TECHNOLOGY AND TRANSPORT

Industrial Management

BACHELOR'S THESIS

INTEGRATING MASTER DATA APPLICATION IN A GLOBAL ENTERPRISE

Author: Juhana Murtola
Supervisor: Juhani Rajamäki
Instructor: Mikko Strömberg

Approved: November 26, 2009

Juhani Rajamäki
Senior Lecturer

ABSTRACT

Name: Juhana Murtola

Title: Integrating Master Data Application in a Global Enterprise

Date: November 26, 2009 **Number of pages:** 65 p. + app. 17 p.

Degree Programme:
Industrial Management

Instructor: Juhani Rajamäki, Senior Lecturer
Supervisor: Mikko Strömberg, IT Manager, Konecranes

An increasing number of applications are used by organizations to support their business processes. All these enterprise applications store similar information regarding business objects, such as customers or suppliers. The information may thus become fragmented as it is distributed across the organization. Therefore separate applications need to be integrated to share information with each other. Properly functioning integrations enable comprising a unified view of the core business objects, the master data.

Integrating stand-alone applications is usually not a simple task as they may share nothing in common. This study introduces different ways of approaching such a complicated issue. The ultimate goal was to create instructions on how to integrate the master data application of Konecranes with other enterprise applications within the company. The final outcome is a step by step roadmap that can be applied to suit varying integration projects.

Instead of just considering the concrete implementation of integrations, this study also discusses the gained benefits. It is proved how integrations contribute to the platform architecture of Konecranes. Another important aspect is explaining the relationship between business reporting and master data management. All in all, this study provides a glance at how integrations can be utilized to rationalize enterprise architecture.

Keywords: Enterprise Application Integration, middleware, Master Data Management, Business Intelligence

TIIVISTELMÄ

Työn tekijä: Juhana Murtola

Työn nimi: Master data – järjestelmän integrointi globaalissa yrityksessä

Päivämäärä: 26.11.2009 **Sivumäärä:** 65 s. + 17 s. liitteitä

Koulutusohjelma:
Tuotantotalous

Työn valvoja: lehtori Juhani Rajamäki
Työn ohjaaja: IT-päällikkö Mikko Strömberg, Konecranes

Nykypäivän organisaatioissa on käytössä yhä kasvava määrä erilaisia järjestelmiä liiketoimintaprosessien tarpeisiin. Näissä järjestelmissä ylläpidetään hyvin samankaltaista liikekumppaneihin liittyvää tietoa. Tämä tieto on kuitenkin usein puutteellista ja se on jakautunut epätasaisesti ympäri organisaatiota. Siksi erillisten järjestelmien on kyettävä jakamaan tietoa keskenään. Integraatioiden avulla näiden järjestelmien sisältämästä tiedosta on mahdollista koostaa yhdenmukainen näkymä, jota kutsutaan master dataksi.

Itsenäiset järjestelmät voivat olla keskenään hyvin erilaisia, joten niiden integrointi on usein vaikeata. Tämä tutkimus esittelee erilaisia lähestymistapoja tähän aiheeseen. Päätavoitteena oli luoda suunnitelma Konecranesin master data – järjestelmän integroimiseksi muiden yhtiössä käytössä olevien järjestelmien kanssa. Lopputuloksena on yleispätevä ohjeistus, jossa käydään askel askeleelta läpi integraatioprojektin vaiheet.

Tämä tutkimus käsittelee järjestelmäintegraatioiden käytännön toteutustapojen lisäksi myös niistä saavutettavia hyötyjä. Tutkimuksessa osoitetaan, miten Konecranesin arkkitehtuurimallissa hyödynnetään integraatioita. Toisekseen siitä käy ilmi, miten liiketoimintaraportointi ja master datan hallinta ovat erottamaton osa toisiaan. Kaiken kaikkiaan tutkimus tarjoaa katsauksen integraatioiden hyödyntämisestä organisaatioiden kokonaisarkkitehtuurissa.

Avainsanat: järjestelmäintegraatio, väliohjelmistot, master datan hallinta, liiketoimintatiedon hallinta

TABLE OF CONTENTS

TABLE OF FIGURES

INTRODUCTION

Large organizations can have a number of applications meant to fulfil certain business needs. There may be their own distinct systems for supply chain management, customer relationship management, human resources, and master data management, just to point out a few examples. Typically, it is not appropriate to replace them with just one major enterprise resource planning software. Still, stand-alone applications should be able to share information with other parties. This study discusses application integrations that enable separate systems to communicate with each other.

1.1 Background

The information of applications in large organizations is often not synchronized and is distributed across the organization. *Master Data Management* (MDM) refers to the management of core information, such as information about customers, suppliers, or products. MDM aims to bring this information together so that all details about customer A or product B, for example, can be found in one place. MDM has become increasingly popular as it enables the organization to understand business entities in a more complete and holistic manner.

Data integration is a fundamental requirement for any successful MDM implementation, both for moving the data and ensuring its quality. Data integration consists of several technologies. These technologies do basically two types of things; they either move information from one place to another or they assure the validity of the data. This paper is concentrated on the first-mentioned aspect.

Enterprise Application Integration (EAI) is the integration of various applications so that they are able to share information and processes. EAI supports moving information by allowing applications to communicate with each other using standard interfaces. From the MDM perspective, EAI can be used to gather data from several applications and merge it into a single database, a master data repository. This paper mainly focuses on EAI itself.

1.2 Objectives

The main objective for this study is to create a roadmap for an application integration project. It should include specific instructions on how to integrate Konecranes' master data application *Global Company Master* (GCM) with any enterprise application that is used within the company. GCM is used for storing and distributing master data across the organization. In practice, the integration roadmap describes a set of steps that needs to be taken during a well-designed integration project.

In addition to the main objective, there are also few secondary objectives set for this paper. It is supposed to give an overview of application integration and especially master data application integration. A lot of research has been done regarding application integration in general but there is not too much information on what the special characteristics of a master data application integration project are. Another objective is to represent different patterns and technologies to approach application integration with. It is possible to reconsider and enhance the current integration environment on the basis of that research. It may appear that some things could be done more efficiently. Understanding alternative solutions makes it easier to design the whole integration architecture. The third secondary objective is to give a high-level representation of the current platform architecture of Konecranes.

1.3 Scope

In most sources, the term EAI is used when talking about organizations' internal application integrations. Sometimes, however, the A2A (Application-to-Application) integration is referring to the same issue. This is to separate internal integrations from B2B (Business-to-Business) integrations. B2B integration means that organization's application is integrated with a third-party application in order to share information across organizational boundaries.

The study is focused on internal application integration. As the main goal is to create integration roadmap for Konecranes' internal master data solution, there is no need to examine the special features of B2B integrations in this

context. There are many similarities between EAI and B2B integration methods but this study is written from EAI's point of view. From now on, EAI is the abbreviation that is used in this study to describe application integrations.

The theory of this paper emphasizes in introducing general integration design patterns and approaches, as well as different models and types of middleware. It is also utterly important to understand the basics of MDM to see why it is applied in the majority of large enterprises. Together with *Data Warehousing* (DW), MDM is the basis of *Business Intelligence* (BI) solutions. Bundling up all those terms proves why integrations are built in the first place.

Technical details are not included in the theory sections of this study. A closer look is taken at some integration tools and technologies but integrations are mainly examined at more general level. When illustrating Konecranes' integration environment, technical elements come up more clearly. The purpose is to write documentation on how the system environment has been implemented and what products have been used.

1.4 Structure

Sections 2 and 3 form the theoretical basis of this study. These two sections include information about all buzzwords that are under discussion throughout the paper and link them together tightly. Section two represents the principles of BI and different ways of implementing it. When utilized correctly, BI has several business benefits. To achieve those benefits, one must know how to gather and store data and how to assure its quality. That is why DW and MDM are important parts of BI and thereby this study.

Section 3 is the most central part of theory in this study. EAI enables organizations' stand-alone applications to share information with each other. First, EAI needs to be defined to understand what it actually stands for. When the meaning is clear, it is investigated why EAI is applied in many organizations, what is required to make it work, and what are the challenges it addresses. There are several approaches to EAI. Section 3 focuses on introducing and comparing those approaches. It is not always obvious how to start building

integration architecture. Thus, the characteristics of different EAI approaches need to need to be known. In addition to the theory of EAI, section 3 also concentrates in middleware, the software that facilitates the requests between integrated applications. Middleware models and different types of middleware are introduced to be familiar with the technology that enables EAI. Middleware selection is an essential part of application integration design and cannot be bypassed.

Sections 4 and 5 form the empirical part of the study. In section 4, the current platform architecture of Konecranes is illustrated. It is examined what layers belong to it and how they are connected to the other ones. Other important questions are what Konecranes expects to gain with its MDM solution and why it is integrated with other applications. It is also assessed what are the EAI approaches that Konecranes is using and how the middleware models fit into the big picture. Section 5 is the main output of this study. It depicts the exact steps that need to be taken in order to integrate GCM with any enterprise application within Konecranes application portfolio. Even though the roadmap is universally applicable, it still aims for describing the steps in detail.

Section 6 is the summary of the study. It summarizes the project and evaluates how the contents match with what was planned in the beginning. The outcome of the study is matched against the main objective and the secondary objectives.

1.5 Client

Konecranes is a Finnish company specialized in the operations of the manufacture and service of cranes. It is a world-leading lifting equipment manufacturer that serves mostly manufacturing and process industries as well as shipyards and harbours. Nowadays, Konecranes has production facilities and sales and service locations in 43 countries. (Konecranes Corporation 2009: 6) Its growth has been fast and several acquisitions have been made in recent history. The new organizations have been tried to absorb into the parent

company as well as possible. Still, some heterogeneity remains in both business processes and applications.

The company consists of three business units: Service, Standard Lifting, and Heavy Lifting. (Konecranes Corporation 2009: 6-7) All business units have their own system architecture and there is huge variety of applications in use. Lots of effort has been made to standardize the system structure within the enterprise and common master data repository is one step towards the correct direction. To be able to create a consistent view of the business entities and to attain certain data quality level, all applications have to be integrated with the master data repository.

This study is written while working in the Integration Services team of Konecranes Group IT unit. The unit acts in the subordination of Konecranes Headquarters as a common IT service support provider for the whole corporation. The business units also have their own IT teams and Group IT works hand in glove with them.

2 BUSINESS INTELLIGENCE

Many organizations have faced a situation where basic enterprise applications are not able to fulfil demanding information analysis and reporting needs. There is a huge amount of data available which has to be controlled to make the most out of it in decision-making. The data of enterprise applications is stored behind the boundaries of business units and applications. The solution is to gather all data into a separate centralized database where various analysis and reporting tools can utilize it. (Hovi et al. 2009: IX-XI)

BI systems are designed to help organizations understand their operations and key business measurements. This information is used to make decisions on organizational direction. (Oracle Corporation 2008: 1) There are three foundations to a complete BI solution. The first one is the MDM solution for ensuring that quality data under enterprise applications and hierarchies is supplied to the data warehouse. Secondly, there is the data warehouse itself for holding the operational history. The third foundation is formed by BI tools that utilize the data warehouse and the master data repository to get clean authoritative information to everyone in the organization that needs it. (Oracle Corporation 2008: 12)

This section introduces the different types of data that an organization holds to explain why different methods are needed to control them. The roles of MDM and DW are also discussed and especially how they connect with the complete BI solution.

2.1 Enterprise Data

An organization has three types of actual business data: master data, transactional data, and analytical data. The *master data* represents the business objects that are shared across several enterprise applications. The transactions are executed around these business objects. Master data does not include any information regarding transactions. It consists solely of basic information such as company name, address, VAT number, and phone number. (Oracle Corporation 2008: 1-2)

An organization's operations are supported by applications that automate key business processes. These include areas such as sales, order management, manufacturing, and purchasing. The enterprise applications require enormous amount of data to function properly. In addition to the data about the objects that are involved in transactions, the applications also need transaction data itself. For example, the *transactional data* can be the time, place, and price of a sale transaction. (Oracle Corporation 2008: 1-2)

The *analytical data* is used to support the organization's decision-making. Customer buying patterns are analyzed and suppliers are categorized, based on analytical data. This data is stored in large data warehouses that are designed to support heavy queries. (Oracle Corporation 2008: 1-2)

2.2 Master Data Management

The master data is some of the most valuable information in an organization. It represents core information about the business; customers, suppliers, and products, for example. However, the master data is often kept in many overlapping systems and it lacks of quality. Fixing poor data quality at its source and managing constant change is what MDM is all about. MDM is a modern method to eliminate poor data quality under heterogeneous IT application landscapes. MDM refers to the disciplines, technologies and solutions that are used to create and maintain consistent and accurate master data for all stakeholders across the organization. (Dreibelbis et al.:2008) In figure 1, master data repository distributes master data to connected parties.

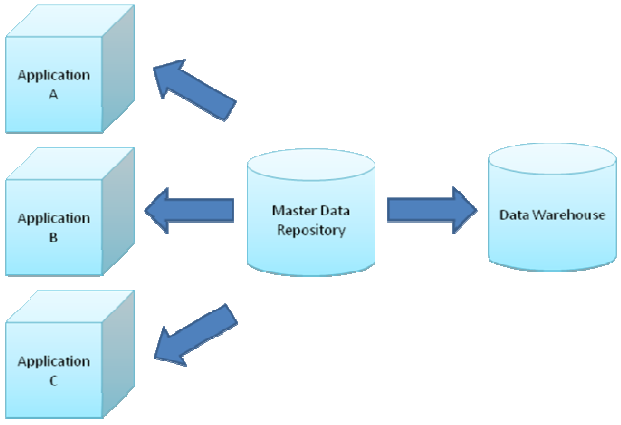

Figure 1. Master data distribution.

