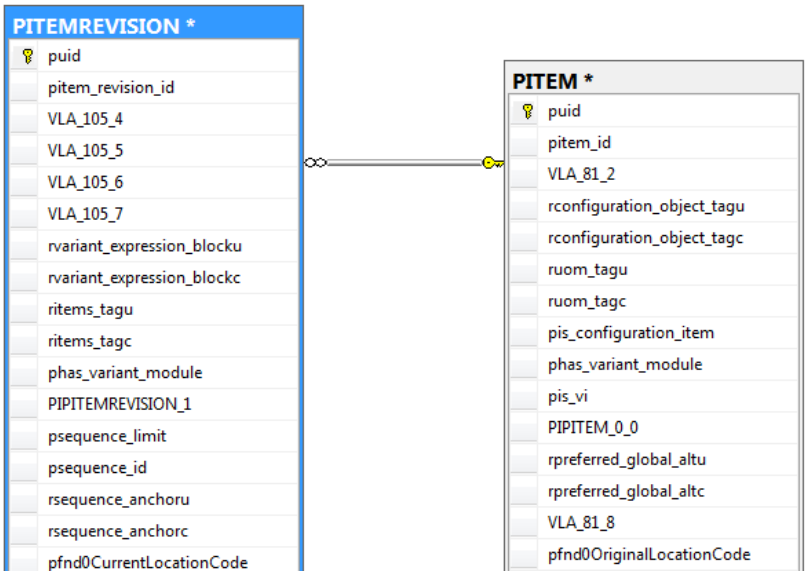

Figure 4. *Teamcenter relation example*

Teamcenter application takes care of the object relational mapping. Apart from tables which contains business data, there are tables that contain Teamcenter system data. Teamcenter database consists of tables corresponding to the object, class and relationships. The mapping is not always one-to-one. Teamcenter does not provide documentation of the object-relational mapping. There is, however, an XML file called “model.xml” in the TC_DATA folder of the Teamcenter installation folder. This file helps in decoding the actual table names in the database corresponding to Teamcenter application objects. “model.xml” becomes an important document for extraction of data from Teamcenter.

2.2 Enterprise data warehouse

Enterprise data warehouse is an enterprise wide centralized repository of business data. Data is first loaded into the staging area of the underlying storage. It is then modelled and finally, this data is primarily consumed to create business reports and perform business analysis. The underlying storage for an enterprise data warehouse can be a relational database, a *no-SQL database [36]*, files or combination of these. The selection usually depends on the data type and structure of the majority of data in an organization.

Typically, for global companies having large volumes of data, a no-SQL distributed database, for example, Oracle Teradata is a good option.

Only having the correct storage for data warehousing is not enough. Having a good and common data modelling technique in a data warehouse is really important. Common examples of data modeling techniques are *entity relationship modeling [37]* and *dimensional modeling [38]*. Dimensional modeling is more suitable for data warehousing solutions. Dimensional data modeling is used for decision support applications that are highly optimized for reporting and analytics purposes and efficient handling of data. In databases, it is often represented as *star schemas [39]*. The name comes from the fact that it resembles a star structure. The star schema comprises two kinds of tables, *fact and dimension tables [39]*. Fact tables are designed to store performance measurements for organization's business process events. Dimensional tables on the other hand are used to store textual context associated with fact tables.

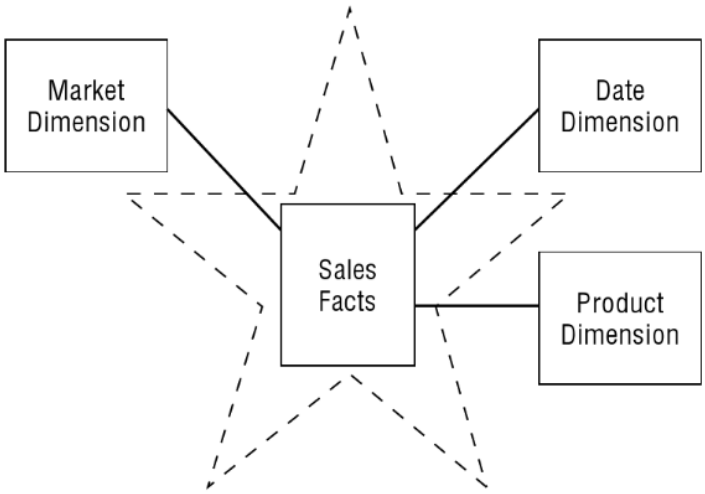

Figure 5. *Star schema [38]*

The data model is flexible and highly scalable. *Figure 5* demonstrates an example of star schema. One fact table can have multiple dimension tables related to it, which makes it organized. Another benefit of dimensional model is that it supports ad-hoc queries which enable self-service. This allows business users to take advantage of self-service reporting applications to create their own reports.

2.3 Enterprise application integration (EAI)

Enterprise application integration (EAI) [22] makes enterprise applications share business data and functions with each other. Applications can be heterogeneous ranging from software platforms like Teamcenter to in house-built solutions. The problem of extracting data from Teamcenter and ingesting it into enterprise data warehouse can be

solved by integration. EAI is implemented based on the business and structure of the organization and vary from enterprise to enterprise. EAI can categorized based on: levels of application integration [40], data processing timeline in integration, topologies followed in integration [22] and finally, technical solutions for integration.

2.3.1 Levels of application integration

EAI can be performed at different levels and is broadly classified into three levels: data level integration, object level integration and process level integration.

Integration of applications may fall under one or combination of the above-mentioned level of application integration based on factors such as size of data and complexity of the design of the source and target application. Teamcenter allows data and object level integration. Enterprise data warehouse only allows data level integration. For this reason, process level integration is kept out of scope for this thesis.

2.3.1.1 Data level Integration

In *Data level integration [41]*, the databases or storage files are integrated directly. One of the very basic ways to implement it is using *FTP (file transfer protocol) [42]*. In FTP, files are copied from source location (storage) to the destination location (storage). Another common way is to connect the databases of the source and target application.

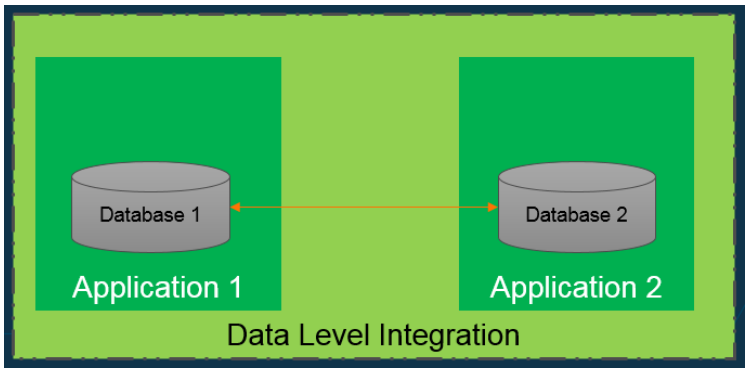

Figure 6. *Data level integration*

In *Figure 6*, application 1 and 2 are having database 1 and 2 running behind them which stores their data. In this type of integration, the application itself does not have any information about the integration. Benefits of performing data level integration is that it provides high speed data transfer due to a smaller number of proxies in between them.

2.3.1.2 Object level Integration

An object in an application refers to an *entity [37]*. In object level integration, the target application object talks with the source application object. It provides an additional abstract layer by giving accessibility to the objects of the source application that is visible in the user interface of the application.

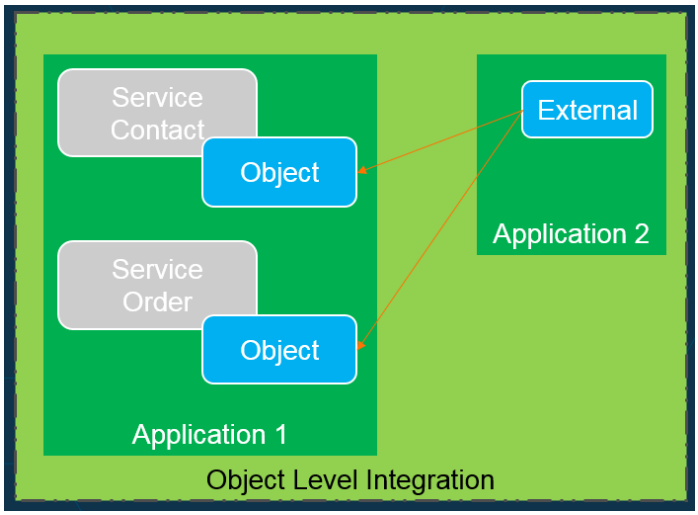

Figure 7. *Object level integration*

In performing object level integration, one application provides an interface to allow another application (application interface) to talk with the internal objects shown in *Figure 7*. These interfaces abstract the complexity and internal architecture of the application. In this kind of integration, the target system has the information of the integration. However, the source application has no information about the integration and the target application.

2.3.2 Data processing

There are two possibilities of how data can be processed while data is loaded to target application from source. Either the data can be processed right away when it is created in the source application or, it can be processed in *batches [43]*. The first case is called real time processing of data and latter is called batch processing. What time is considered real time is relative to business needs.

2.3.3 Network topologies followed in integration

EDW can have many software applications as a source for data. When more than two application comes into picture of integration, common network topologies enabling connectivity are followed. Three network topologies followed in integration are point-*to-point topology* [44], *hub and spoke topology* [22] and *bus topology* [22].

2.3.3.1 Point-to-point topology

Point-to-point topology connects one application to another individually with a link between them. It is trivial when there are only two applications to be integrated. More than two applications mean each application has to be integrated with each other shown in *Figure 8.* It is a tightly bound integration topology, cost effective but less scalable.

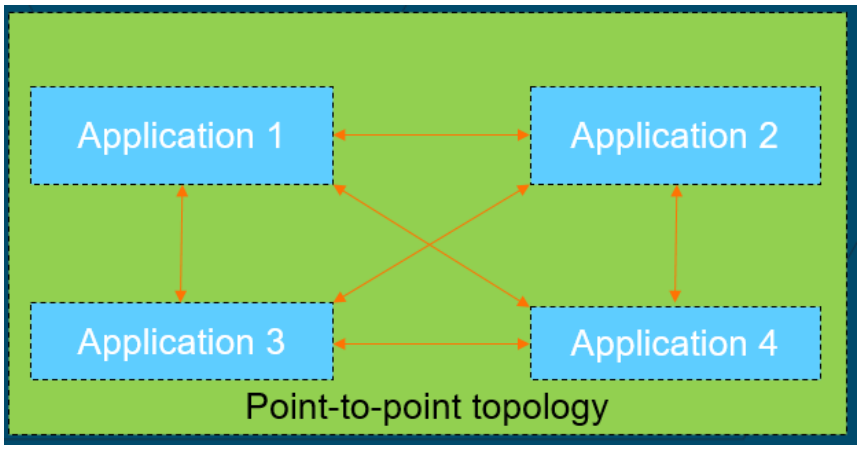

Figure 8. *Point to point topology*

The bottleneck of such topology is the number of applications participating in the integration. As soon as the number grows, these links in between applications start growing resulting in a mesh of links hard to manage.

2.3.3.2 Hub and Spoke topology

Visually hub and *spoke topology* [22] resembles the tire of a bicycle shown in *Figure 9*. Hub represents the central axis of the tire with software applications connected at each end of the spoke. This topology reduces the number of links between applications. Hub acts as the centralized source and destination for data. This is comparatively more expensive to build than point-to-point because it requires to maintain a centralized hub.

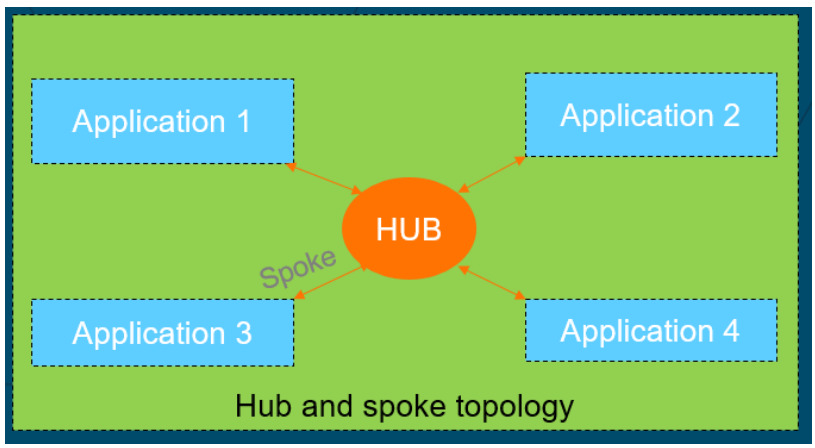

Figure 9. *Hub and spoke topology*

Hub in technical implantation and can represent a relational database, a data warehouse or an application depending on the use case. As hub acts as the centralized storage of data, it also means it becomes a single point of failure. Therefore, often the hub is kept behind a *load balancer* with a *failover* hub to provide *high availability [45]*.

2.3.3.3 BUS topology

Bus topology, similar to hub and spoke has a centralized application for communication among different applications. However, the bus only acts as a channel for communication

and not storage. Bus provides an interface to more than one application. If information changes in one application, the change travels through the bus to communicate the change in all other applications as shown in *Figure 10*. Implementation of bus topology requires high initial investment but is cost effective when adding one more application to the hub. It is also flexible and requires no change to existing connections. The bottleneck of this topology is the failure of the centralized bus.

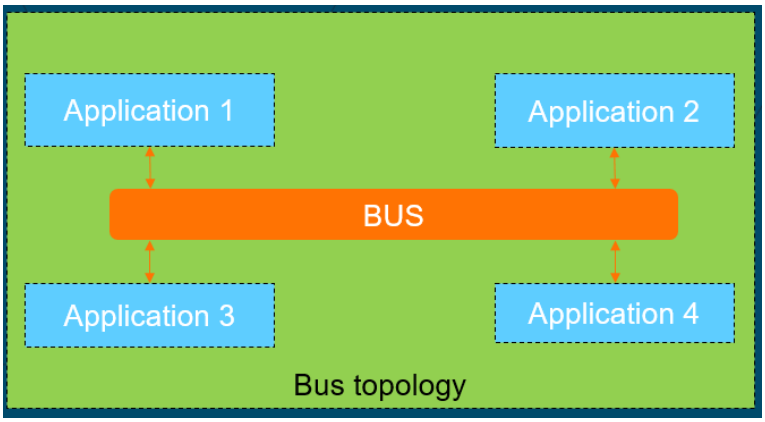

Figure 10. *Bus topology*

2.3.4 Technical implementations in EAI

Common ways of implementing EAI in an enterprise are by using *application programming interfaces (APIs)* [27], *enterprise service bus (ESB)* [46] and *extract transform load (ETL)* [47]. Teamcenter provides a set of TKI APIs which can be used for extracting data from Teamcenter. Teamcenter and EDW both have databases and hence ETL is a valid alternative. ESB is not applicable in this case and kept out of scope and discussed in brief.

2.3.4.1 Application programming interface (API)

Application programming interface defines how an application can open doors to other applications for communication. It defines the protocol of how the services or data offered by the application can be accessed. Some APIs can be pre-build coming from commercial applications. Sometimes it is also possible to create custom APIs tailored to business needs.

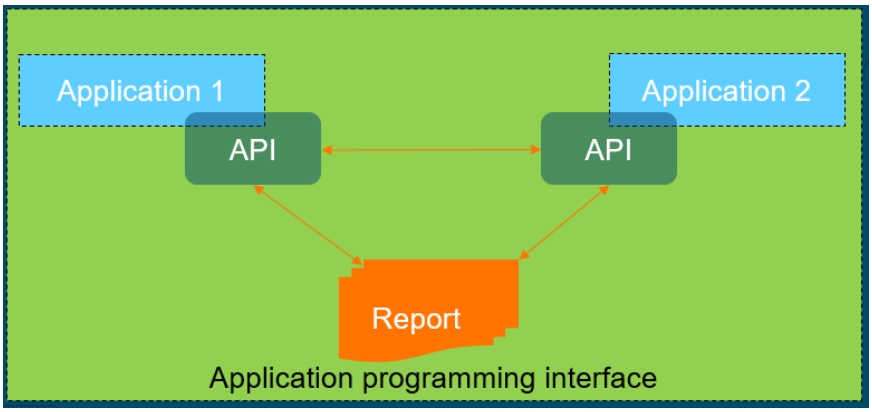

Figure 11. *Application programming interface*

APIs can have multiple uses. It can provide an interface for application integration and also can be used by reports to access data as shown in *Figure 11*. APIs operate at application level, meaning they hide the internal architecture behind the application. Teamcenter provides a set of pre-built APIs called *information toolkit (ITK) [15].*

2.3.4.2 Enterprise Service Bus (ESB)

Enterprise service bus can be seen as a technical implementation of bus topology. The centralized bus (*Figure 11*) acts like a message server. The message, usually in *XML,* travels in the bus to reach the destined application. This *decouples [48]* applications with each other. Some commercially available ESB software platforms include *IBM WebSphere [49]* and *open ESB [50].*

2.3.4.3 Extract Transform Load (ETL)

ETL is the process of extracting data from a source application, transforming data and loading it to the target application. Transformation phase in ETL includes processes such as *data validation [51], data cleaning [52], and data aggregation [53] (Figure 12).* Transformation of data requires processing of data. Processing requires a processing engine. There are two ways in which data can be processed. One way is to utilize processing capabilities of the underlying storage. For example, if data is stored in a database then data can be transformed by writing SQL queries that perform functions like data validation, data cleaning and aggregation. Another way of processing data is to load data to an external processing engine and then load it back to the storage. The orchestration of data transformation and loading can happen either by using functions provided by a database or an external software.