The MDM data model is unique in that it represents a superset of all ways master data has been defined by all attached applications. It holds all necessary hierarchical information, all attributes needed for duplicate removal and prevention, as well as cross-reference information for all attached enterprise applications. Hierarchy information is invaluable for proper rollup of aggregate information in the BI tools. MDM holds the official hierarchy information used by the enterprise applications. An important part of an MDM solution is mechanism for finding duplicate records. A primary technique is to configure a rules engine to find potential matches using a large number of attributes. MDM also holds the organizational cross-reference information for enterprise applications. It maintains the ID of every connected system and attaches them to the ID of the particular master data object. When the data warehouse uses the master cross-reference data, it correctly combines the separate entries for accurate reconciliation. This is the key for accurate reporting and analysis. If the data is not recognized as the same entity to the BI tools, it can lead to misleading results. (Oracle Corporation 2008: 4-7)

2.3 Data Warehousing

In many organizations, the central problem of data management is that the data is fragmented across enterprise applications. Also, the data may not have been mapped and people are not fully aware of what each field in a database consist of. It is usual that there is not any common data model of the data content of different enterprise applications. The data fragmentation makes it difficult to create reports and analyses on the basis of the data in several separate applications. They are usually not easily connectable with each other and the general view is not clear enough. (Hovi et al. 2009: 5-6)

To get full advantage of the data in enterprise applications, it is necessary to have an own separate database for reporting and analysis purposes. This database is called data warehouse. It supports BI tools especially well if the database is particularly designed and built only for this use. Getting detailed re-

ports out of the raw data in enterprise applications requires well-tuned refining chain as illustrated in figure 2. (Hovi et al. 2009:14-15)

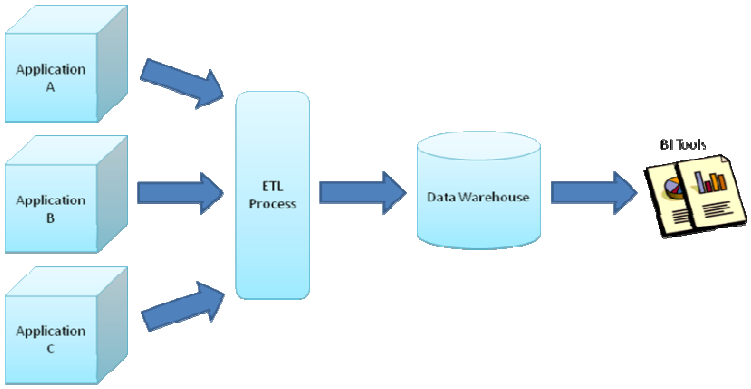

Figure 2. Data refining chain.

The first step is to modify the extract of raw data from enterprise applications, transform it into a suitable format, and load it into a data warehouse. This procedure is called an Extract, Transform, Load (ETL) process. When the data is in the data warehouse in consistent format, the BI tools are able to create reports and analyses on its basis. (Hovi et al. 2009:14-15)

2.4 Application Integration and Business Intelligence

A comprehensive and unified operating mode is the main target of a BI system. That is why enterprise applications cannot be used in isolation. Building a corporate-wide EAI infrastructure requires the integration of many different enterprise applications. EAI is essentially the ability to communicate with all the applications and data sources across the organization. Such integration supports unified views of information and lets end-users update information in real-time across systems. Decision-makers can view information at a global level being sure that one application's information is synchronized with the rest of the organization's applications. Using EAI as the layer of glue attached to each application provides an interface from each application to an external integration system. This approach guarantees the appropriate information to be forwarded to the BI system. (Thierauf 2001: 157-158)

Applications should not only be able to react to their environment but also to affect their environment in a proactive way. These applications within a BI

system should help decision-makers change the ways of working and to reinvent the organization if necessary. For enterprise applications, this means being able to share information in real-time. Solutions must give decision-makers the capability to analyze information and draw conclusions based on this information. As organizations use EAI technologies to link previously stand-alone applications the opportunity to implement real-time capabilities increases significantly. The real-time response provides decisions-makers with a better understanding and insight into their operations. (Thierauf 2001: 159)

3 ENTERPRISE APPLICATION INTEGRATION

Many organizations are using an increasing number of applications and services to solve specific business problems. Usually, these applications and services have been built over a long period of time to face new business needs that were identified. Consequently, they probably were written by different people using different languages and technologies, reside on different hardware platforms, use different operating systems, and provide very different functionality. Now, organizations are facing the challenge of providing a method by which these applications can work together to address business goals that are constantly evolving. Many applications may have very little in common at all, resulting in isolated functionality and multiple instances of the same data. (Microsoft Corporation 2003: 1-3)

3.1 Application Integration Approaches

Application integration is a combination of problems. Each organization has its own set of integration issues that must be solved. Thus, it is often difficult to find a single technological solution set that can be applied universally. That is why each application integration solution requires different approaches.

Approaches to application integration vary significantly but some general categories can be defined: Information-Oriented Application Integration, Business Process Integration-Oriented Application Integration, Service-Oriented Application Integration, and Portal-Oriented Application Integration. This subsection concentrates on these four application integration approaches.

3.1.1 Information-Oriented Application Integration

Most application integration projects are focused on *Information-Oriented Application Integration* (IOAI). It is the basis of application integration as it provides a simple mechanism to exchange information between two or more systems. IOAI allows information to move between source and target systems. The data could come for example from a database, an Application Pro-

gramming Interface (API) or a peripheral device. It is important to understand that IOAI deals with simple information instead of processes or application services. (Linthicum 2003: 25)

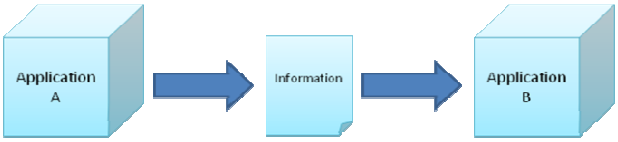

Figure 3. Information-Oriented Application Integration.

Figure 3 illustrates the basic model of IOAI. Data is simply transferred from one application to another. The information-oriented approach is the correct solution in many cases. Accessing information within databases and applications is a relatively easy task. It can be done with few changes to the application logic or database structure which is a major asset. Even though IOAI is quite straightforward it is not always that simple. Migrating data from one system to another requires detail understanding of all integrated systems, and application semantics make this problem even more complex. The semantics in one system are not usually compatible with other systems and sometimes they are so different that the systems cannot understand each other. That is why IOAI is not just about moving information between data stores, but also managing the differences in schema and content. (Linthicum 2003: 26) There are three main types of IOAI; data replication, data federation, and interface processing. They are introduced the next.

Data replication means simply moving data between two or more databases as shown in figure 4.

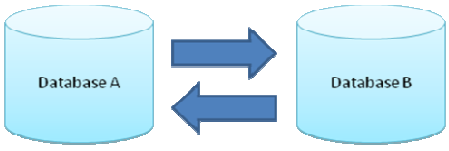

Figure 4. Data replication.

Figure 4 shows that data replication is the simple exchange of information between databases. The basic requirement of database replication is to be able to handle the differences between database models and schemas by pro-

viding the infrastructure to exchange data. The advantages of data replication are simplicity and low cost. It is easy to implement and the technology is cheap to purchase and install. However, data replication does not suit to such environment where methods need to be bound to the data or if methods are shared along with the data. (Linthicum 2000: 28-29)

In Figure 5, data federation means the integration of multiple databases and database models into a unified view of the databases.

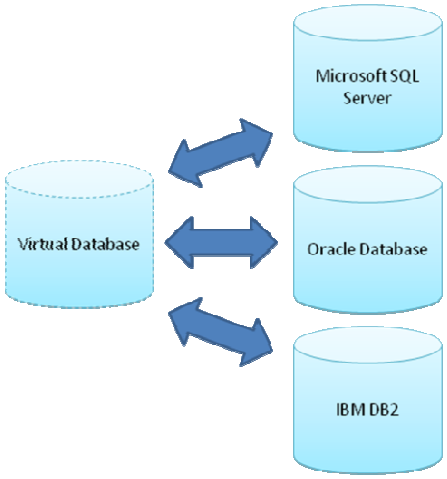

Figure 5. Data federation.

Data federations are certain kind of virtual enterprise databases that are comprised of many physical databases. The advantage of using data moderation software is that it can bind many different data types into a single model that supports information exchange. It allows access to any connected database in the organization through just one interface. (Linthicum 2000: 29-30)

Interface processing solutions use application interfaces to focus on the integration of both packaged and custom applications. In other words, interface processing externalizes information out of applications into an application integration engine, such as integration broker for example as illustrated in figure 6.

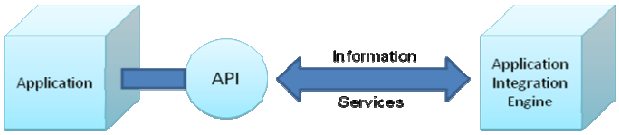

Figure 6. Interface processing.

The main advantage of using application interface-oriented products is the efficient integration of many different types of applications. However, there is little regard for business logic and methods within the source or target systems. Those may be relevant to a particular integration issue and application interface-oriented integration is not the correct option in such case. (Linthicum 2003: 9)

3.1.2 Business Process Integration-Oriented Application Integration

Business Process Integration (BPI) is the mechanism of managing the invocation of processes in the proper order. It supports the management and execution of common processes that exist between applications.

Business Process Integration-Oriented Application Integration (BPIOAI) introduces another layer of centrally managed processes. The layer is set on the top of an existing process and data contained within a set of applications as illustrated in figure 7.

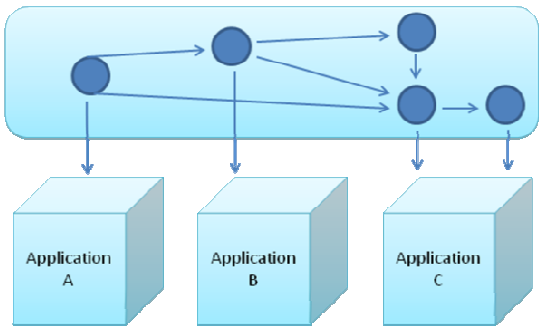

Figure 7. Business Process Integration-Oriented Application Integration.

In brief, BPIOAI is the ability to define a common business process model. The model can address the sequence, hierarchy, events, execution logic, and information movement between systems in the same organization. The idea

of BPIOAI is to provide a single logical model that covers many applications and data stores. (Linthicum 2003: 10-12)

BPIOAI is a strategy as much as a technology. It increases organization's ability to interact with any number of systems by integrating entire business processes within the organization. It is important for BPIOAI technology to be flexible as it deals with many systems using various metadata, platforms, and processes. Moreover, the BPIOAI must be able to work with several types of technologies and interface patterns. (Microsoft Corporation 2003: 18-19)

The use of common process model that spans multiple systems in the above mentioned domains can provide many advantages such as modelling, monitoring, optimization, and abstraction. *Modelling* means the ability to create common process between computer systems. It enables all information systems to react in real time to business events. *Monitoring* allows analyzing all aspects of the business and organization to determine the current state of the process. *Optimization* is the ability to redefine the process at any time in support of the business. The goal of optimization is to make the process more efficient. *Abstraction* hides the complexities of the enterprise applications from the business users. Business users are then more easily able to work with the common set of business semantics. (Linthicum 2003: 64-65)

3.1.3 Service-Oriented Application Integration

When using an application service, common business logic or methods are utilized instead of simply extracting or publishing information to a remote system. This application service is usually abstracted into another application known as a composite application as shown in figure 8.

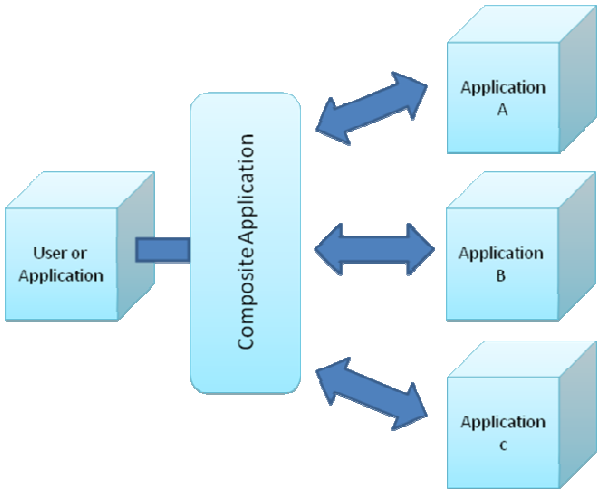

Figure 8. Service-Oriented Application Integration.

Service-Oriented Application Integration (SOAI) allows organizations share not just information, but also common application services. This sharing is accomplished either by defining shared application services or by providing the infrastructure for such sharing. Application services can be shared by hosting them on a central server or by accessing them inter-application. The goal of SOAI is a composite application made up of many application services. (Linthicum 2003: 16-18)

A common set of methods among organization invites reusability that reduces the need for overlapping methods and applications. Utilizing the tools of application integration enables sharing those common methods. Thus, the applications are integrated so that information can be shared while providing infrastructure for the reuse of business logic. The downside of SOAI is its expensiveness. As well as changing application logic, there is the need to test, integrate, and redeploy the application within the organization. Before choosing SOAI instead of IOAI for example, organizations must clearly understand the opportunities and risks. (White 2005: 19-20)

3.1.4 Portal-Oriented Application Integration

Many end-users have to access more than one system to answer a specific question or to perform a single business function. Portals aggregate information from multiple sources into a single user interface or application. (Hohpe and Woolf 2003: 5-6) The idea is illustrated in figure 9.

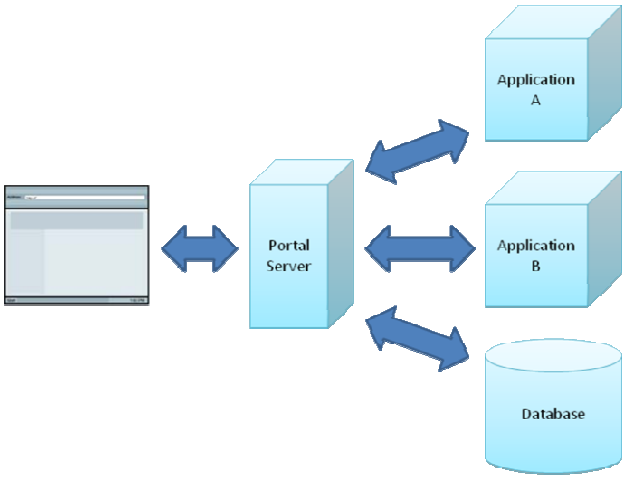

Figure 9. Portal-Oriented Application Integration.

Portal-Oriented Application Integration (POAI) can be a very effective way of integrating applications. It allows multiple applications to be presented as a single cohesive application, often using existing application user interfaces. (Microsoft Corporation 2003: 23) POAI avoids the back-end integration problem by extending the user interface of each system to a common user interface, typically a web browser. Thus, it does not directly integrate the applications or databases within organization. (Linthicum 2003: 99)

The use of portals to integrate applications has many advantages. Back-end systems do not have to be directly integrated which decreases the associated costs and risks. POAI is usually faster to implement than real-time information exchange between back-end systems. The technology that enables POAI is mature enough and has proved to be reliable. There are also lots of case examples available to learn from existing solutions. POAI also has its disadvantages. Information does not flow in real time and it requires human interaction. It means that systems do not automatically react to business events. Another POAI related problem is that information must be abstracted through a new application logic layer which adds complexity to the solution. The extra layer can also turn out to be a performance bottleneck. POAI may have problems with security issues too when organization data is being extended to users over the web. (Linthicum 2003: 19-22)

3.2 Middleware and Middleware Models

The previous section concentrated on different types of EAI approaches. The next sections are devoted to middleware, the technology that makes EAI possible. Ruh, Maginnis, and Brown (2000: 52) define *middleware* as 'a type of software that facilitates the communication of requests between software components through the use of defined interfaces or messages'. Middleware also provides the environment to manage the requests between those software components.