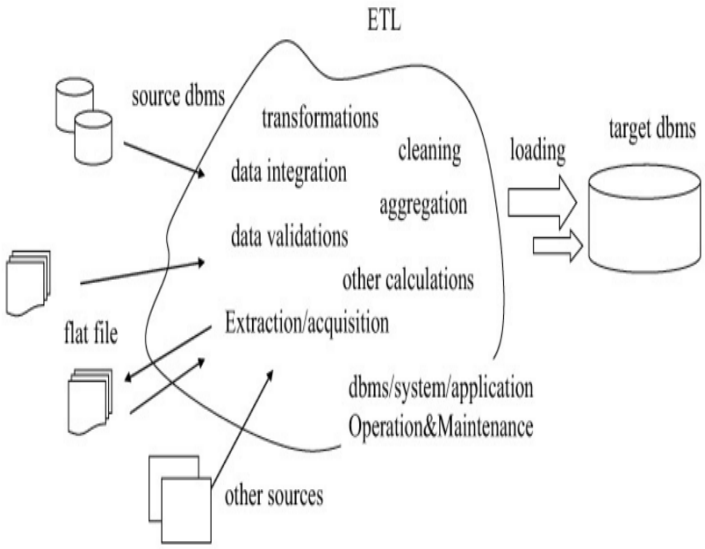

Figure 12. *ETL process [38]*

There are a variety of ETL tools available in the market. Some are commercial and some are open source. Some organizations also invest in building their own *in-house* ETL tool. Commercial tools supporting ETL not only provides an easy way implementing ETL processes, but also provide additional features that help in maintenance and continuous service of data integrations such as [54] monitoring of the ETL workflow, scheduling of the workflow, data profiling, built-in transformation functions, code versioning and data lineage.

2.4 Existing solution

Re-visiting the research question again, an approach to integrate product data to EDW already existed. Existing solution utilizes Teamcenter API to extract data from Teamcenter. This solution was developed by an external vendor for Wärtsilä. Maintenance and enhancements were also taken care of by the same vendor. Solution design is shown in figure 13.

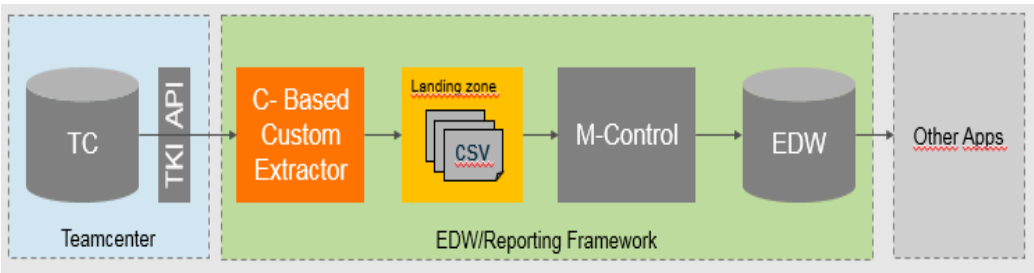

Figure 13. *C-Program API based Integration Architecture*

Solution includes a custom C-program which uses Teamcenter ITK (Integration Toolkit) API to query Teamcenter to fetch objects and populates a list of CSV files. These files are categorized based on Teamcenter item type. Another 3rd party tool M-control picked

these *CSV files [14]* from the server and loaded data into the enterprise data warehouse. This setup included multiple steps and it was expensive to maintain and required resources which were not easily available. It was complicated to configure the extraction as it was configured using flat files and command line parameters. The load setup was based on windows scripts. Code was running on the same server where Teamcenter was hosted which was causing performance issues. Another challenge with this solution was data reloading. Interview with stakeholders for the tool clearly indicated their day to day struggle with managing the tool. And they needed a better solution which was automated and more stable.

Week Number	Days of Week	Loading Time (in Hours)	Reason	Delay	Expected Time for Completion (in Hours)
16	Monday		Failure in Extraction		10,00
16	Tuesday		Failure in Extraction		10,00
16	Wednesday	11,50	Success with Delay	1,5	10,00
16	Thursday	9,00	Success with Delay	-1,0	10,00
16	Friday	8,00	Successful in Time		10,00
17	Monday	4,42	Successful in Time		10,00
17	Tuesday		Failure in Extraction		10,00
17	Wednesday		Failure in Extraction		10,00
17	Thursday	11,50	Success with Delay	1,5	10,00
17	Friday		Failure in Extraction		10,00
18	Monday	4,17	Successful in Time		10,00
18	Tuesday	4,42	Successful in Time		10,00
18	Wednesday		Failure in Extraction		10,00
18	Thursday		Failure in Extraction		10,00
18	Friday		Failure in Extraction		10,00
19	Monday	3,21	Successful in Time		10,00
19	Tuesday	4,17	Successful in Time		10,00
19	Wednesday	4,42	Successful in Time		10,00
19	Thursday		Failure in Extraction		10,00
19	Friday		Failure in Extraction		10,00
	Avg. Load Time=	6,48			

Figure 14. *Daily Load Status for API based Integration*

Daily loads with existing tool was monitored to analyse stability. *Figure 14* shows daily status of the tool for 4 weeks. In about 50% of the days during the monitoring period loads were failing. Also, average load time for loading data from Teamcenter to enterprise data warehouse was 6,48 hours which sometimes extended up to 11.50 hours. It resulted in problems with performance of Teamcenter server by throttling CPU of the sever hosting Teamcenter during normal working hours .

This chapter discussed the theoretical background of Teamcenter, data warehousing and enterprise application integration. The next chapter research methodology and materials discusses in detail about how research was carried out at Wärtsilä for the selection for an appropriate data extraction and ingestion tool for integrating product data to enterprise data warehouse.

3. RESEARCH METHODOLOGY AND MATERIALS

The research or practice for this thesis was carried out by applying *action research [55]* methodology. Action research is also sometimes referred to as *action science [55]*. Action research is driven by four core principles shown in *Figure 15.* Implementation of action research is carried by iterating between planning, acting, observing and reflecting.

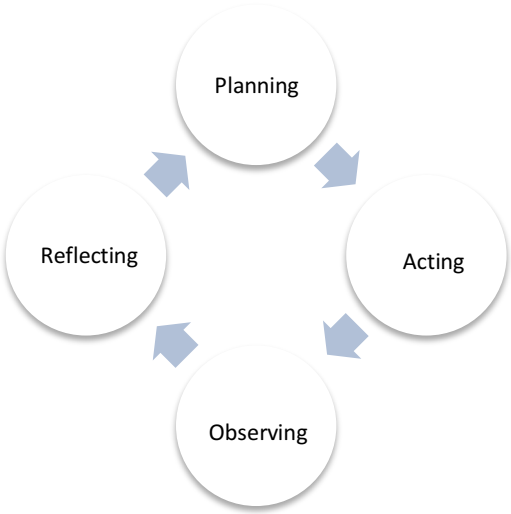

Figure 15. *Action research [55]*

Initially, in the planning phase the practitioner identifies the focus of the research. The research questions are set at this phase. What change the practitioner is planning to make, and how that change will get measured is defined in the planning phase. In the next phase, as the name suggests, action or acting is the most important aspect of action research. Acting which starts with gathering data, finding out the background of the problem and solutions that are already available. Acting requires the involvement of the practitioner to implement the change. Moving forward, in the observation phase the implemented change is observed. It is compared based on the defined measurements and the previous iteration of the research. Based on the observations, the practitioner makes a reflection of the solution. The learnings from the iteration are analyzed. Feedback is gathered from stakeholders who are affected by the change. Decision is then made whether the practitioner wants to make improvement in the same solution or should look into another approach. Based on the reflection, either the iteration repeats or settles.