Middleware has certain advantages that have made its use popular when implementing EAI. It is able to hide complexities of the source and target systems. For example, the use of common middleware API hides the details of APIs, network protocols, and platforms of both the target and the source system. Middleware can also serve as an additional layer of security and data integrity to the data transfer across the organization. (Linthicum 2000: 119-120)

Middleware models can be categorized into two types: logical and physical. Those two models and their divisions are discussed the next subsection.

3.2.1 Logical Models

The logical middleware model depicts the concept of how the information moves throughout the organization. Understanding the content of the logical model requires comparing point-to-point middleware to integration hub type of middleware. The communication models need to be also examined.

The simplest way to start building application integration is to use *point-to-point* model.

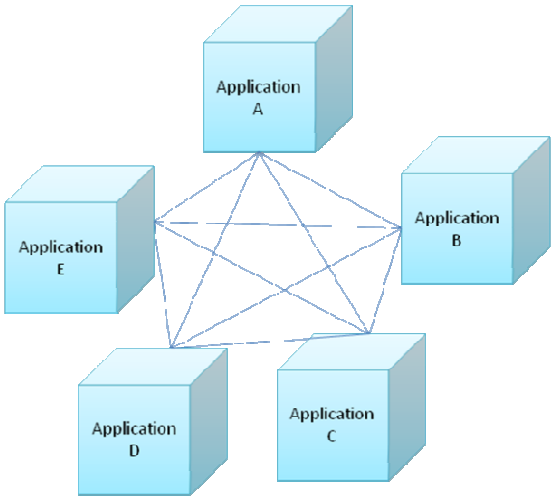

Figure 10. The point-to-point model.

In figure 10, it consists of a decentralized structure in which each application communicates directly with the other applications. However, the limits of point-to-point model become evident soon when integrations are needed to built between more than just few applications. In case an organization is using n number of applications and information needs to be shared between all of them, the total amount of required connections can be calculated as follows: $\frac{n(n-1)}{2}$. It is typical for point-to-point integrations that they are created one by one as business needs arise. This approach causes consistency problems because each solution may have been developed by different person using different technologies. Point-to-point implementations are also difficult to maintain when changes are directed at the applications. All existing integrations have to be checked separately and if they are poorly documented, no one may be able to evaluate what has to be reconfigured. (Tähtinen 2005: 65-66)

The *integration hub* is an alternative middleware model. It provides a centralized structure, in which an integration hub is placed between the applications. Each application communicates with the hub and not directly with the other applications as shown in figure 11. Thus, they need only one interface and connection, the ones that are for the integration hub. (Tieturi Oy 2009: 72-73)

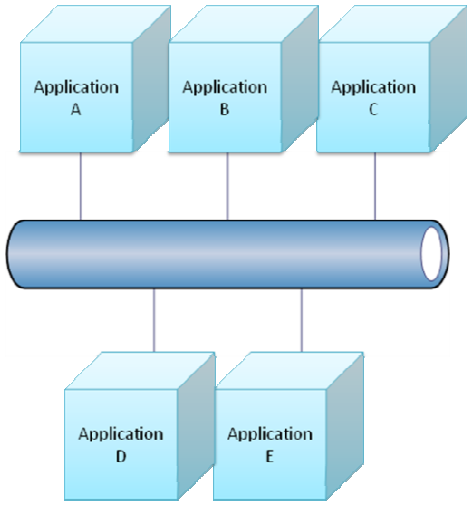

Figure 11. The integration hub model.

Scalability is the main advantage of an integration hub environment. In figure 11, a single connection is needed to the integration hub instead of several others when using point-to-point model. A large-scale organization may have hundreds of applications and it is impossible to create individual interfaces for all of them. It is also much easier to modify or update elements if middleware is based on the centralized model. However, if there are a couple of applications to be integrated and they are relatively simple, it might be too expensive or technically difficult to use this approach. (Microsoft Corporation 2003: 9)

When designing application integration solution, it is important to consider how the applications communicate with each other. There are two possibilities, synchronous and asynchronous communication. Usually, the final solution is a combination of the both methods. (Microsoft Corporation 2003: 28)

Synchronous communication is basically a communication where one application converses with another. It is an interface between two applications where an invocation results in a response once the requested processing is completed. (Kanis 2003: 4) Synchronous communication best suits in such situations where the application must wait for a reply before it continues to processing. (Microsoft Corporation 2003: 28)

Asynchronous communication simply means that one application sends a message to another. (Kanis 2003: 4) It usually guarantees better performance than synchronous communication as applications are not waiting for a response all the time. Also, the connection is not continuously maintained which does not overload the network. Asynchronous communication is mostly used when the application can continue processing after it sends a message to the target application. (Microsoft Corporation 2003: 28)

3.2.2 Physical Models

The physical middleware model depicts both the actual method of information movement and the technology employed. There are several messaging models under the umbrella of physical models. These messaging models are covered in this subsection.

In *connection-oriented communication*, two parties connect and exchange messages. The parties do not disconnect before the exchange is fully completed. This model usually utilizes the synchronous process but it can also be done using the asynchronous one. (Linthicum 2003: 120) *Connectionless communication* means that the source application just passes messages to the target application. It is possible to send messages to both directions but it is not guaranteed that the messages are delivered. The target application responds only if it is required by the source application. (Linthicum 2003: 120) *Direct communication* refers to a model where middleware accepts the message from the source application and passes it directly to the target application. Synchronous processing is commonly used in the direct model. (Linthicum 2003: 121)

Queued communication means that the source application sends the message to a queue and the target application reads the message from the queue as illustrated in figure 12.

Figure 12. Queued communication.

A queue manager is typically required to place a message in a queue. It is possible for the target application to retrieve the message from the queue whenever it is ready. If the target application is required to verify the message or data content, it sends verification back to the source application through the very same queuing mechanism. Compared to the direct communication model, the queued communication has an advantage of enabling the target application to be inactive while the source program sends the message. Also when using a queue, the applications can proceed with processing while they do not have to wait for attention of each other. (Linthicum 2003: 121)

In figure 13, *publishing and subscribing* means that the source application, a publisher sends out the message to the middleware layer without addressing the target application.

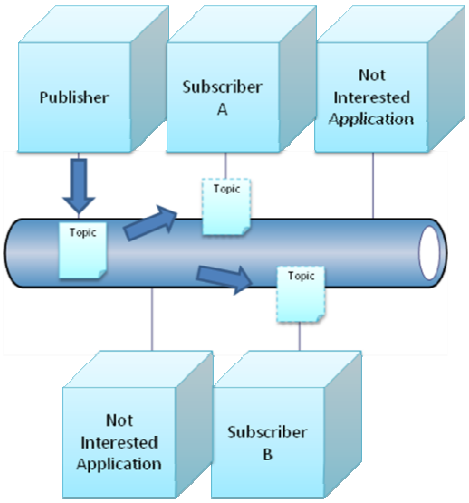

Figure 13. The model of publishing and subscribing.

The publisher does not even have to know anything about the receiver. Instead, a topic name of the message is provided by the publisher. Potential target applications, subscribers register with the middleware and announce what topics they are interested in. (Microsoft Corporation 2003: 111) After the

message has been sent, the middleware redistributes the information to any interested subscribers on the basis of the topic. As a result, the subscribers receive only the desired information. (Linthicum 2000: 137-138) The procedure is clarified in figure 13. Publishing and subscribing can be used in situations where a reply is not necessary and the target application is determined by the content of the request. (Ruh, Maginnis, Brown 2000: 47) Another advantage is that applications can be added or removed at any time because the publisher does not need to know who is listening. (Microsoft Corporation 2003: 112)

Requesting and replying means that the source application sends a request to the target application and waits until the reply is received. The source application does not do any processing while waiting for the reply. However, it is possible to set a timeout parameter that defines a certain amount of time in which the request is resent. Using requesting and replying requires the two applications to understand each other. That is why common process semantics and data format have to be agreed beforehand. (Microsoft Corporation 2003: 110) Requesting and replying type of approach is typically used when the reply is expected to contain information that is necessary for the source application to continue processing. A problem occurs when the target application is unattainable and the source application is not able to finish its task. (Ruh, Maginnis, Brown 2000: 43)

Firing and forgetting allows the source application just to send a message and not to worry if anyone receives it or not. It can be used to broadcast messages to a large number of target applications without checking the content of the message or waiting for a reply. This type of approach suits if the message is wished to attain many target applications but it does not matter if someone misses it. (Linthicum 2000: 139)

3.3 Types of Middleware

As mentioned before, middleware is software which enables applications with different communication protocols and message formats to communicate with each other. Nowadays, there are various middleware solutions

available for organizations to choose from. The solutions are all based on different approaches and this section concentrates in examining what are the characteristics of those approaches. At the moment, five basic middleware types are recognized: Remote Procedure Calls, Message-Oriented Middleware, distributed objects, Database-Oriented Middleware, and transactional middleware (Pinus 2004: 1-5)

Each of the above-listed types of middleware has been developed to solve a problem of sharing information between applications that do not understand each other. The logical and physical middleware models are closely tied to the middleware selection. (Ruh, Maginnis, Brown 2000: 53) For example, some middleware types support either synchronous or asynchronous communication more naturally than others. Different types of middleware are discussed in the next subsections.

3.3.1 Remote Procedure Calls

Remote Procedure Calls (RPCs) represent the oldest middleware type as they were introduced in the 1970s. RPCs are perhaps the easiest middleware type to understand and implement too. RPCs invoke a function within one application, pass the shared data to another application, and invoke the function that tells the server application how to process the data. In figure 13, a result is returned to the client application on the basis of the processing.

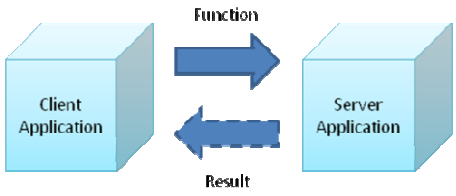

Figure 14. The Principle of Remote Procedure Calls.

For the client application end-user, the procedure is hidden and it seems that the function is executed locally. (Linthicum 2003: 125) RPCs are a good example of synchronous communication. While the RPC is carried out, the current program needs to be stopped until the result is received. If consecutive RPCs are sent to several applications, it ties different systems into a knot.

Doing certain things in a particular order can make it difficult to change applications without affecting the other ones. (Hohpe and Woolf 2003: 51-52)

The advantages of RPCs are simplicity and relatively easy configuration. Still, they are weighed against the disadvantages. Most RPC solutions are not performing well and their functioning requires way too much processing power. Furthermore, many exchanges must be done back and forth across a network to carry out a request. Despite its weaknesses, the RPC technology is still used in many organizations even though a modern EAI architecture cannot be developed on its basis. (Ruh, Maginnis, Brown 2000: 53-54)

3.3.2 Message-Oriented Middleware

The weaknesses of RPCs were brought up in the preceding part. Those weaknesses resulted in the creation of *Message-Oriented Middleware* (MOM). In figure 15, traditional MOM includes basically queuing software that uses messages to move information from one application to another.

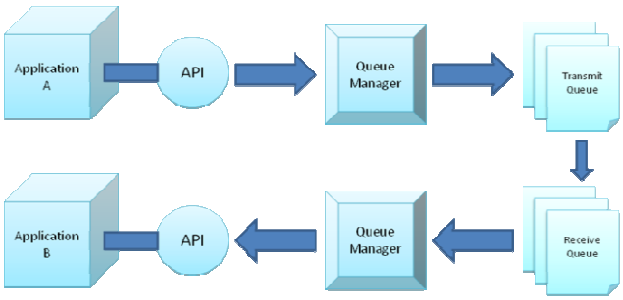

Figure 15. Message-Oriented Middleware.

As the communication is based on messages, direct coupling with the middleware and the application is not needed. Decoupling allows the application to function more independently than with RPCs. (Linthicum 2000: 123-124) Point-to-point is another existing MOM model but message queuing is the primary focus in this paper being far more popular and useful.

The asynchronous model of MOM allows the application to continue processing after sending a message to the middleware layer. The message is sent to a queue manager which takes care of delivering the message to its correct destination. Returning messages are handled when the application has free

time to process them. The asynchronous model makes MOM a better choice than RPCs especially when available network and processing resources are limited. MOM is also able to ensure message delivery from one application to another by message persistence. It guarantees the messages to stay in a queue until the target application is reachable. (Linthicum 2000: 124)

MOM is quite easy to understand as the principle is relatively simple. Messages are just byte-sized units that are easy to manage. They consist of two parts; a schema that defines the structure of the message and data which forms the actual content of the message. (Linthicum 2000: 124-125) MOM also provides the ability to create, manipulate, store, and communicate messages without applications even having to know about it. For example, middleware layer can transform a message from one data type to another in order to make the receiving application to be able to handle it. (Ruh, Maginnis, Brown 2000: 55)

3.3.3 Distributed Objects

Distributed object technology is similar to RPCs but it is based on object-oriented model. *Distributed objects* enable creating object-oriented interfaces to new and existing applications that are accessible from any other application. Interfaces are developed for applications that make software look like objects. (Ruh, Maginnis, Brown 2000: 55-56) In addition, they provide a standard mechanism to access the shared objects as seen in figure 16.

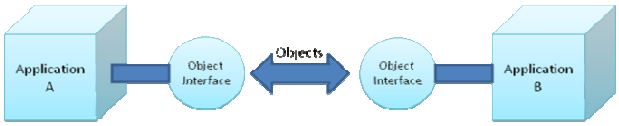

Figure 16. Distributed object technology.