Selection of the correct research methodology for this thesis itself was also one key study. Most common research methodologies that study suggests are *empirical research*

[56] and *quantitative research [57]*. The reason why Action research was selected to carry out the research for this thesis was because of the following reasons:

- The author of the thesis is not only an observer but is also influencing the events of research
- action research requires total involvement of the practitioner and they become the *change agents*
- action research also involves *informal* and *in-depth interviews, ethnographic methods of observation and participation [55]*
- Action research is usually carried out by *academic* practitioners who are employees or an *external consultant* of an organization

Above mentioned criteria satisfied the needs of the current problem author was looking into. Hence, thesis work proceeded with applying action research.

3.1 Planning

To ramp up, the planning phase started with identifying the focus for the thesis. Wärtsilä decided to phase out their legacy reporting application WDMS (Wärtsilä Database Management System). This decision naturally created the need to look for an alternative solution for business reporting with Teamcenter data and also combining data from other applications. Because Wärtsilä had already invested in a data warehouse, it became the default option to integrate Teamcenter data to data warehouse (Figure 16). Also, integration was a broad topic. It had to be narrowed down to smaller steps to individually focus on solutions. With the information gathered from theoretical studies, it was evident that there are multiple approaches to achieve integration. It needed some strategy and evaluation criteria to select the best approach.

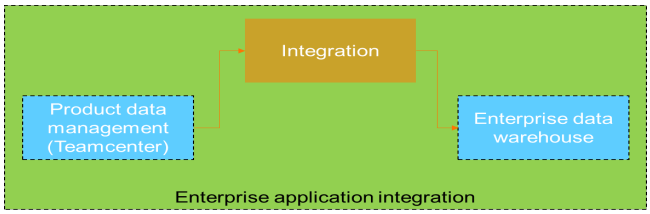

Figure 16. *Enterprise application integration*

At Wärtsilä, tied to this integration there are several teams under the business unit 'information management'. Firstly, the PDM team is responsible for managing the product data lifecycle. Secondly, the analytics team responsible for managing and maintaining

data warehouse and business analytics solutions. And finally, the enterprise architecture team responsible for maintaining the enterprise architecture of software applications at Wärtsilä. Individual *informal interviews [58]* were carried out with at least one stakeholder from each team to get a better understanding of how this integration would affect in terms of data ownership, vision for enterprise integration and also to learn from past experiences with each team. This would allow the work to be carried out in harmony with other applications. Stakeholders that were closely related to the applications involved in enterprise application integration included: solution architect [9] and solution manager [59] from product data management team at Wärtsilä, solution architect from analytics team at Wärtsilä and enterprise architect [10] from enterprise architecture team at Wärtsilä.

Purpose of the interview was to find out the current situation of the integration, how is the current situation affecting the stakeholder, what are the problems faced by stakeholders and expectations of stakeholders from the future solution. Response from the stakeholders helped in understanding the current situation. It was found out that a vendor had built a custom-made application for extracting data from Teamcenter. This application was dumping files to a server. Then these files had to be manually copied to another server from where another application picked up data and loaded into the data warehouse. However, stakeholders were not satisfied with the existing solution. Everyday business was getting affected due to missing files, manual errors etc. Following were some concerns and feedback gathered about the existing solution:

the solution was hard to maintain
adding changes to existing solution was difficult
there was frequent failure in loads
team lacked skills to maintain the solution
data load took long time
reloading of data involved lengthy manual work
And solution in general was unstable and unreliable.

Along with the interviews, study of the technical nature of the two applications Teamcenter and enterprise data warehouse was important. Interview also revealed that there was already a custom application made to satisfy the need. Study of the existing solution was also necessary to understand the gaps. This information also was gathered based on meetings with different stakeholders.

First phase started in parallel with the interviews by studying Teamcenter architecture. Teamcenter on a very high level consists of two main components: an application handling different functions and database for storage Figure 3. Data generated and manipulated in the application is persistently stored in the database. Teamcenter also stores application metadata also in the database along with user data. Meaning the relational

tables in the database consists both application metadata tables and business data tables. To extract business data from Teamcenter, either the application can be queried or the database. Teamcenter provides out of the box application querying interface using ITK (integration toolkit) API.

Second phase was carried out by studying the Enterprise data warehouse (EDW). At Wärtsilä Oracle Terada is used for data warehouse storage and processing. It is a commercial tool with a license. Data modelled methodology used in data warehousing is dimensional modelling. In *dimensional modelling [60]*, tables that are created are categorized basically in two types fact and dimensions. Fact tables by definition should represent metrics, measurements or facts of a business process. In the modelling diagram, it can be found in the center surrounded by corresponding dimension tables (*figure 15*). Dimension table by definitions should represent descriptive or textual attributes which are related to its corresponding fact table.

Figure 17 shows the snapshot of Wärtsilä EDW dimensional model. Table name with TF prefix is a fact table and tables with TD prefix are dimensional tables. Data coming from all applications are combined and models are created for specific business needs.

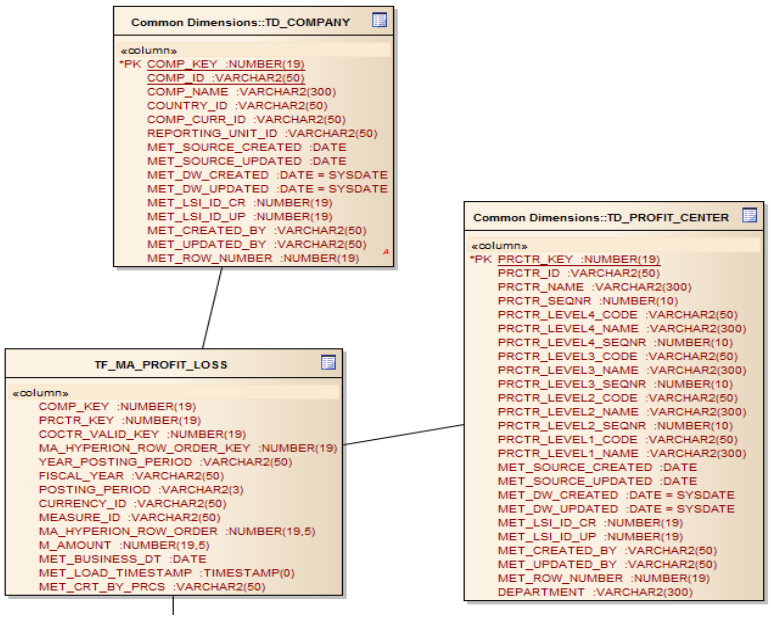

Figure 17. *EDW data model example*

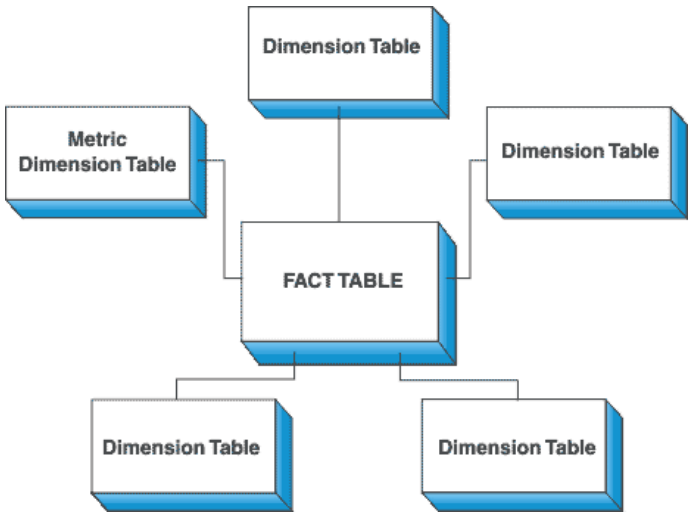

Figure 18. *Fact and dimension table structure*

Data modelling after ingesting data to data warehouse is carried out in these steps:

- Data when loaded to the data warehouse lands in the staging area. Here data is in raw state. Meaning there are no transformations applied to data which is captured from the source application.
- Data is then modelled using dimensional modelling. Transformations and functions related to dimensional modelling are applied.
- Finally, views are created on top of dimensional models from where applications consume data. This layer is also known as data marts.

In the third phase, the existing solution for data extraction and ingestion was observed. Existing data extraction tool was an in-house build solution using *C-language [61].* This application was querying Teamcenter using TKI API. Data queried was then written into flat files to a local machine. Triggering the extraction of data was manual and needed manual changes in the configuration files. Another tool called *M-control [62]* then fetched files from the server and loaded them into data warehouse tables. Data loaded to the data warehouse was also not modelled. This solution was developed by a 3rd party vendor [63] and the internal IT team at Wärtsilä lacked skills to develop it to make it stable. Loads were failing very often due to large numbers of flat files. It was getting inconvenient in handling files and maintaining data quality.