Distributed objects make it possible to create both applications that share common methods and composite applications that support method-oriented application integration. Thus, using distributed object technology may lead to sharing the whole common business logic. (Linthicum 2003: 161)

By applying distributed object technology an application makes an invocation on any object without being aware of its location. Software components can thereby be moved, replaced, or replicated without affecting any other components. Distributed objects are generally considered as synchronous technology. Yet, it has also been extended to cover asynchronous communication. (Ruh, Maginnis, Brown 2000: 56)

It is a major task to change several applications to start using distributed object technology and expose their methods for access by other applications. Distributed objects do not fit for most application integration problem domains as it is quite complicated method and requires many changes in enterprise applications. Sometimes however, distributed objects are the correct solution. The biggest advantage is that they adhere well to many application development and interoperability standards. Distributed object technology is also continuing to mature and new features are introduced addressing its former shortcomings. As always, the most important thing is to calculate the benefits that are expected to gain using certain technology and to compare it to the required resources. (Linthicum 2003: 161-163)

3.3.4 Database-Oriented Middleware

Database access is crucially important part of application integration especially in the case of data-oriented application integration. There are lots of simple solutions available to retrieve information from, or place it into a database. However, *Database-Oriented Middleware* (DOM) has become more complicated recently. It focuses on the exchange of queries, management of results, connectivity to databases, pooling of connections, and other data management related tasks. (Ruh, Maginnis, Brown 2000: 25) DOM has developed into a layer for placing data, a virtual database. It is possible to view data using any model regardless of how the data is stored or what platform the database exists upon. Such layer also enables accessing to any number of databases. (Linthicum 2003: 169-170)

The before-mentioned DOM functionalities are typically achieved through a single common interface such as Open Database Connectivity (ODBC) or Java Database Interface Connectivity (JDBC). Using those technologies, one

can map any difference in the source and target databases to a common model. As a result, integrating the databases is much easier. Both ODBC and JDBC are categorized as Call Level Interfaces (CLIs) that provide a single interface to a number of databases as shown in figure 17.

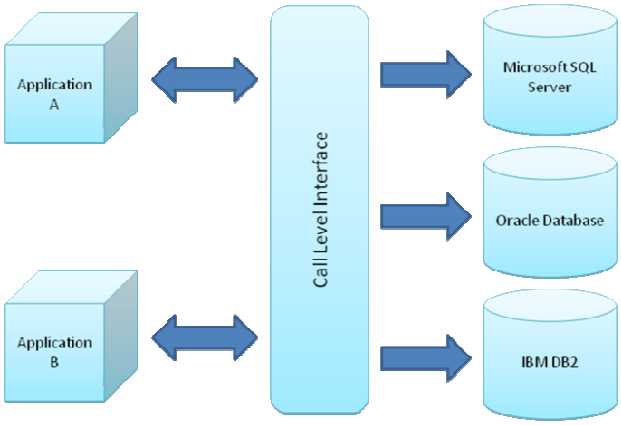

Figure 17. The Model of Call Level Interface.

CLIs translate common interface calls into as many database dialects as necessary. Their job is also to translate the responses into a format that particular application understands. (Linthicum 2003: 173)

The focus of EAI extends beyond data access capabilities. That is why DOM is not usually appropriate as the core of integration architecture. Instead, it may be a useful adjunct to other middleware solutions. (Ruh, Maginnis, Brown 2000: 55) Many application integration products already contain the required DOM middleware to access commonly used database types. The main problem of DOM is that once links to databases have been created, major renovations are needed to change databases. Still, it is relatively easy method and provides can act as a starting point for organizations' integration learning curve. (Linthicum 2003: 169-176)

3.3.5 Transactional Middleware

A transaction is a single logical unit of work that is composed of subunits. All subunits must be completed successfully in order for the transaction to be successful. In information technology, the basic idea of transaction is the notion of two or more processes which all must be successfully completed. When updates occur in applications or databases for example, the updates are

treated as a single indivisible operation. The individual updates may occur at different times based on the structure of the systems but all updates must be completed before the transaction is determined to be completed. (Ruh, Maginnis, Brown 2000: 107-109)

Transactional middleware is based on a centralized server capable of processing information from many different resources, such as databases and applications as illustrated in figure 18.

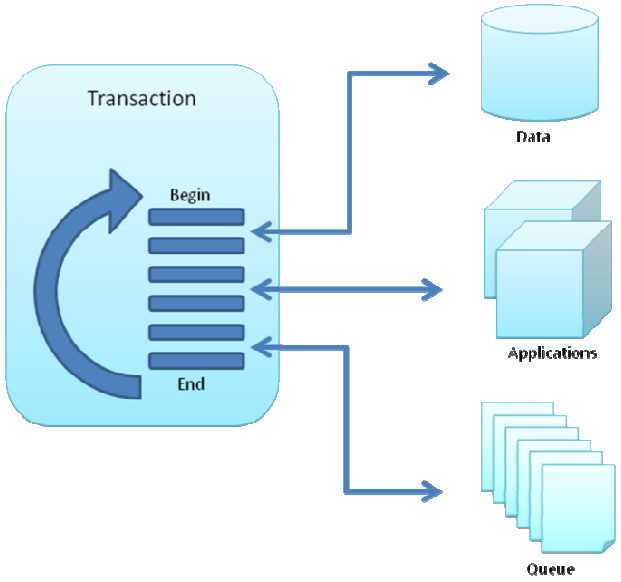

Figure 18. Transactional middleware.

Transactional middleware ensures information delivery from one application to another and supports a distributed architecture. The main benefits of transactional middleware are scalability, fault tolerance, and centralized application processing. On the other hand, the cost of implementing such integration solution is relatively high and Applications must be configured to take the most of it. Despite its disadvantages, transactional middleware fits the best for certain types of integration problem domains. Traditional MOM may be a better option for simple information sharing between applications. However, when there is a need to work at the application service level, transactional middleware could be the correct choice. (Linthicum 2003: 138-139)

Transactional middleware is usually understood to be consisted of two main categories: Transaction Processing Monitors (TPMs) and application servers. TPMs represent traditional technology while application servers approach the issue from a slightly different point of view. TPMs are a type of middleware that preserves the integrity of transaction supporting features such as roll-back, failover, auto restart, error logging, and replication. They allow transactions to be formed by the sender and then ensure that it gets to the right place, at the right time, and completed in the right order. (Ruh, Maginnis, Brown 2000: 56-57) Application servers not only provide a location for application logic and interface processing but they also coordinate many resource connections. Application servers take many existing enterprise applications and expose them through a single user interface. (Linthicum 2003: 144-145)

The previous two sections consist of the theoretical part of this study. In section 2, BI is examined to see why MDM and DW are needed in a global organization. Understanding the purpose of MDM is particularly important for this study as it affects the choices that are made regarding integration architecture. Section 3 introduces technologies and patterns that are available to complete the integration architecture of an organization. All the mentioned approaches are not necessarily applicable in the case of master data application integration but it

Now that it is clear why integrations are needed and what tools and approaches can be used in integration solutions, it is time to take consider how they are being utilized in Konecranes. The next two sections form the empirical part of this study. Section 4 represents the platform architecture of Konecranes and reveals how it has been built up. Section 5 introduces the integration roadmap and it is the most important output of this study. It gives detailed instructions on what has to be made in order to enable integration between GCM and an enterprise application.

4 THE PLATFORM ARCHITECTURE OF KONECRANES

This section is dedicated to representing the platform architecture of Konecranes. The architecture is needed to get various workings and processes together under a clear aggregate. This is also where integrations become beneficial. The platform architecture of Konecranes includes BI tools, data warehouses and master data application, for example. These separate systems are not able to communicate without common integration methods. This section introduces how the systems are connected and what roles do they play in the complete business reporting solution.

As told in section 2, there are three types of business data within an enterprise. To be able to create extensive reports and analyses, the data must be governed in a systematic way. All different data types are managed in their own dedicated systems to ensure the governing methods are chosen correctly and particularly for certain data type. BI tools build up reports and analyses based on that data. If some data is invalid, decision-makers cannot trust the BI information and the whole BI structure is useless. Therefore, it is vital for an organization to take care of the data quality. However, none of the data types are useful as an island of information. The data needs to be bundled up together, that is, to integrate the systems that contain data. That is what commonly-recognized platform architecture is needed for.

The platform architecture of Konecranes can be put to a layered model as illustrated in figure 19. The platform architecture consists of five layers:

- Data repository layer
- Application layer
- Data warehouse layer
- Presentation layer
- Integration layer

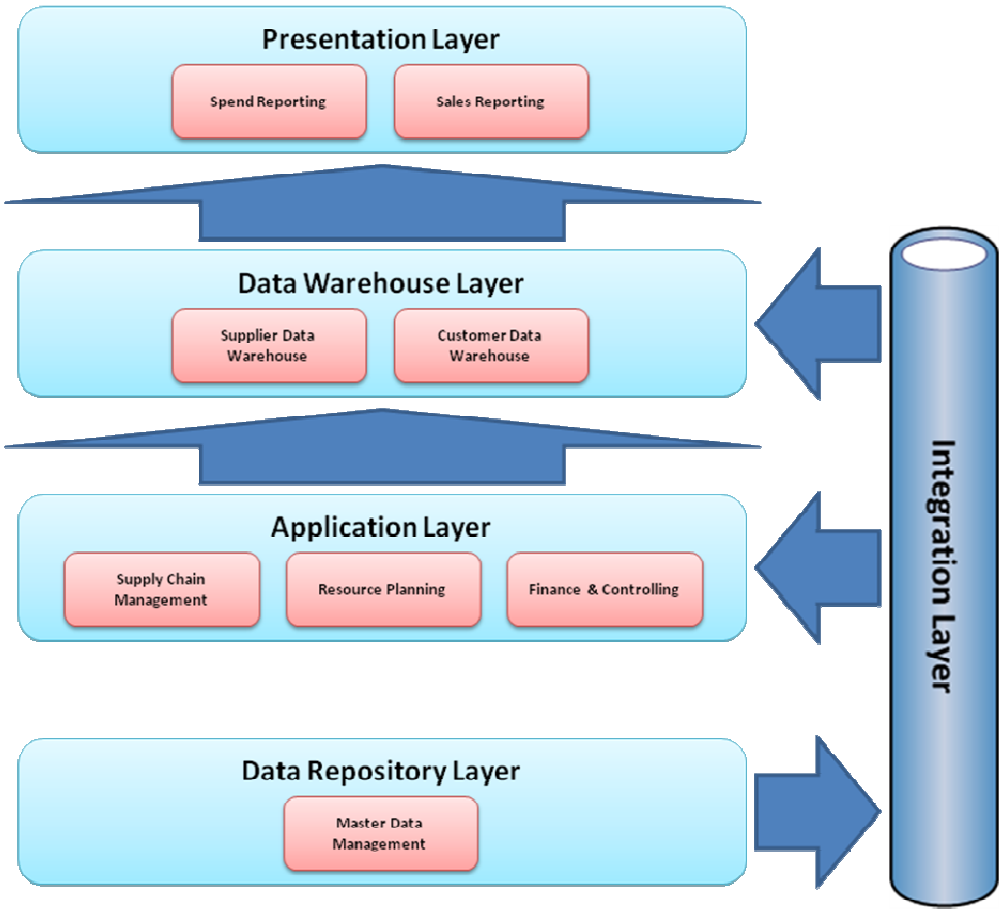

Figure 19. The platform architecture of Konecranes.

The three first-mentioned layers hold the actual business data. Data repository layer is where master data is stored and maintained. Application layer is composed of number of enterprise applications that may have nothing in common. Nonetheless, each of them is used for certain purpose and together they hold all transactional data in the enterprise. Data warehouse layer is the location of Konecranes' analytical data and it is gathered by combining transactional data and master data. Even though all business data lies in those three layers, the platform architecture is not complete without the other two layers. Presentation layer is the one that includes the tools needed for creating reports out of the data warehouse contents. The analytical data itself is worthless without having proper BI tools to prepare it into elucidative form. Integration layer is the glue that ties other layers together. It enables separate layers to share different types of business data with each other. Thus, there is no such BI solution or platform architecture that could function without carefully designed integrations.

This section concentrates on the five above-described layers and their roles in Konecranes platform architecture. As this paper is about the integration of master data application, the main focus is on the integration layer and the data repository layer. Piecing together the big picture requires understanding the other layers and that is why they are also discussed.

4.1 Data Repository Layer

Data repository layer forms the stone base of the Konecranes platform architecture. It is dedicated to storing and maintaining master data within the organization. At the moment, only customer and supplier data are considered as master data and, for example, product data has been left outside the scope. The purpose of master data is to have a single version of the truth for each and every one of the customer and supplier records. In other words, there should be only one global master data record for each customer and supplier entity. Enterprise applications can all have their own record for the very same entity but master data holds the best knowledge of what is the correct address of some customer for example. The master record includes cross-references to those enterprise applications that store the same entity. So the master data application can distribute its data to the interested applications.

GCM is the global MDM solution of Konecranes. It contains basic information regarding Konecranes' customers and suppliers. Trading partner is the common term for company entities in general. The most meaningful information on trading partners is stored in three main tables in GCM database as shown in figure 20: TRADINGPARTNR_TP, TRADPAADR_TD, and TRADPARREL_TL.

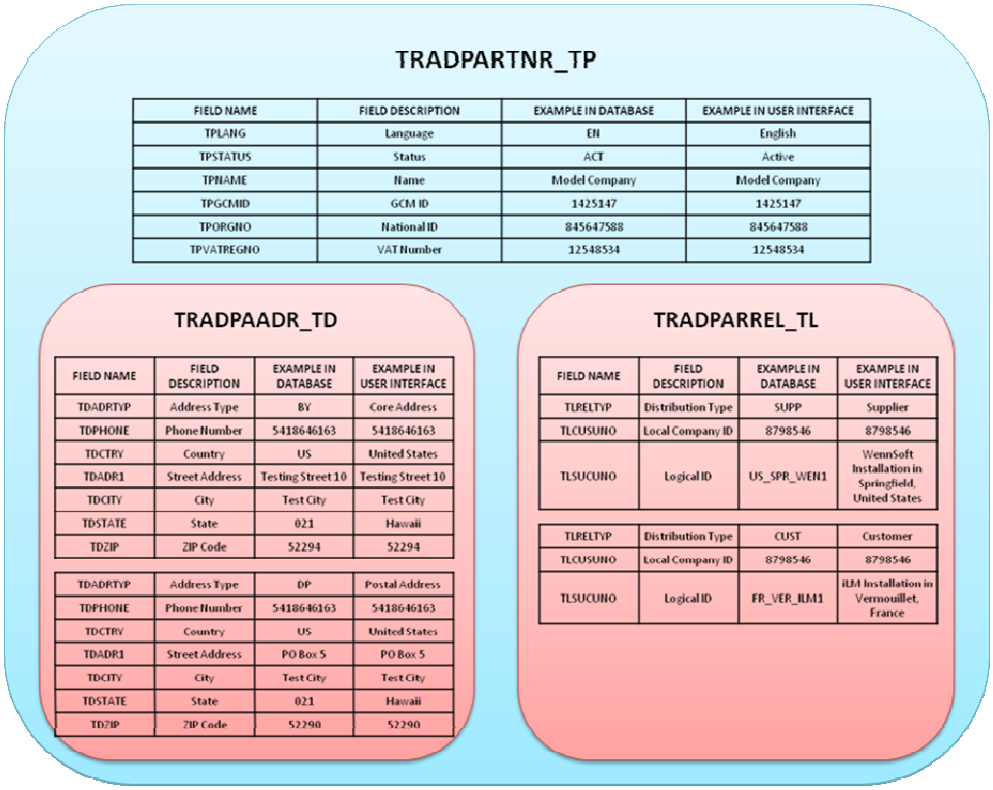

Figure 20. An example of trading partner record in GCM.

The TP table includes very basic information on trading partners. This information is used for separating entities out of each other. The most important fields of the TP table are name, language, status, GCM ID, national ID, and VAT number. The TD table is reserved only for contact information. There are two types of addresses in GCM: core address and postal address. Contact information is regarded as part of master data as it is not transactional and can be expected to stay constant for relatively long period of time. The main fields of the TD table are address type, street address, ZIP code, city, state, country, and phone number. The TL table is needed for managing cross-references to enterprise applications. That information reveals the relationships between the master record and the records in tens of enterprise applications. The fields of the TL table include distribution type, local company ID, and logical ID. The three tables are connected with common trading partner ID.

When building integrations to GCM application, XML messages provide the required interface for communication. There is a standard XML message

format in use to force every enterprise application integration solution to follow the same pattern in GCM's end. The previous section introduced the data structure of a trading partner record. The same example is used also in this part to see how it is transformed into the form of an XML message. The example message can be found in appendix A.

GCM is integrated with the application layer and the data warehouse layer using the integration layer to transfer and transform data from one layer to another. The integration technologies by which this operation is performed are discussed in section 4.5. Application and data warehouse layers need master data to have the best possible information regarding the customers and suppliers of Konecranes.

In practice, GCM integration is implemented so that every time when an end-user creates a new record or updates an existing one in GCM user interface, an XML file is sent to integration layer. The message includes everything that is needed to distribute information to correct destinations. As seen in appendix A, the message contains all enterprise applications that share the same trading partner entity. The master data record can be recognized by the enterprise application by local company ID that is also a part of the XML message. Information in the message is examined by integration layer tools that forward the trading partner information to all related enterprise applications. Data is only distributed from GCM to other layers. GCM does not receive any data regarding trading partners as the master records are solely maintained in the application itself.

4.2 Application Layer

All important enterprise applications have been brought to GCM by creating a new logical ID for them. It requires a significant amount of manual work for enterprise application administrators to maintain customer and supplier information in both their own application and in GCM. As long as integrations are not there to automate the process, the end-users really have to create a record for the same entity twice. Integrations reduce the amount of extra work and increase the quality of data by eliminating human errors.

Konecranes application portfolio consists of tens of heterogeneous systems that are running on multiple servers around the world. Each business unit has its own applications for different purposes because there has not been corporate-wide policy on tool selection. Lately, lots of effort has been made to reduce the number of applications across the organization. Still, the variety remains considerably high as illustrated in figure 21.

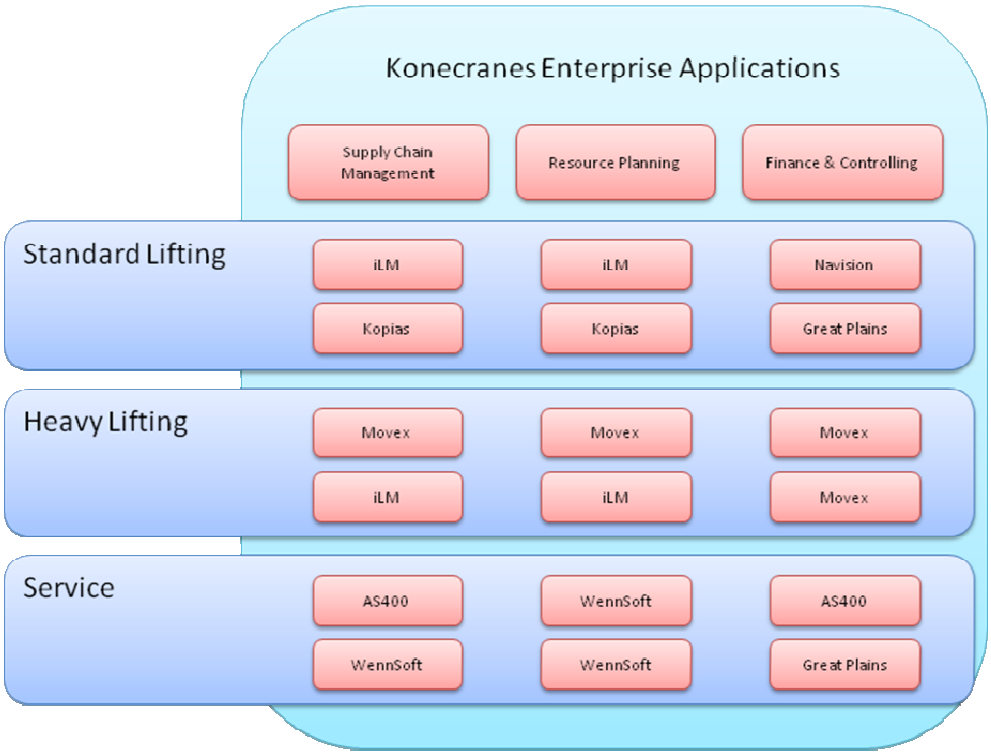

Figure 21. A sample of Konecranes application portfolio.

The figure only shows the main applications of each business unit but there are also minor applications in addition to this. Usually there are several installations of each application in use. For example, WennSoft and iLM have more than ten installations in different countries and plants globally. Even though the basic technology is the same everywhere, the installations are sometimes slightly modified and it may cause problems when considering integrations. The application layer of Konecranes platform architecture consists of all these enterprise applications.

The application layer of Konecranes platform architecture is integrated by many different ways. As described earlier, all enterprise applications need to

be integrated with data repository layer. This paper concentrates particularly on this matter. Konecranes Heavy Lifting application Movex has already been integrated with GCM as the pilot solution. Other integrations solutions have also been discussed and the main purpose of this paper is to support the actual integration projects of those applications.

Application layer is also integrated with data warehouse layer. That integration is not related to integration layer but is created directly using ETL process. Transactional data from enterprise applications is transferred to Konecranes data warehouses. There are two data warehouses in use; one for customers and one for suppliers. They will be covered in section 4.3 in more detail. Also the integration between the application layer and the data warehouse layer is taken a closer look at in the same section.

4.3 Data Warehouse Layer

The application layer of Konecranes platform architecture is formed by a number of heterogeneous applications. Each of them holds significant amount of data regarding purchase and sales transactions for example. It would be possible to create reports based on their data system by system. The problem is that they do not share the same data model and the technology is different in each application and their separate installations. Thus, it would be nearly impossible to combine all reports into one corporate-wide report that covers business transactions all over the fragmented application portfolio. Unconnected reporting would also require too much resources from BI tools as it is very consuming to use so many data sources.

The solution for creating solid reports out of business transactions of the entire corporation is to use a common data warehouse that gathers transactional data from enterprise applications. The data warehouse has a commonly-agreed data structure which makes it a lot easier to govern the reporting process as a whole. The data warehouse also receives master data from GCM to get a consistent view on customers and suppliers as business objects. Master data does not only increase the reliability and quality of reporting but helps with concluding the hierarchies and legal structures of Konecranes trading

partners. All in all, enterprise data warehouse is a necessity of consistent reporting and it maintains business critical data on which decision-makers can rely on.

As illustrated in figure 19, Konecranes data warehouse layer consists of two data warehouses. One is dedicated to supplier data and the other one to customer data. The following technologies are used in the data warehouse solution of Konecranes: the IBM InfoSphere DataStage supplier data warehouse and the Domino customer data warehouse

DataStage gathers data about the suppliers that Konecranes purchases material from. The most important information is the spend volume to see how the total spend amount is distributed. *Domino* performs the same duties on the customer side. It holds sales transactions of the Konecranes and it is possible to see who the most important customers are and what the amount of total sales is.

Data warehouse layer is integrated directly with application layer. DataStage and Domino are using different principles when gathering data from their sources. DataStage follows the ETL process that was introduced in section 2.3. First, data is extracted from each enterprise application. Next, it is transformed into such form that is fulfils the requirements of the DataStage data structure. Finally, the data is loaded into DataStage where the maintenance is done. Domino, however, is not operating according to ETL process. The data is imported into Domino manually via user interface.

Integration takes place between the data warehouse layer and the data repository layer. Both supplier and customer data warehouses take advantage of master data that is kept up in GCM. Again, DataStage and Domino do not share the same integration procedure. Even though the basic approach is information-oriented for both integration solutions, the exact method of implementation differs. DataStage makes use of data replication that was discussed in section 3.1.1. It has direct access to GCM database. On the other hand, Domino uses messaging as the integration method. An XML message is sent to Domino every time when customer record is created or modified in GCM.

4.4 Presentation Layer

The purpose of the presentation layer of Konecranes platform architecture is to provide decision-makers with comprehensive information on business operations. There is so much raw data available that it must be refined into more easily understandable format. The data is hidden in the background to keep its extensiveness but the manner of representation is simplified to clear up the high-level trends. Business management does not necessarily want to know all little details behind the figures but it is also possible to dig deeper into grass roots.

There is one common BI solution in Konecranes for creating spend and sales reports. The product is IBM Cognos Business Intelligence that enables creating versatile reports and analyses based on the data in supplier and customer data warehouses. It helps comprehending the big picture of the business trends and finding bottlenecks within business processes. Common BI tool also lightens the load of enterprise applications as the reporting does not have to be built up one by one. Figure 22 shows few samples of what kind of content can be created using Cognos.

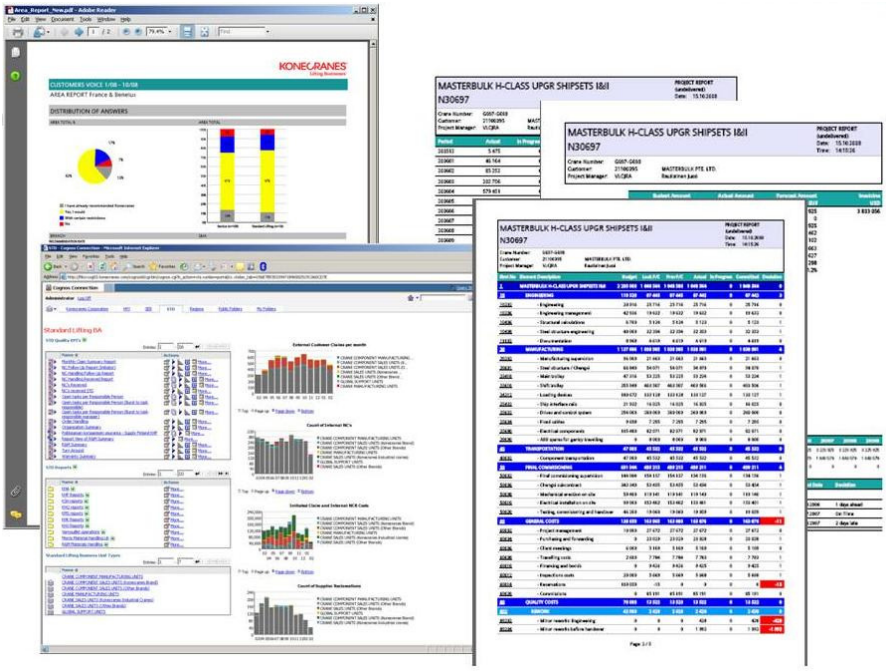

Figure 22. Cognos reporting samples.

The presentation layer is not directly in touch with the integration layer as it is not supposed to store business data itself. BI tools just utilize existing data in data warehouses to run predefined reports. The presentation layer is only integrated with data warehouse layer where the data has already been structured in the way that is supports reporting needs. Reports are created by making queries directly to the data warehouses and picking up the desired information. There are ready-made reports for most common purposes and they can be examined by those who are eligible to access that particular information.

4.5 Integration Layer

Konecranes application portfolio consists of tens of enterprise application installations that differ notably from each other. It would not be efficient to build integrations one by one between all applications. The purpose of the integration layer is to enable integration of data across different systems and applications in a standardized and managed manner. The integration is realized through several functional concepts such as mediation, routing, transformation, and queuing of messages between those systems. At the same time, the integration layer enables standardization and manageability by introducing standard patterns and centralized focus point for integration which is monitored and operated according to well-defined processes.

Figure 23 shows the two main components which Konecranes integration layer is based on. These components are WebSphere Enterprise Service Bus and WebSphere Message Broker as figure 23 illustrates.

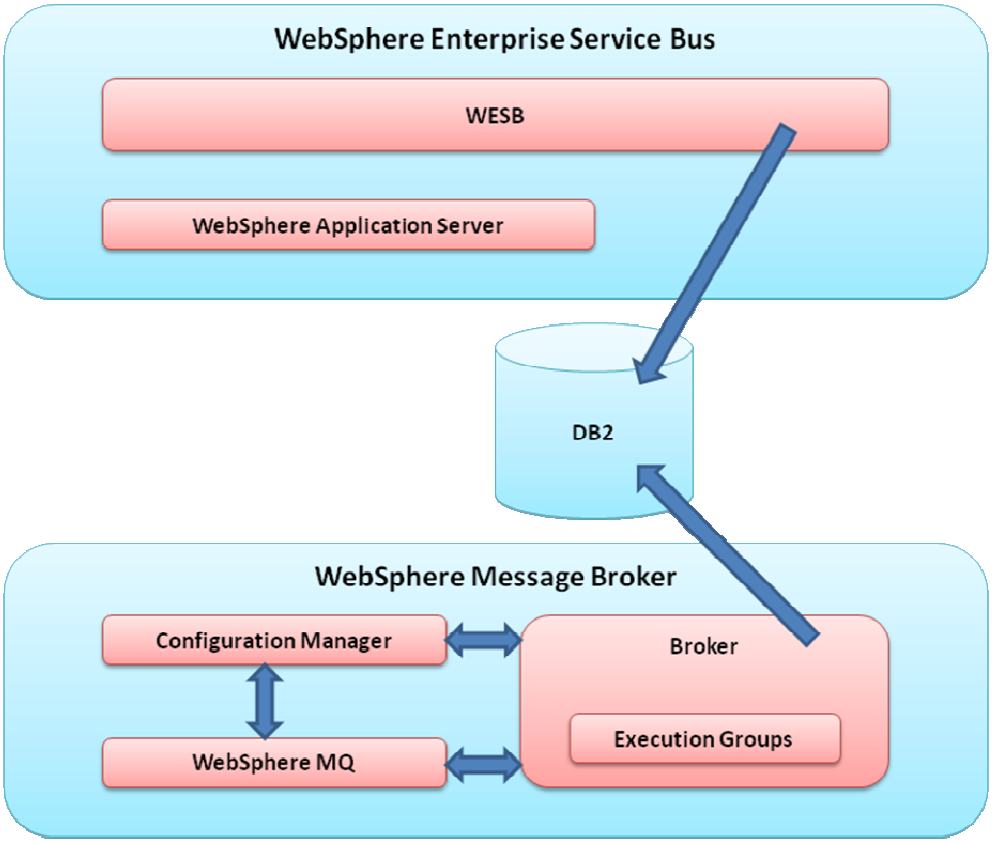

Figure 23. Konecranes integration layer high-level overview.