Based on evaluation from the result of interviews and observation of the existing solution, it was evident that a more robust and automated solution was needed to integrate data from Teamcenter to data warehouse. To improve the overall situation, there were couple

of alternative solutions for this problem. First by improving the existing solution. Second by building another custom-made solution. Third by looking for open-source solutions for ETL. Finally, by looking into commercial ETL tools.

Also, based on common opinion from all stakeholders a set of evaluation criteria were established to evaluate the future integration solution. Criteria included connectivity, performance, configurability, operability, tool development cost and maintenance cost

First evaluation criterion is *connectivity* [64]. In the context of EAI it means the ability of the integration application to be able to connect with different source or target applications. Applications can be connected via API, *ODBC/JDBC drivers [68]* or native drivers for certain applications. Connectivity also determines *protocols [64]* used, size of data transfer and security. It is important that connectivity is easy to establish, reliable and maintainable. Next, performance [65] can have more than one context. In EAI context, it is the key matrix that defines the time elapsed for loading data from one application to another. In object level, performance can be measured as the time elapsed in transferring the data from one object in one application to the same object in another application. The unit of measure here is a variable of time. So, less time in data transfer rate indicates better performance.

Another criterion is configurability [66]. It can also be defined as the ease of reusability of functions in the application. Configuration allows you to specify parameters based on which the application invokes the same function for a slightly different input. For example, based on input parameters, the application can either connect to a development database or a production database. So, a low configurable solution means it is not flexible enough to take parameters and the functionalities are limited. Next is operability [67]. Operability is the measure of the quality of an application. In other words, measuring stability and reliability. Ability of an application to perform functions without failures, and ease of recovery in case of failure. An application is also considered better in operability if it provides enough functionalities for collaborative development, testing and maintenance. This ensures that the development team is able to deliver quality results.

Development tool cost is another important criterion for evaluation. It is the estimated cost for developing or purchasing a license for the tool. In this case, the integration application. Another cost associated with this is the cost of building pipelines for integrating data. Development cost also includes cost for resources and infrastructure. Infrastructure here refers to server, database and network application. Generally, the cost of developing a custom tool is lower but more often consumes more effort in maintenance of the

application. And finally, maintenance cost is the estimated cost for maintaining the servers, monitoring the data pipeline workflow, fixing the defects and upgrading the system. This includes the estimated cost of resources to maintain the solution.

3.2 Acting

Based on the interview and discussion with the stakeholders, it was found out that there are two methods for extracting data out of Teamcenter. First by using TKI API provided by Teamcenter. And second, by direct querying the database of Teamcenter.

The existing solution was using TKI API to fetch data from Teamcenter. The main problem in this approach was data volume. Volume of product data required for creating business reports was high. This resulted in delay in extraction of data which eventually meant obsolete values in reports. Business users were heavily affected and needed up-to-date information. Another challenge with this approach was that Teamcenter API was using server resources to cater the queries for extraction. This led to slowing down of Teamcenter application during normal working hours.

To resolve these two problems, there were two alternatives. First alternative was to improve the existing solution to reduce the time of extraction, in such a way that it can finish extracting data in non-working hours. Challenge with this solution was competence. The existing extractor was developed using C programming language. It is a powerful yet old programming language. Second alternative was to try another approach where data can be queried directly from the Teamcenter database. To proceed further, there were three alternative solutions planned. The idea was to create *MVP (minimum viable product) [69]* with different approaches. And finally study them based on the defined evaluation criteria. The first solution included improvement in the existing solution. To overcome knowledge of C programming language, *Java* was selected as the programming language for development of a custom tool which will be extracting data from Teamcenter and loading it to data warehouse. Second solution was planned to utilize an open source ETL tool from *Talend [70]*. And the third solution included utilizing a commercial ETL tool called *Informatica Power Center from Informatica [71]*.

3.2.1 Solution - I: Custom Java Application

First solution started by developing a custom standalone Java application (*Figure 19*). Scope of this tool was to extract one entity from Teamcenter and ingest it to a data warehouse. In this case data was extracted from *item revision* [72] entity and loaded to item revision table in EDW. Scope also covered the ability of Java applications to trigger this

extraction, create dynamic queries based on the configuration parameters and finally, it should be possible to schedule workflows.

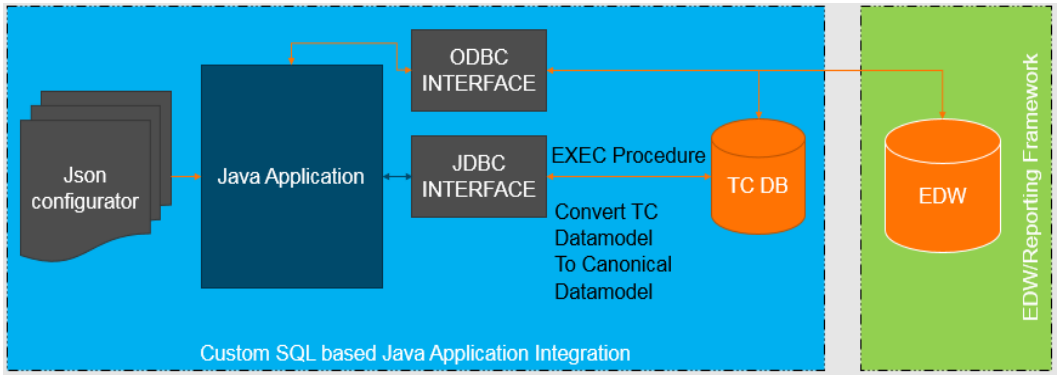

Figure 19. *Java application*

Teamcenter database was used as a source to extract data from Teamcenter in this approach to examine data extraction time. There was no official documentation available with instructions to query the Teamcenter database. However, Teamcenter server included a file called 'model.xml' in the installation directory, which described the model of table relationships in the database of Teamcenter. Stored procedures were created to extract data from these tables and dumped it into a *canonical model* [77]. There was a reason why the canonical model was placed in between Teamcenter tables and enterprise data warehouse. Querying data directly from Teamcenter tables would slow down Teamcenter application during normal working hours. Therefore, stored procedures were created to run during non-working hours to populate canonical models. These stored procedures were triggered using a custom Java application which was also connected to EDW using ODBC drivers. *Java application [73]* can also load data from canonical models to EDW during normal working hours without affecting Teamcenter performance.

3.2.2 Solution – II: Talend ETL tool

Talend open studio is an open source integration platform for data integration. The reason for selecting this ETL tool in this solution was the fact that it was open source and also one of the popular integration tools in the market. Idea was to utilize Talend for orchestrating loads by triggering stored procedures created in solution – I. However, soon it was found out that the open source version of Talend was not having enough functionalities to support this integration. To proceed, Wärtsilä must buy a commercial license from Talend. After discussions with enterprise architect and solution architect for analytics at Wärtsilä, it was found out that they had already invested in an enterprise license for Informatica power center. Soon after getting approval, a user was created in Informatica power center tool for development of the integration of product data to enterprise data warehouse.

3.2.3 Solution – III: Informatica ETL tool

Informatica power center is a data integration platform and market leader in providing solutions for ETL. This commercial tool is sold with a license. Another reason behind selecting this tool for solution was that it was already being used in other projects. And no additional license fee was needed. This also came as a cost-effective solution. Solution design after adding Informatica power center in the picture is shown in *figure 20*. Again, without any re-engineering Informatica would utilize existing stored procedures to load canonical models. And eventually, transfer data from the canonical model to EDW.

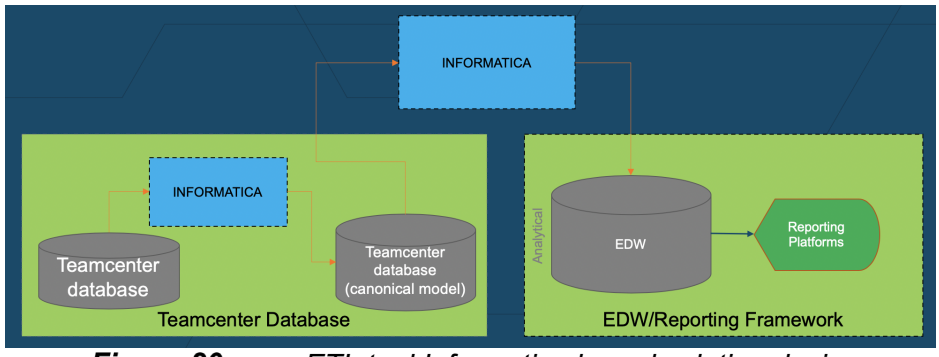

Figure 20. *ETL tool Informatica based solution design*

Informatica power center tool comes with four user interfaces for different operations: repository manager, designer, workflow manager and workflow monitor [71]. Repository manager allows managing of folder structures for collaboration in development and deployment, providing access control and other admin level properties to manage the server. Designer allows developers to create mappings for data pipelines. These mappings are then exported to the workflow manager. Workflow manager is the configuration manager for Informatica power center loads. It allows the user to set parameters for creating connections to different data applications, create dependency between workflows and also scheduling. Finally, a workflow monitor is used for maintenance of data pipelines.