WebSphere Enterprise Service Bus (WESB) is designed to provide a middleware for IT environments that are built on open standards and service-orientation. It acts as a runtime environment that enables loose coupling of service requestors and service providers. Using mediation flows, WESB supports protocol transformations, message transformations, and dynamic routing decisions. It runs on WebSphere Application Server which also leans on open standards. WESB is authored using WebSphere Integration Developer which makes it possible to use uniform invocation and data representation programming models and monitoring capabilities.

The main focus of this paper is on the other integration layer component, *WebSphere Message Broker* (WMB). All GCM related integrations are to be built on WMB due to its flexible messaging services. It enhances the flow of messages without the need to change either the Applications generating messages or the applications consuming them. In practice, WMB is a set of application processes that host and run message flows. Those flows consist of a

graph of nodes that represent the processing needed for integrating applications. In addition to flows, WMB also hosts message sets including message models for predefined message formats. The basic idea of WMB is illustrated in figure 24.

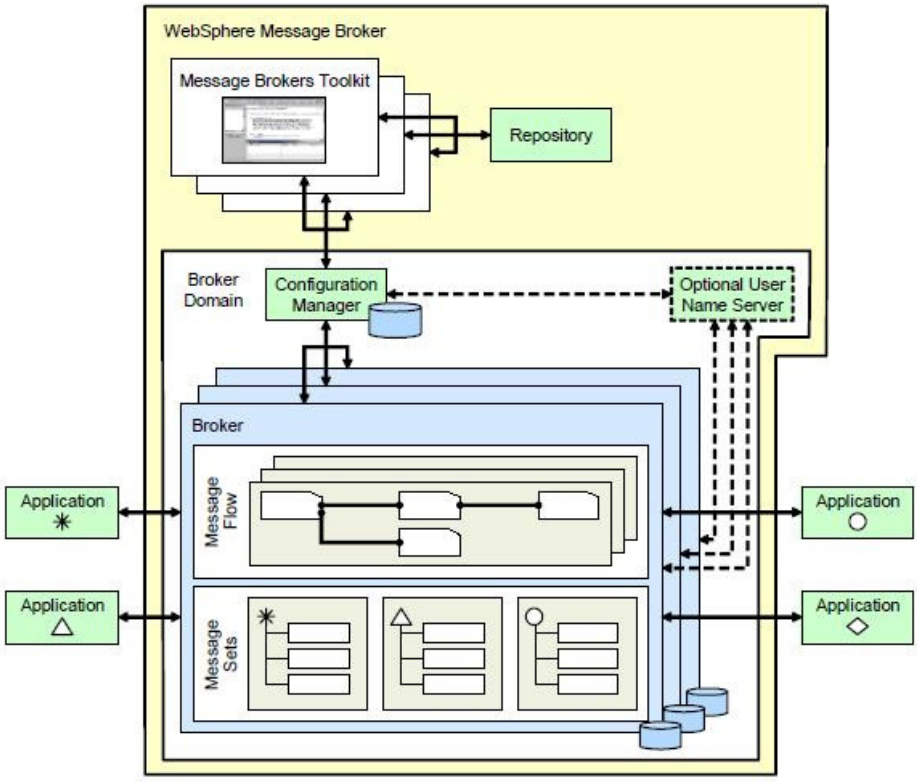

Figure 24. WebSphere Message Broker overview. (Davies et al. 2007: 47)

When a message from an enterprise application arrives at WMB, it processes the message before passing it on to one or more other enterprise applications. WMB routes, transforms, and manipulates messages according to the logic that is defined in message flows. WebSphere MQ is used as the transport mechanism to communicate with the configuration manager, from which it receives configuration information. As there can be several brokers within WMB, the configuration manager is also needed to communicate with any other associated brokers. Execution groups enable message flows within a broker to be grouped together. Message flows are deployed to a specific execution group. WMB is configured using WebSphere Message Broker Toolkit. The toolkit uses the configuration manager as the interface to access the broker.

The role of the integration layer has already been touched in previous sections. It acts as glue to combine the other layers of Konecranes platform architecture. In this context, the most important task of the integration layer is to receive XML messages from GCM and forward them to enterprise applications and data warehouses. The GCM data structure is discussed in section 4.1.1. The structure of an XML message which is sent from GCM is available in appendix A. All distributions to enterprise applications are defined inside the element TRADPARREL_TL. Based on that information, WMB is able to conclude where that particular message needs to be forwarded to. For example, code FI_HVK_DOM1 stands for Domino customer data warehouse and FI_HVK_MVX1 for Heavy Lifting enterprise resource planning software called Movex. WMB is configured to be familiar with all codes and it knows where to route the incoming messages.

Besides routing the messages to correct addresses, WMB can also be used to transform them. If some enterprise application needs to receive the message in another format than XML, WMB can be configured to modify the file type according to the needs of the receiver. It is also possible to restructure the contents of the message too if receiving application needs some data transformations. All this can be added to be done inside the broker message flows. The idea is to hide these operations from the related applications because all message routings and transformations are done in the single place.

5 ROADMAP FOR MASTER DATA APPLICATION INTEGRATION

GCM integration project begins when there is need to integrate GCM with an internal enterprise application. Until now, there have not been common guidelines to follow when starting to build a new integration solution. The purpose of this section is to introduce a universal roadmap that can be utilized in the case of all GCM related integration projects. Even though the roadmap needs to be universally applicable, it should also go into details as much as possible. In figure 25, there are certain recurring steps that characterize this type of integration projects. As told in the introduction, representing those steps in a logical order is the main objective of this paper. GCM integration can be divided into eight steps:

- Organization
- Documentation
- Data cleansing
- Design
- Implementation
- Testing
- End-user training
- Launch

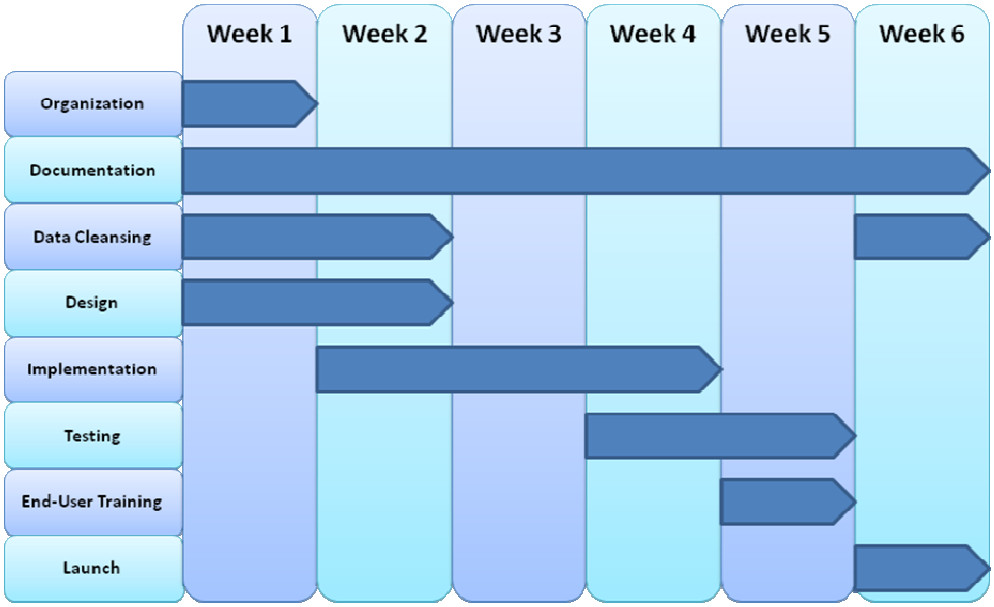

Figure 25. GCM integration project template.

Organization step describes what roles need to be determined in order to ensure all aspects of an integration project are taken care of specifically assigned persons.

Documentation step cannot be underestimated when considering the maintainability and reparability of an integration solution. Each solution is implemented in a slightly different way and they cannot be modified unless there are proper documents available.

Data cleansing step describes the importance of data quality in a master data application. As data accuracy is the most important value of such application, GCM's data needs to match the data of an enterprise application before integration solution can be launched. It requires data cleansing as there are always inconsistencies between the data of two separate systems.

Design step asks all questions that need to be answered to before actually implementing the integration. What data is needed in the enterprise application and in what form? How is the data transferred from GCM to the enterprise application? Are some data transformations needed?

Implementation step explains in practice what must be done to enable integration successfully between GCM and the enterprise application. What are the concrete actions behind an integration solution and who does them?

Testing step states the importance of testing in an IT project. The integration solution is first implemented in development environment and it requires extensive testing to ensure everything is working as expected. After the solution is found out to be flawless in that environment it is also tested in test and production environments.

End-user training step is an utterly important part of a carefully-planned integration project. The quality of master data ultimately depends on the know-how of end-users. They must be trained to feed the information into GCM correctly and in systematic way. If the integration implementation changes the working methods, those changes have to be pointed out as clearly as possible.

Launch step takes place when all previous steps have undoubtedly been successfully finalized. The integration is launched in production environment and end-users are informed about the new solution. This is the easiest step of all if the previous steps have been conducted in a thorough manner.

5.1 Organization

When starting a new integration project, the first step is to arrange a project start-up meeting and organize an integration project team. Each team member is assigned responsibilities in a particular area. In GCM related integration project, the following roles need to be defined:

- Project manager
- Integration architect
- Integration developer
- Application owner
- Application developer

The first three roles are stacked with people from the integration services team of Konecranes Group IT. The latter two are from the IT department of certain Konecranes business unit.

The main task of the project manager is to supervise the overall progress of the integration project. He/she does not participate in the concrete creation of the integration solution but is in charge of integration development. The project manager understands the benefits that are expected to be gained with new integration solutions and provides required resources for the project. He/she also communicates the requirements and benefits to the business people in the company.

Integration architect is the one who designs the integration solution in cooperation with the enterprise application owner. It is important for him/her to be aware of different integration patterns and technological options that can be chosen for various integration needs. Integration architect knows the characteristics of GCM and how enterprise applications should be connected to the corporate integration architecture. In addition to being responsible of integration design, the architect somehow participates in all eight project steps. He/she makes sure the documentation is properly written, organizes data cleansing together with the enterprise application owner and GCM key user, and communicates the integration design to the integration developer. The integration architect also tests the integration solution from the GCM's point of view, arranges GCM end-user training with the GCM key user, and is present when the integration is launched.

Integration developer concentrates on the true implementation of the integration. He/she fully knows the technical details and is familiar with the Konecranes integration layer. The integration developer is in close contact with the enterprise application developer. They are the two people who know the best how to connect two applications together in practice. Of course, the developers participate already in the design step to make sure the designed solution is possible to be implemented in a real life situation. Another important task of the integration developer is to keep documentation up-to-date throughout the project. An integration specification document is written about each solution.

Application owner is usually the key user of an enterprise application. As mentioned, he/she designs the integration solution with the integration architect. The role of the application owner is crucial as he/she defines what the possible technical options are that can be used in the case of this particular enterprise application. Sometimes, changes need to be made to the enterprise application to enable sharing data with GCM. Even a whole new interface may have to be build for this purpose. As the integration architect has many responsibilities on GCM's side, the role of the Application owner is very similar on the side of the enterprise application. He/she validates the documentation and arranges data cleansing, testing, and end-user training. The application owner decides when the integration is ready to be deployed in the production environment and informs the end-users about it. He/she also provides support if the end-users bump into problem situations while using application according to new process.

As told in the previous part, changes may have to be made to the enterprise application to make it fully compatible with GCM. This is when the application developer comes forward. He/she knows everything about the enterprise application and gives instructions to the application owner while the integration design is negotiated. This role can be compared to the role of the integration developer as the job includes filling in the integration specification document and lots of testing to make sure the integration solution works perfectly.

5.2 Documentation

Proper documentation is widely recognized to be one of the most important areas of an IT project. In practice, however, when working according to tight project time scale, documentation is often neglected as too time-consuming or unnecessary. This is not how it should be as lack of documentation makes it impossible to maintain and develop a complicated IT solution. Application integration project is a good example of a project in which documentation is part and parcel of the final project outcome. This is because application integration always concerns at least two distinct systems and very few master all related applications and technologies.

There is a document template that needs to be filled in the case of each GCM integration solution. The template and an example case are available as appendix B. The example represents all technical aspects that should be included in true integration project documentation too. There is very detailed information regarding WMB configurations, routing settings, and the actual implementation of integration. Some parts of the integration specification document go so much into details that they are not even in the scope of this study. However, as integration project documentation requires taking a look at that detailed information, it is worthwhile to add the documentation template itself as an appendix.

Filling in the integration specification template belongs to the integration developer and the application developer. They have enough knowledge to be able to write thorough documentation about complex integration solutions. Of course, integration developer has the biggest responsibility of the documentation because he/she works at WMB on daily basis and knows how to configure it. Even though, he/she cannot take care of the whole documentation phase by him/herself. The application developer is an expert in the questions regarding the enterprise application. The integration architect and the application owner may also be needed for reviewing new versions of the documentation and the project manager is the one who approves the final version.

Documentation step starts in the very beginning of an integration project. Common procedure is that documentation is bunched together in rush after the solution has already been implemented. The correct way is to complete the integration specification document little by little while the project moves on. As documentation should be treated as continuous process, there can be several initial versions of the integration specification document. The final version, however, is not ready before the whole project is completed.

5.3 Data Cleansing

Data is the actual capital of a master data application. Data is also the most important aspect of the Konecranes master data application, GCM. Thus, data

quality is the best way to measure GCM's value for the corporation. If the data is consistent and accurate, decision-makers are able to trust it and draw conclusions on its basis. On the other hand, if GCM contains incomplete or even false information, reports and analyses cannot be fully trusted and the whole application becomes useless.

In an integration project, data cleansing has to be taken care of before deploying the solution into production environment. If cleansing is done improperly, it is difficult to do it afterwards because the applications are already connected and changes affect other parties too. The target is to make the data of GCM to exactly match the data of an enterprise application. There are usually quite a few differences in the data of GCM when compared to the data of the enterprise application. It is because the data has been manually typed into both systems and some errors always occur.