3.3 Reflections

Based on these three alternative solutions, reflections were studied. Individually each evaluation criteria that were established in the beginning were revisited to compare the results:

3.3.1 Connectivity

In both Talend and Informatica power center, it was equally easy to create connections to Teamcenter and Oracle Teradata. Both Informatica power center and Talend also

provides a wide range of connectivity drivers to source and target such as flat file, Microsoft SQL server, Oracle, Excel and *cloud databases [74]*. However, in the case of custom Java application connectivity was not straightforward. Also, database credentials were stored in json files and were visible to anyone who has access to code. Informatica and Talend have more secure ways of storing credentials and are only visible to administrative users.

3.3.2 Performance

In the first solution, with Java application, data transfer rate drastically improved compared to data transfer rate in existing solutions. Existing solution was using TKI API for data extraction. The Java application was querying the Teamcenter database. This proved that extracting data directly from the Teamcenter database in batch mode is much faster compared to extracting data using Teamcenter API. In the second solution, using Talend the transfer rate was slow because the open source version of the Talend has a threshold on CPU usage and needed a license to remove that threshold. Finally, the Informatica power center data transfer rate was equally fast like in the Java application case. Although, informatica power center workflows added a delay of a few seconds in data transfer.

3.3.3 Configurability

Custom Java application was configured using a *Json file [75]*. It accepts parameters for database connection, for example, connection string and credentials. Development of functionality was limited due to the scope of application development. Talend and Informatica power center provide user interface to configure connections. These tools take configurability to a next level. It is also possible to control advanced functionalities like truncating table before loading and table lookup.

3.3.4 Operability

Test for operability was conducted by studying robustness, monitoring features, trouble-shooting features, availability, recoverability and deployment [67]. Results are summarized in the following table:

	Existing solution	Java application	Talend Integration	Informatica power center
Robustness	Frequent load failures	Few load failures	No load failure	No load failure
Monitoring features	Not available	Not available	Limited monitoring features in open source	Advanced monitoring features
Trouble-shooting	Not available	Not available	Debugging	Debugging, enterprise technical support
Availability	Availability dependent on hosting server – single cluster	Availability dependent on hosting server - single cluster	Single cluster in open source version	High availability with multi-cluster
Recoverability	Manual	Manual	Manual	Automated
Deployment	Manual	Manual	Automated	Automated

Figure 21. *Solution comparison chart for operability*

Timeline for development was a key factor that influenced these results. It is always possible to develop advanced functionalities in a custom developed tool. Although, that requires time and resources. This opens up discussion about tool development and maintenance cost.

3.3.5 Tool development and maintenance cost

Scope for each solution was limited to bring one entity from Teamcenter to data warehouse. Development cost for Java application can be calculated based on resource allocation. It was estimated that cost for development includes price for 2 developers working for 15-man days. This wasn't too much. But since the tool will have limited functionality for monitoring and maintenance eventually, this may end up in higher maintenance cost. Talend and Informatica both are commercial software products.

Software is sold usually in two models. Perpetual license where customer buys the license on time and pays for technical support yearly. Another model is subscription model where customer pays monthly or yearly license fee which generally includes technical support. Subscription license cost ranges from around 30k euros per year for Talend to

80k euros per year for Informatica power center. Benefits of investing in a commercial tool includes getting latest updates without manual patching, technical support from an expert team, plugins to connect with popular tool and faster development and focus on business rather than tool development.

Enterprises tend to invest in commercial tools because they are not a software development company. They continue to focus on development of their business instead of trying to be a software development company. The Wärtsilä analytics team had already invested in purchasing license for informatica power center and was being used in other projects. This became an advantage in continuing development using informatica power center for integrating product data to enterprise data warehouse.

All in all, to summarise on tool development and maintenance cost, commercial tools for ETL have high license but low maintenance cost. And vice versa for custom developed tool.

4. RESULTS AND ANALYSIS

This thesis discussed different approaches for integrating product data with enterprise data warehouse at Wärtsilä. Keeping action research in mind there were three alternative solutions panned. First solution included custom Java application development which extracted data from the Teamcenter database and ingested into the data warehouse. Second solution utilized Talend integration tool for orchestrating data extraction from Teamcenter and ingestion into data warehouse. And finally, the third solution utilized Informatica power center integration tool for the same. This chapter first discusses the shortcomings of the existing solution. Next it compares results from three solutions and finally, analyses the result and how Wärtsilä proceeded with implementing a solution for integrating product data to enterprise data warehouse.

4.1 Analysis from solutions

To evaluate different approaches to integrate product data to enterprise data warehouse, common evaluation criteria were established. These later became acceptance criteria for selecting the best out of studied approach: connectivity, performance, configurability, operability, tool development cost and maintenance cost.

4.1.1 Solution – I: Custom Java application

In the first solution a custom Java application was developed to execute stored procedures in the Teamcenter database for extracting data. It was observed that the extraction process took remarkably less time, compared to the approach of extracting data using Teamcenter API. Stored procedures were created using Teamcenter table information stored in ‘model.xml’ file. This file was stored in the installation directory of the server where Teamcenter was installed. Stored procedures were stored in the Teamcenter database. The Java application connects with the Teamcenter database to execute these stored procedures. In the first stage stored procedures populated a canonical model which simplified the extraction process. This canonical model was then used as a source to copy data from Teamcenter to enterprise data warehouse using the same Java application.

Custom Java tool lacked features such as monitoring, code debugging and scheduling. Tool development cost was estimated to be medium and the internal team at Wärtsilä lacked skills to maintain the solution. This would mean a high cost of maintenance.

Pros:

fast connectivity
fast loading of data
no license cost
medium development cost

Cons:

un-secure connectivity
complex configuration
no scheduling of the workflows
no load monitoring capability
high maintenance cost
less operability

4.1.2 Solution – II: Talend ETL tool

In the second solution Talend ETL tool was selected. Talend provides both commercial and open source ETL tools. For the purpose of evaluation, a free version of the tool was downloaded. In this case, Talend tool connected to the Teamcenter database and triggered existing stored procedures created in the previous solution. This populated the canonical model. Talend then copied data from the canonical model in the Teamcenter database to the enterprise data warehouse.

Pros:

fast and secure connectivity
fast loading of data

Cons:

high license cost
low maintenance cost
low operational cost
low development cost

4.1.3 Solution – III: Informatica ETL tool

Informatica power center was already being used in other projects at Wärtsilä. This already brought confidence with this tool. Wärtsilä has commercial licenses for Informatica power center. After approval, access for development was granted to practitioners for this solution. Informatica power center is a similar tool as compared with Talend. However, the licensed version of Informatica power center had additional features like version control and multi-cluster setup. This made it more available and robust.

Pros:

fast and secure connectivity
fast loading of data
built-in workflow scheduling
easy configurability
high availability
no additional licensing cost (license already purchased for other projects)

low maintenance cost
low operational cost

Cons:

internal team lacked Informatica skills

4.2 Tool Selection and solution implementation

Implementation for all solutions went successful and reflections from each solution were ready. A good sign was that all three approaches were able to serve the purpose of integrating product data with enterprise data warehouse. Another key finding was data extraction performance:

Faster rate of data extraction using Teamcenter database
Comparatively slower rate of data extraction using Teamcenter ITK APIs.

Results from each solution are summarized in the table below. For simplicity and convenience in comparison, results for each of these evaluation criteria are indicated with low, medium and high values. Where low indicating poor result and high indicating best result. In case of costs, high indicates higher cost.

It was easily evident that Informatica power center with enterprise license outperformed most of the evaluation criteria that were established in the beginning.

Evaluation Criteria / solution	Existing solution	Custom Java application	Talend ETL tool	Informatica power center
Connectivity to Teamcenter	Teamcenter ITK API	JDBC	Talend connector	Informatica connector
Connectivity to EDW	M-control	ODBC	Talend connector	Informatica connector
Performance criteria	last 7 days data	last 7 days data	last 7 days data	last 7 days data
Performance	5-7 hours	12-18 minutes	40-60 minutes	14-20 minutes
Scheduling	bash script	Windows scheduler	internal scheduler	internal scheduler
Configurability	INI files	Json files	visual tool	visual tool
Operability	low	low	high	high
Development Cost	high	medium	low	low
License Cost	M-control license cost - low	no - license	high	no additional license cost for Wärtsilä
Maintenance Cost	high	high	low	low

Figure 22. *Comparison table- based on evaluation criteria*

Finally, based on results from these solutions and with common agreement with stakeholders, Informatica power center was selected as an ETL tool for integrating product data to enterprise data warehouse. Final solution design selected for deployment [4] is

shown in *figure 22*. Next steps followed after selection of ETL tool included implementation of solution design, deployment and maintenance of the solution for continuous service.

4.2.1 Solution implementation

During the implementation of alternative solutions, scope of development was limited to one entity of Teamcenter: Item Revision. For the final implementation of the solution, development of all remaining entities had to be carried out. Teamcenter domain knowledge was within the product data management team at Wärtsilä. Therefore, development for extracting data from the Teamcenter database into a canonical model by creating stored procedures was undertaken by the author.

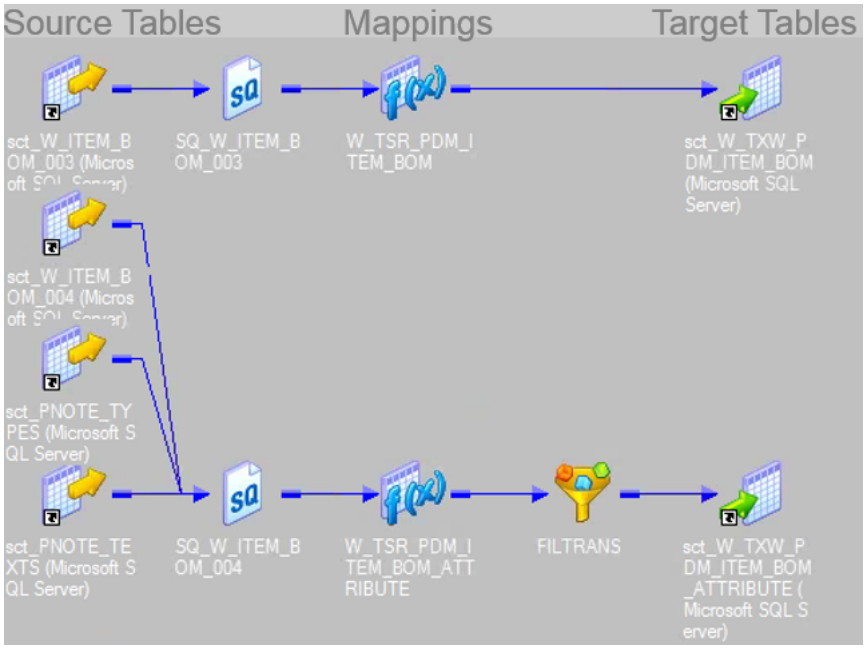

Figure 23. *Example mapping from Informatica power center*

Another tool provided by *Teamcenter (Teamcenter Walker) [15]* which allowed browsing through Teamcenter entities also played an important role in understanding Teamcenter structure of storing data. It provided information about the table and attribute names for each entity in Teamcenter. *Figure 23* shows an example of mapping using Informatica power center. Source table refers to Teamcenter database tables. Target tables are part of the canonical model created in the Teamcenter database. However, development of data transfer from canonical model in Teamcenter to enterprise data warehouse was not covered during this development.

4.2.2 Solution deployment

To carry out development for the overall data pipeline for integrating product data to enterprise data warehouse needed more developers. Wärtsilä has their *IT competence center [76]* in China. To finalize the deployment author-initiated knowledge transfer to a team of 5 developers located in Wärtsilä competence center at China. Knowledge transfer sessions were carried out remotely and finally they took over the remaining development for the solution. Analytics team at Wärtsilä also took part in these sessions to provide knowledge for data modelling using dimensional modelling methodology in the data warehouse. This ensures common ways of modelling data and allows business users to combine data coming from multiple applications to enterprise data warehouse. Additionally, deployment also had to be replicated across three different environments of Teamcenter for development, testing and production use cases.

4.2.3 Maintenance

Maintenance activities such as monitoring daily data loads, reloading failed loads and adding new changes to data pipelines was also outsourced to the same team in China. They took responsibility for maintaining data quality and server maintenance for server hosting Informatica power center.

4.2.4 Risk and Challenges

Investing in a commercial tool like Informatica power center reduces common risks and challenges. One reason for it is having a large customer base for these commercial players. They constantly gather bug reports from their customers and roll out updates to their tools. This ensures resolving common bugs without any interruption in operations. However, a common challenge with data level integration is that they are tightly coupled. Changes in database tables are rare. But still any change will require manual modifications in the data pipeline. Also, the maintenance team has to take care of any possible updates coming to Teamcenter and its impact on Teamcenter databases. Overall while performing implementation for alternative solutions, load performance was optimal and stable.

5. CONCLUSIONS

Results from solutions showed that performance of integration improved drastically. Data loading time with existing extractor took on an average 6.5 hours for daily loads. It was reduced to 14-20 minutes using Informatica power center. This improved performance for data integration by almost 95%. Applying this solution also reduced data load failures up to 99%. This had a great impact on business users who initially were struggling because of obsolete data in their reports.

While making a choice between development of a custom software tool and investing in a commercial tool, key parameters are cost and timeline. Usually, the cost of internal development of a tool is cost effective compared to paying for licenses. However, it requires the right set of skills developers and development time. On the other hand, commercial tools allow businesses to focus on building solutions quicker. This helps with faster time to market their product and services. Another advantage of investing in an enterprise level tool is to maintain a common approach to similar problems within a company. Having an enterprise tool also enables visibility data lineage. In companies like Wärtsilä where core business is not software development it makes sense to invest in a commercial tool instead of reinventing the wheel. This takes away also the overhead of support and maintenance of the tool. And employees can focus on developing their core business and accelerate their growth.

REFERENCES

[1] Digitalization, in: The International Encyclopedia of Communication Theory and Philosophy, 2016, pp. 1-11.

[2] Value-added services — problems of definition and data, in: Telecommunications Policy, 1992, pp. 388-400.

[3] J. Babicz, Encyclopedia of Ship Technology, WÄRTSILÄ Corporation, 2015.

[4] S. Wirts, G. Albelli, E. Lazarus, Software application development methods and framework, 2006.

[5] D. Linstedt, Method and system of data warehousing and building business intelligence using a data storage model, 2002.

[6] D.L. Moody, M.A. Kortink, From enterprise models to dimensional models: a methodology for data warehouse and data mart design. pp. 5.

[7] I. Crnkovic, U. Asklund, A.P. Dahlqvist, Implementing and integrating product data management and software configuration management, Artech House, 2003.

[8] J. Kääriäinen, Product data management (PDM): design, exchange and integration viewpoints, Technical Research Centre of Finland, Espoo, 2000.

[9] B. Berenbach, The other skills of the software architect, pp. 7-12.

[10] G. Sparks, Enterprise architect user guide, 2009.

[11] Product life cycle, in: A Dictionary of Economics, 5th ed., Oxford University Press, 2017.

[12] J. Antchak, J.W. Dell, B. Mevissen, Crankshaft torque modulator, 2009.

[13] G. Damiand, Topological model for 3d image representation: Definition and incremental extraction algorithm, Computer Vision and Image Understanding, Vol. 109, Iss. 3, 2008, pp. 260-289.

[14] G.R. Klaus, Flat file, 1982.

[15] Siemens PLM Software's Teamcenter, Aircraft Engineering and Aerospace Technology, Vol. 81, Iss. 2, 2009.

[16] A. Sangiovanni-Vincentelli, G. Martin, Platform-based design and software design methodology for embedded systems, IEEE Design & Test of Computers, Vol. 18, Iss. 6, 2001, pp. 23-33.

[17] Wärtsilä Oyj Abp, Mergent, Fort Mill, 2017.

[18] R. Sharda, D. Delen, E. Turban, Business intelligence: a managerial perspective on analytics, Prentice Hall Press, 2013.

[19] E.J. Umble, R.R. Haft, M.M. Umble, Enterprise resource planning: Implementation procedures and critical success factors, European Journal of Operational Research, Vol. 146, Iss. 2, 2003, pp. 241-257.

[20] T. Matsumoto, T. Saito, Y. Yamashita, Customer relationship management system, 2003.