The data cleansing step is divided into two phases. The first two weeks are spent standardizing the existing data and making it as complete as possible. Also the unnecessary and faulty data is cleared at this point. The second phase takes place just before launching the integration solution. It is the last check to assure the validity of the data. The second phase is also needed to correct the flaws that have possibly been entered after the first data cleansing phase.

The data cleansing is started by taking a batch out of the databases of both GCM and the enterprise application. The batch should include all information that exists in both databases even though the field names are probably different. The next step is to match the field names with each other. For example, if GCM's field containing street address is called TDADR1, it should be matched with corresponding field of the enterprise application. When all fields are matched, the data is ready to be sorted. The sorting categorizes records into three classes. There are records that exist in both applications, records that only exist in GCM, and records that only exist in the enterprise application. If a record exists in both applications, there is no need for further actions. The record is identified using so called GCM ID number which is unique for each record. It links the record in GCM with the record in the en-

terprise application and integration enables the changes in GCM to be automatically updated into the enterprise application. If a record only exists in GCM, the distribution to this particular enterprise application has to be deleted. The record is not in the enterprise application and there should not be a distribution either. The third class is formed by those records that only exist in the enterprise application. It may be because of someone has forgotten to create the record in GCM or to create a distribution to the enterprise application. Identifying the problem requires manual work as the GCM database must first be gone through to check if distribution can be added to some existing record. If this is not possible, a whole new record must be created and distribution to the enterprise application created.

Completing the data cleansing may require lots of time and manual work. It is hard to be estimated precisely as it depends on the data quality in GCM and in the enterprise application. End-users' skills and motivation are of great concern to this matter. If they have done a good job and updated information in the both applications conscientiously, the data cleansing step could be carried out in one day. However, enough time has to be reserved to be sure that even the messiest of applications can be cleansed according to schedule. That is why two weeks are booked for data cleansing in the roadmap. The data quality is also rechecked on the last week of the integration project. It is possible to make last minute improvements and to make sure the data is exactly the same between the two applications just before deploying integration solution.

5.4 Design

Successful application integration is all about understanding the requirements of connecting parties. Each integration solution needs to be designed individually as applications often differ from each other in many ways. They have different data structure, interfaces, and operations model for example. Thus, out of the box solution does not fit into varying integration needs. Instead of trying to create a ready-made solution for application integration, one should concentrate on drawing common guidelines to ease the integration design process. This subsection introduces few viewpoints that the integra-

tion architect must consider. To help designing an integration solution, common issues are put together. These are the issues that need to be considered in order to be able to take all related aspects into account.

The first issue is about the number of connecting parties. It must be clarified how many applications are included in the integration solution. This paper concentrates on GCM integrations meaning that there are only two connecting applications by default. GCM is the one that sends messages and some enterprise application receives them. Despite that, the whole truth is not quite that simple. As mentioned, most enterprise applications have many installations across the world. Different business units, factories, and sites may have their own installations of the same system. When integrating those separate installations with GCM, it has to be decided whether it is practical to integrate them one by one or if they can be bundled up somehow. Each installation is always considered as a unique application by GCM but sometimes the integration logic can be shared between the installations of the same enterprise application product. When integrating several installations at once, more attention has to be paid to those little differences they have. It is worth the effort if it supports more simple integration architecture.

An important issue to be solved is whether the messaging is implemented synchronously or asynchronously. It must be decided if GCM needs to wait for a response before continuing processing or not. In this particular integration case, the answer is certainty. The solution is built using transactional processing and if an error occurs, it is rolled back to a previous phase. Another way to secure successful delivery of messages is the usage of message queues. The queue is set to be persistent which means that messages will stay there until they have been read from the queue. All these choices refer to asynchronous messaging. The response time of an integration solution depends on how often messages are read from the queues. In GCM integrations, the message queues are read as soon as a message appears. In practice, there is basically no delay.

The next design issue concerns transformation. It depends on the connecting enterprise application whether the data need to be transformed before it can

be passed to its destination. That is, if the enterprise application does not understand the original message format, it has to be modified in the integration layer. It must also be considered if it is enough just to do simple mapping between the different fields of databases or if some complex logic is required. GCM sends messages out in XML format. Most applications are able to process those messages but some may need the same data in different format. If such situation occurs, WMB is configured to transform the message in another format. Even though the receiver could read XML messages, there is always mapping to be done. For example, the trading partner name element in the XML message is TPNAME. However, the receiver does not know what it means. The enterprise application must be told that this field corresponds to its field named COMPANY_NAME. All elements in the XML message must be gone through and mapped to match the fields of the enterprise application in question.

Correct routing of messages is a crucial part of integration design. The routing decision is done between two options; static and dynamic. The answer depends on whether the messages can always be routed to the same enterprise application or if they need to be routed according to message contents. Again, in the case of GCM integrations, the answer is simple. A trading partner record can have tens of distributions to different enterprise applications and data warehouses. Every time a record is updated, a new message needs to be sent to all of those applications. The XML message includes the distribution information as seen in appendix A. Therefore, WMB reads the message to conclude where the message should be forwarded. Each enterprise application has its own logical ID and WMB sends the message in correct format to all those applications that are mentioned in the original message.

The last issue concerns volumetric and other non-functional requirements. It needs to be defined how many messages need to be transmitted per day, what is the size of a GCM originated XML message, and what is the urgency level of these messages. The number of transmitted messages is hard to estimate as it grows as integrations go along and bring more traffic. When talking about single integration solutions, they should be prepared for handling few thou-

sand messages a day. The size of an XML message varies from a couple of kilobytes to few tens of kilobytes depending on its contents. The size is so small that it will not become a problem with current amounts. As told, GCM originated message gets to enterprise application almost instantly. The end-users can see updates in their local applications right after a GCM record is modified.

5.5 Implementation

The previous section introduced the most important viewpoints on the design of GCM integrations. As well begun is half done, the design phase must be carried out carefully. It eases up the actual implementation phase if all details are well-designed. However well an integration solution is designed, it is impossible to be prepared for unexpected flaws. That is why implementation is started already when the design phase is still ongoing. Thereby those flaws can be discovered in practice and there is enough time to fine-tune the design accordingly.

In typical GCM integration project, there are three parties involved. Data repository layer sends messages that contain master data. Integration layer takes care of routing the messages to correct enterprise applications, transforming the message formats, and modifying the message contents according to receiver's needs. Application layer provides an interface for the integration layer to distribute the messages to. This section examines the tasks of those three parties. What needs to be done in each layer in order to enable master data delivery from GCM to enterprise applications?

There is not much to be done in GCM when new integration solution is implemented. GCM's biggest workload is related to the data cleansing step which is its own, separate step. The actual implementation requires just creating a new logical ID and defining its interface. Let us use take an enterprise application as an example to show how it is defined in GCM. The following configurations should be used: Logical ID = FI_HVK_ENT1, Interface Name = WMQ, and Interface Instance Name = KCI.GCM.FI_HVK_ENT1.IN.

The logical ID consists of three pieces: the country code for the local office, the code for that particular site, and the code for the enterprise Application. In the above example, FI stands for Finland, HVK for Hyvinkää, and ENT1 for an enterprise application. Interface name should always be set to WMQ. GCM integrations have been decided to be done using message queues so it is the same for all. Interface instance naming is also standardized which means that only the logical ID item changes. The interface instance name is equivalent to the queue name in WMB. It ends with IN because everything is thought from WMB's point of view and traffic from GCM is considered to be incoming.

GCM sends messages to WMB using a single message queue. WMQ is already installed on GCM application server so it does not bring more work at this point. Another constant particle is the outgoing XML message. GCM always sends the message in the same format that can be found in appendix A. Both of these aspects justify the use of the integration hub model in GCM integrations. Everything does not have to be recreated when integration need arises. The basic infrastructure already exists.

Usually, the implementation step requires the most effort in the integration layer. WMB is at the centre of events as it reads XML messages from a message queue, modifies their contents and transforms them into another file type, and distributes them to the correct enterprise applications. On the other hand, a basic, standardized integration solution is not that complicated if the enterprise application is able to understand the XML message in the raw.

GCM sends as many messages to WMQ as there are distributions in a record. WMB then reads the messages from WMQ and sorts them into their correct incoming queues inside the broker. The sorted messages are then transformed according inbuilt logic and passed to outgoing queues inside WMB. All these functions inside WMB are done by dedicated message flows. Those flows determine what is done to certain messages and all enterprise applications have their own flows in WMB. The modified messages are then distributed to enterprise applications using whatever technology. The most common solution is to use WMQ between WMB and enterprise applications. In that case,

WMQ reads the messages from WMB's outgoing queues and enterprise applications pick them up from their dedicated WMQ.

When implementing new GCM integration, the first step is to set up incoming and outgoing queues inside the broker. Their naming follows the same rules as the interface instance naming in GCM. The difference with the outgoing queue is that it ends with OUT instead of IN. Another task is to create message flows that are used to receive, transform and forward messages. There are four types of message flows in use: main flows, sub flows, mapping flows, and service flows. An example of message flows is in the integration specification document model which is available as appendix B.

An enterprise application is the third involved party in an integration project. Even though most of the integration solution is standardized, there are still some selections that need to be made on the basis of the enterprise application. The complexity of integration is heavily affected by its capabilities. If the enterprise application supports XML messages, then it is enough just to install WMQ client on its application server and to use message queuing. If it needs messages in another format, then a flow has to be created to change the message type to suit the needs of the enterprise application. However, the connection can be established using ODBC for example. These decisions have to be made in the design phase already but the point is that the enterprise application has to provide an interface for WMB to distribute master data. That is the main task to be completed from the side of the application layer.

Integration architect handles the tasks in the data repository layer in cooperation with GCM key user. The integration layer is the specialty area of integration developer. He/she is able to organize message queues and message flows to match the integration design. The application layer is mastered by the enterprise application developer who knows how to create an interface towards WMQ or other connecting technology.

5.6 Testing

In addition to documentation, testing is another project step which is often not paid enough attention to. Whereas documentation is needed to enable maintainability and modifiability of an integration solution, testing ensures the current functioning of the solution. One has to be sure that all features of integration are functioning properly before it can be introduced. There are three different environments in which the functioning has to be tested: development environment, testing environment, and production environment.

The development environment is meant to be used by the integration project team to develop and unit test integration solutions. The governance of the development environment is relatively loose and it can be used for testing and prototyping new ideas and approaches. Even though the working principles in the development environment are not that tight, the solution must be developed into its final design already at this point. Only then it can be transported into environments that are higher in the hierarchy. By default, all performance testing should take place in the development environment,

The testing environment is primarily used for functional acceptance testing of integration solutions. The governance of the testing environment is tighter, and the namespaces and repositories should only contain real objects. In long term, the testing environment should be a copy of the production environment. This is to guarantee the testing environment corresponds exactly to the circumstances of the production environment.

The production environment is only used for running production-usage integrations. At this point, the integration solution has already been found out to be work as expected. However, to be absolutely sure, the solution must be extensively tested in the production environment too. The environment is strictly governed and only accessible by certain persons.

The purpose of GCM integration is to distribute master data to enterprise applications across the whole enterprise. This is also the starting point of the testing step. When thinking of what tests need to be performed to ensure the integration works properly, the possible use cases can be considered. At least,

the following scenarios have to be gone through in GCM user interface when testing the integration solution:

- Create a customer record.
- Create a supplier record.
- Create a record with both customer and supplier distributions.
- Add a customer distribution into existing record.
- Add a supplier distribution into existing record.
- Modify all Core tab fields of an existing record.
- Modify all Postal tab fields of an existing record.
- Modify all Hierarchy/D&B tab fields of an existing record.

The above list describes just the basic use cases. There is much more beyond them and testers have to pay attention to all little details that matter the most. Are the special characters transferred correctly? Is the number of characters limited in some fields? Does the delay stay in predicted level? These are just to mention few possibilities. Testing is done to find hidden flaws and to make sure they are eliminated. All possible failures cannot be expected and taken care of in the design and implementation steps.

Testing takes place in all three involved layers. The integration architect helps the GCM key user to enter information into GCM and to make sure everything is carefully tested. The integration developer is in charge of the integration layer to see if something goes wrong with WMB or message queues. The application owner and the application developer are the ones who monitor the enterprise application.

5.7 End-User Training

At least two local administrators are nominated for each enterprise application installation to create and update trading partner records in GCM. There

are own administrators for customer and supplier records. The number of administrators depends on the amount of information to be fed in. The best way to affect the quality of data is to devote to end-user training. Unless the administrators know how to maintain consistent and complete master data, there is too much to be cleansed afterwards. End-user training sessions are arranged on a regular basis. Still, when an integration solution is about to be launched, it is essential to train all administrators of that particular enterprise application once again. It reminds them of the importance of paying attention to every little detail in GCM data input process.

In practice, end-user training is arranged using Konecranes' internal video-conferencing equipment, or if there is not one available, Microsoft Net-Meeting does the same thing. Sometimes, it is reasoned to have on-the-spot training session in some of Konecranes' local offices. In such situation, for example, when it is possible to assemble large number of local administrators in one place at the same time. Integration related end-user training always takes place just before taking the solution into production environment. There can be several sessions during the last week of the integration project to make sure everyone is able to participate within the limits of their schedules. Also, the difference in time has to be considered.

GCM key user arranges the training sessions in cooperation with the integration architect and the enterprise application owner. Together, they cover all three layers that are involved in the integration solution. When a new integration solution is introduced, a typical training session consists of such subjects as why MDM is needed in the first place, why the administrators' work is important, how GCM is linked with corporate decision-making, and, of course, how master data is correctly maintained following predefined processes. Naturally, the administrators have to have enough knowledge to create and update records in GCM. However, it is equally important to motivate them to take the job seriously. There are several tasks that may seem insignificant at first sight but actually contribute to the data quality notably. Such tasks include merging duplicate records, maintaining organizational hierarchies, and specifying address information.

5.8 Launch

Launching an integration solution does not require much concrete work. Almost all hands-on tasks have been taken care of in the previous project steps. The solution has been taken into production environment and once more tested to be flawless. The only thing is to send an email to the administrators of that particular enterprise application to announce the exact time when the integration is planned to go live.

Launch step includes also being ready to take action if something goes wrong after all. If the problem is found out to cause too much trouble to continue, the project must be rolled back to the previous phase. This means basically postponing the integration introduction and to keep on testing to solve the problem. Documentation comes in handy in this situation when developers try to find out what has gone wrong in the design or implementation step.

After the integration solution is successfully launched, the system maintenance begins. It is an ongoing process that includes regular message and system monitoring, end-user training, and support towards the local GCM administrators.