[21] S. Nastic, S. Sehic, D. Le, H. Truong, S. Dustdar, Provisioning software defined IoT cloud systems, IEEE, pp. 288-295.

[22] R. Dejan, An integration strategy for large enterprises, Yugoslav Journal of Operations Research, Vol. 17, Iss. 2, 2007, pp. 209-222.

[23] J. Mundy, W. Thornthwaite, The Microsoft data warehouse toolkit: with SQL Server 2008 R2 and the Microsoft Business Intelligence toolset, John Wiley & Sons, 2011.

[24] M. García, B. Harmsen, Qlikview 11 for developers, Packt Publishing Ltd, 2012.

[25] T. Lachev, E. Price, Applied Microsoft Power BI: Bring your data to life! Prologika Press, 2018.

[26] S. Gautam, I. Books24x7, IBM Cognos Business Intelligence v10: The Complete Guide, First ed. IBM Press, Upper Saddle River, N.J, 2013.

[27] M. Masse, REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces, " O'Reilly Media, Inc.", 2011.

[28] C. Supaartagorn, PHP Framework for Database Management Based on MVC Pattern, International Journal of Computer Science & Information Technology, Vol. 3, Iss. 2, 2011, pp. 251-258.

[29] K. Loney, Oracle database 10g: the complete reference, Osborne, New York, NY, 2004.

[30] P. Turley, R.M. Bruckner, Microsoft SQL Server reporting services recipes: for designing expert reports, Wrox Press Ltd., 2010.

[31] M. Bernardo, A. Simon, J.J. Tarí, J.F. Molina-Azorín, Benefits of management systems integration: a literature review, Journal of Cleaner Production, Vol. 94, 2015, pp. 260-267.

[32] A. Savinov, Concept-Oriented Model: Extending Objects with Identity, Hierarchies and Semantics, Computer Science Journal of Moldova, Vol. 19, Iss. 3, 2012, pp. 254-287.

[33] Object-relational mapping, in: A Dictionary of Computer Science, 7th ed., Oxford University Press, 2016.

[34] V. Gecevska, T. Stojanova, B. Jovanovski, PRODUCT LIFECYCLE MANAGEMENT TOOLS, Annals of the Faculty of Engineering Hunedoara, Vol. 11, Iss. 1, 2013, pp. 219.

[35] Relational Database, in: Dictionary of Information Science and Technology, 2013.

[36] Z. Parker, S. Poe, S.V. Vrbsky, Comparing nosql mongodb to a sql db, pp. 1-6.

[37] M. Golfarelli, D. Maio, S. Rizzi, Conceptual design of data warehouses from E/R schemes, Proceedings of the Thirty-First Hawaii International Conference on System Sciences, pp. 334-343 vol.7.

[38] R. Kimball, M. Ross, The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modeling, Third; 3; 3rd ed. John Wiley & Sons Ltd, US, 2013.

[39] R. Kimball, M. Ross, The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2. Aufl.; 2; 2nd ed. Wiley, Hoboken, 2002.

[40] K. Chari, S. Seshadri, Demystifying integration, Communications of the ACM, Vol. 47, Iss. 7, 2004, pp. 58-63.

[41] M. Vujasinovic, Z. Marjanovic, Data Level Enterprise Applications Integration, in: Anonymous (ed.), Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 390-395.

[42] J. ZHAN, F. LIU, Implementation and Design of Material Transfer Server Based on FTP [J], Computer Technology and Development, Vol. 3, 2008.

[43] G. Wilson, D.L. Deitz, W.G. Irwin, G.R. Sherriff, Distributed batch processing system and methods, 2001.

[44] F.A. Ware, E.K. Tsern, I.P. Shaeffer, Point-to-point connection topology for stacked devices, 2010.

[45] O.M. Gadir, K. Subbanna, A.R. Vayyala, H. Shanmugam, A.P. Bodas, T.K. Tripathy, R.S. Indurkar, K.H. Rao, High-availability cluster virtual server system, 2005.

[46] B.A. Christudas, I. Books24x7, Service Oriented Java Business Integration: Enterprise Service Bus Integration Solutions for Java Developers, 1st ed. Packt Publishing, Olton, 2008.

[47] P. Dhanda, N. Sharma, Extract Transform Load Data with ETL Tools, International Journal of Advanced Research in Computer Science, Vol. 7, Iss. 3, 2016.

[48] L. Aldred, W. van der Aalst, M. Dumas, A. ter Hofstede, Understanding the challenges in getting together: The semantics of decoupling in middleware, BPM Center Report BPM-06-19, BPMcenter.org, 2006.

[49] K. Yusuf, Enterprise messaging using JMS and IBM websphere, Prentice Hall Professional, 2004.

[50] T. Rademakers, J. Dirksen, Open-source ESBs in action, Manning Publications Co., 2008.

[51] I.M.T. Sandford, T.G. Handel, Data validation, 2000.

[52] E. Rahm, H.H. Do, Data cleaning: Problems and current approaches, IEEE Data Eng.Bull., Vol. 23, Iss. 4, 2000, pp. 3-13.

[53] P. Jesus, C. Baquero, P.S. Almeida, A survey of distributed data aggregation algorithms, IEEE Communications Surveys & Tutorials, Vol. 17, Iss. 1, 2014, pp. 381-404.

[54] J. Schiefer, J. Jeng, R.M. Bruckner, Real-time workflow audit data integration into data warehouse systems. pp. 1697-1706.

[55] J. McNiff, Action research: Principles and practice, Routledge, 2013.

[56] C. Wohlin, M. Höst, K. Henningsson, Empirical research methods in software engineering, in: Anonymous (ed.), Empirical methods and studies in software engineering, Springer, 2003, pp. 7-23.

[57] S. Sukamolson, Fundamentals of quantitative research, Language Institute Chulalongkorn University, Vol. 1, 2007, pp. 2-3.

[58] J.H. Frey, A. Fontana, The group interview in social research, The Social Science Journal, Vol. 28, Iss. 2, 1991, pp. 175-187.

[59] M.O. Schafer, M. Melich, SAP solution manager, SAP PRESS, 2007.

[60] Introducing Dimensional Data Modeling, in: Anonymous (ed.), Second ed., Apress, Berkeley, CA, 2005, pp. 377-420.

[61] D.M. Ritchie, B.W. Kernighan, M.E. Lesk, The C programming language, Prentice Hall Englewood Cliffs, 1988.

[62] Q. Ding, BMC Control-M 7: A Journey from Traditional Batch Scheduling to Workload Automation, Packt Publishing Ltd, 2012.

[63] V. Wadhwa, A.R. Ravindran, Vendor selection in outsourcing, Computers & Operations Research, Vol. 34, Iss. 12, 2007, pp. 3725-3737.

[64] D. Duda, J.T. Pascoe, W. Matyjewicz, K. Maziarz, Method and apparatus for analyzing and migrating data integration applications, 2012.

[65] V. Ranjan, A comparative study between ETL (Extract, Transform, Load) and ELT (Extract, Load and Transform) approach for loading data into data warehouse, 2009.

[66] M. Mcsi, Configurability in SaaS (software as a service) applications, pp. 19-26.

[67] C. Ford, I. Gileadi, S. Purba, M. Moerman, Patterns for performance and operability: building and testing enterprise software, CRC Press, 2007.

[68] T. Heninger, R. Rasmussen, Server-side scripting language and programming tool, 2002.

[69] E. Ries, Minimum viable product: a guide, Startup lessons learned, 2009.

[70] R. Katragadda, S.S. Tirumala, D. Nandigam, ETL tools for data warehousing: an empirical study of open source Talend Studio versus Microsoft SSIS, 2015.

[71] J. Levin, ETL Tools Comparison, 2008.

[72] Z. Xianghui, L. Xiaoda, Methods of Mapping Model Data to Teamcenter, IEEE, pp. 783-786.

[73] V. Kodaganallur, Incorporating language processing into java applications: A javacc tutorial, IEEE Software, Vol. 21, Iss. 4, 2004, pp. 70-77.

[74] W. Al Shehri, Cloud database database as a service, International Journal of Database Management Systems, Vol. 5, Iss. 2, 2013, pp. 1.

[75] K. Maeda, Performance evaluation of object serialization libraries in XML, JSON and binary formats, IEEE, pp. 177-182.

[76] E. Müller, H. Hopf, Competence center for the digital transformation in small and medium-sized enterprises, Procedia Manufacturing, Vol. 11, 2017, pp. 1495-1500.

[77] M. Dietrich, J. Lemcke, A Refined Canonical Data Model for Multi-schema Integration and Mapping, 2011 IEEE 8th International Conference on e-Business Engineering, pp. 105-110.