6 DISCUSSION AND CONCLUSIONS

The main objective of this study was to create a roadmap for integrating the global master data application of Konecranes with various existing enterprise applications within the company. Integrations are needed to enable distributing solid and consistent master data across the whole organization. Until now, there have not been common guidelines on how to manage an integration project. The implementation method of integration solution always depends on the requirements of connected parties. Thus, it was not realistic to aim at creating specific instructions for all possible scenarios. Instead, the purpose was to outline general directions that support the development of each individual integration solution. The integration roadmap is premised on the basis of eight steps. These steps are to be followed in order to successfully integrate stand-alone applications with each other. The steps are divided into time period of six weeks. The timescale is drawn up loosely by design to make it possible to stretch some phases in the case of problem situations.

In addition to the main objective, there were three secondary objectives to be fulfilled within the limits of this study. The first secondary objective was to give an overview of application integration. This subject is not covered in any particular section but throughout the entire study. The intention is to provide the reader with general impression on why integrations are needed in the first place, what integrations are used for, and what the most typical ways of implementing integrations are. The emphasis is particularly on master data application integration to introduce the special characteristics of such integration solution in which a master data application is involved. Another secondary objective concerned application integration patterns and technologies. Fulfilling this objective forms a major part of the theoretical section of this study. Often, integration technologies have been chosen case-specifically without considering the big picture. This may be due to the lack of knowledge of available options. It easily leads to fragmented integration environment and point-to-point integration development. Introducing alternative approaches makes it easier to choose integration patterns and technologies that support the organization’s integration architecture and centralized structure.

The third and final secondary objective was to represent the current platform architecture of Konecranes. It is crucial to understand what has been done in the past and what the environment looks like at the moment. Illustrating the platform architecture also helps to see how the apparently disconnected layers are actually interconnected with each other. Besides the integration roadmap, representing the platform architecture forms the other half of the empirical section of this study. The architectural layers and their purposes are discussed as well as how the layers are connected with the other ones.

In total, there were one main objective and three secondary objectives set for this study. Creating the integration roadmap was the most important task of all. The outcome is an eight-step guideline on how to organize an integration project. Most steps are described in general level although some parts are described in more detail. For example, the implementation step is written giving some specific instructions on how to modify certain configurations. On the other hand, design step only advices what aspects need to be considered. Only one pilot enterprise application has been integrated with the master data application so far. If there was more experience in such integrations, it could have helped improving the roadmap. Now, it mostly remains a high-level description of the phases of integration project. Improving it would require more concrete approach to the subject.

On average, the secondary objectives were reached quite well. The biggest improvement would be needed in focusing the theoretical sections on master data application integration in particular. The theory is written in too general level, not concentrating enough on the essential parts. Still, the theory extensively covers the most common integration techniques and proves the importance and versatile uses of application integrations. Maybe the most successfully fulfilled objective was representing the current platform architecture of Konecranes. There are plenty of system documentations available in the company but this one clarifies the complicated environment into understandable format. Even though the representation is has been done in simplified manner, it still includes exact information.

This study can be used for several purposes in Konecranes Group IT. It acts as a text book about application integration patterns and technologies as well as a basic guide to BI, DW, and MDM. While the theoretical merits of the study are unarguable, the most valuable information lies in the latter part. Examining the platform architecture representation is an easy way to become familiar with the system environment of Konecranes. The integration roadmap helps integrating internal enterprise applications with GCM. It provides a standardized way to enable distributing organization's core information to all interested parties therefore improving data quality across the company.

REFERENCES

Davies, S., Birkler, K., Heite, R., Klein, U., Mäkelä, M., Matsuki, C.A., Parizek, P., Sall, A., Schefencaker, S., Sulzmann, R., and Wilms, T. (2007) *Connect WebSphere Service-Oriented Middleware to SAP.* http://www.redbooks.ibm.com/redbooks/pdfs/sg247220.pdf (Accessed Aug 13, 2009)

Dreibelbis, J., Hechler, E., Milman, I., Oberhofer, M., van Run, P., and Wolfson, D. (2008) *Enterprise Master Data Management: An SOA Approach to Managing Core Information*. US: IBM Press.

Gable, J (2002) *Enterprise Application Integration* http://findarticles.com/p/articles/mi_qa3937/is_200203/ai_n9019202/ (Accessed Jul 8, 2009)

Hohpe, G. and Woolf, B. (2003) *Enterprise Integration Patterns: Designing, Building, and Deploying Messaging Solutions*. US: Addison-Wesley.

Hovi, A., Hervonen, H., Koistinen, H. (2009) *Tietovarastot ja Business Intelligence.* Porvoo: WS Bookwell.

Kanis, P (2004) *The Asynchronous- Synchronous War: Much Ado about Nothing*. http://www.addc.com/articles/sync-async%20wars.pdf (Accessed Jul 8, 2009)

Konecranes Corporation (2009) Annual Report 2008. http://www.konecranes.com/attachments/investor/annual_reports/annual_report_2008/konecranes_annualreport2008.pdf (Accessed Jun 26, 2009)

Linthicum, D.S. (2000) *Enterprise Application Integration*. US: Addison-Wesley.

Linthicum, D.S. (2003) *Next Generation Application Integration: From Simple Information to Web Services*. US: Addison-Wesley.

Microsoft Corporation (2003) *Guidelines for Application Integration.* http://download.microsoft.com/download/8/d/a/8daa6e7f-eee5-4e85-b8fd-14d72bd00729/EAIArch.pdf (Accessed Jul 14, 2009)

Oracle Corporation (2008) *Better Information through Master Data Management – MDM as a Foundation for BI.* http://www.oracle.com/master-data-management/better-information-through-master-data-management-mdm-as-a-foundation-for-bi.pdf (Accessed Jul 14, 2009)

Pinus, H. (2004) *Middleware: Past and Present a Comparison*. http://userpages.umbc.edu/~dgorin1/451/middleware/middleware.pdf (Accessed Jul 8, 2009)

Ruh, W.A., Maginnis, F.X., and Brown, W.J. (2000) *Enterprise Application Integration: A Wiley Tech Brief.* US: John Wiley & Sons.

Thierauf, R.J. (2001) *Effective Business Intelligence Systems*. US: Quorum Books.

Tieturi Oy (2009) *Järjestelmäintegroinnin valmennusohjelma 4.0* Training course and material.

Tähtinen, S (2005) *Järjestelmäintegraatio: tarve, vaihtoehdot, toteutus*. Jyväskylä: Talentum Media Oy.

White, C (2005) *Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise.* http://download.101com.com/tdwi/research_report/DIRR_Report.pdf (Accessed Jul 21, 2009)

APPENDICES

Appendix A Sample of an XML Message

```
<?xml version="1.0" encoding="UTF-8"?>
<SYNC_TRADINGPARTNER_040>
  <ROW>
    <TPLANG>en</TPLANG>
    <TPAD2NUM>1</TPAD2NUM>
    <TPAD7VAR></TPAD7VAR>
    <TPSTATUS>ACT</TPSTATUS>
    <TPNAME>Model Company</TPNAME>
    <TPDUNSNO></TPDUNSNO>
    <TPAD1NUM>1</TPAD1NUM>
    <TPGCMID>1425147</TPGCMID>
    <TPORGNO>845647588</TPORGNO>
    <TPVATREGNO>12548534</TPVATREGNO>
    <TPSHNAME></TPSHNAME>
    <TPAD2TIM></TPAD2TIM>
    <TPGLULGCM></TPGLULGCM>
    <TPGLULDUNSNO></TPGLULDUNSNO>
    <TPDOULGCM></TPDOULGCM>
    <TPDOULDUNSNO></TPDOULDUNSNO>
    <TPLEPAGCM></TPLEPAGCM>
    <TPLEPADUNSNO></TPLEPADUNSNO>
    <TRADPAADR_TD>
      <ROW>
        <TDADRTYP>BY</TDADRTYP>
        <TDPHONE>5418646163</TDPHONE>
        <TDADRNO>1</TDADRNO>
        <TDNAME></TDNAME>
        <TDCTRY>US</TDCTRY>
        <TDADR1>Testing Street 10</TDADR1>
        <TDADR2></TDADR2>
        <TDCITY>Test City</TDCITY>
        <TDSTATE>021</TDSTATE>
        <TDZIP>52294</TDZIP>
      </ROW>
      <ROW>
        <TDADRTYP>DP</TDADRTYP>
        <TDPHONE></TDPHONE>
        <TDADRNO>1</TDADRNO>
        <TDNAME></TDNAME>
        <TDCTRY>US</TDCTRY>
        <TDADR1>Testing Street 10</TDADR1>
        <TDADR2></TDADR2>
        <TDCITY>Test City</TDCITY>
        <TDSTATE>021</TDSTATE>
        <TDZIP>52294</TDZIP>
```

```
      </ROW>
    </TRADPAADR_TD>
    <TRADPARREL_TL>
      <ROW>
        <TLASRTR>0</TLASRTR>
        <TLRELTYP>SUPP</TLRELTYP>
        <TLCUSUNO>8798546</TLCUSUNO>
        <TLSHWATP>ATP</TLSHWATP>
        <TLAD1NUM>0</TLAD1NUM>
        <TLSUCUNO>US_SPR_WEN1</TLSUCUNO>
        <TLAD3VAR></TLAD3VAR>
        <TLINPATY>CUST</TLINPATY>
      </ROW>
      <ROW>
        <TLASRTR>0</TLASRTR>
        <TLRELTYP>CUST</TLRELTYP>
        <TLCUSUNO>FR_VER_ILM1</TLCUSUNO>
        <TLSHWATP>ATP</TLSHWATP>
        <TLAD1NUM>0</TLAD1NUM>
        <TLSUCUNO>48646841</TLSUCUNO>
        <TLAD3VAR></TLAD3VAR>
        <TLINPATY>CUST</TLINPATY>
      </ROW>
    </TRADPARREL_TL>
  </ROW>
</SYNC_TRADINGPARTNER_040>
```

Appendix B Sample of an Integration Specification Document

KONECRANES®
Lifting Businesses™

INTEGRATION SPECIFICATION
GCM – Enterprise Application

Current Version:	1.0
Owner:	Konecranes
Date Last Updated:	19.08.2009
Last Updated By:	Application Developer
Author:	Integration Developer
Date Created:	03.05.2009
Reviewed By:	Juhana Murtola
Approved By:	Mikko Strömberg
Approval Date:	22.08.2009

Revision History

Version Number	Date Updated	Revision Author	Brief Description of Changes
0.1	03.05.09	Integration Developer	Initial version
1.0	19.08.08	Application Developer	Version for review

Referenced Documents

Reference Number	Document Name	Version	URL/File Name etc Target Source
1	Operational Handbook	0.7	Operational_Handbook.doc
2	Architectural Overview	1.0	Architectural Overview.doc
3	Governance Model	0.9	Governance_Model.doc

KONECRANES®
Lifting Businesses™

Table of Contents

KONECRANES®
Lifting Businesses™

1. Introduction

1.1 Background

This documentation has been gathered to update and document the current situation of Konecranes integration layer integrations. It also works as template for new integrations to be developed.

1.2 Document Purpose

Integration Specification for GCM – Enterprise Application brings together all the information necessary for transition, testing, and further development of integration concerning the integration between GCM and Enterprise Application including the interfaces of GCM and Enterprise Application.

The purpose of this document is to function as a detailed integration service specification for maintenance, development, testing, bug fixing, and to:

- Describe the high-level structure of the integration
- Describe the responsibilities, relationships, and interactions of components
- Explain how application/technical parts of the system are related
- Specify how existing, acquired, and developed components are related
- Define the components that have to be placed on the operational model, that is, that have to execute and be managed on the target platforms
- Help organizing the development project
- Reduce complexity through the encapsulation offered by a component

1.3 Audience

The target audience for this document includes Konecranes IT managers, integration architect, integration developer, enterprise application owner, and enterprise application developer.

1.4 Scope

The scope of topics covered in this document is to present specific technical information about the integration solution between GCM and Enterprise Application.

1.5 Notation

Term	Description
GCM	Global Company Master
Enterprise Application	A model application
WMB	IBM WebSphere Message Broker
MQ	IBM WebSphere MQ

KONECRANES
Lifting Businesses

2. System Overview

The integration includes the following systems:

- GCM
- Enterprise Application

This section documents each back-end system and their features.

2.1 General Info of Enterprise Application

Back–end System Name	Enterprise Application
Contact Person/Owner	Application Developer
System Description	A model application
System Role	An enterprise application in some Konecranes business unit

2.2 General Info of GCM

Back–end System name	GCM
Contact Person/Owner	Key User
System Description	The global company master data solution of Konecranes
System Role	To store and share company master data

2.3 General info of Integration Layer

Back–end System Name	Konecranes Integration Layer
Contact Person/Owner	Integration Developer
System Description	The integration Layer of Konecranes
System Role	To enable integrations between the layers of Konecranes platform architecture

KONECRANES
Lifting Businesses

3. Summary of Interfaces

Name of interface	From using technology	Format	To using technology	Format	Description
GCM to Enterprise Application	MQ	.xml	MQ	.xml	Master data from GCM to Enterprise Application

4. Integration Configurations

4.1 Installed Software

The following products have to be installed and started:

DB2 Enterprise Database version 8.2
WebSphere MQ Server version 6
WebSphere Message Broker version 6.0.0.3
WebSphere Message Broker version 6.0.0.3 bug fix(IC49793)

4.2 MQ Channel Definitions

For GCM integration, a Server-Connection Channel is needed. The Server-Connection's name is used in all clients that connect to the Queue Manager.

Channel definition
Using WebSphere MQ Explorer select Queue Manager's Channels node, right click and create Server channel with the name “FI_HVK_ENT1.SVR”. Using command line (example from test environment): 1. runmqsc KONECRANES_TEST 2. DEFINE CHANNEL(FI_HVK_ENT1) CHLTYPE(SVR)

4.3 MQ Process Definitions

Process explanation	Process definition
Process for reading queue FI_HVK_ENT1	PR.FI_HVK_ENT1

KONECRANES®
Lifting Businesses™

5. MQ Queue Definitions

Within the Queue Manager, create the queues required for the integration. Queues creation script is in the distribution package in mq/populate.conf. Queues can be created manually via WebSphere MQ Explorer, however the recommended way is to use command line:
runmqsc KONECRANES_TEST <populate.conf

Queue definitions
MQ is the default protocol for the data interchange for the solution. The following queues are defined: • KCI.FI_HVK_ENT1.IN • KCI.FI_HVK_ENT1.OUT

5.1 MQ Security Definitions

There are no specific security recommendations, or implemented security features in the Konecranes integration layer at this time.

5.2 Flows Used in This Integration

Main Flows:

MAINFLOW_KCI.GCM.FI_HVK_ENT1.IN
MAINFLOW_KCI.FI_HVK_ENT1.OUT

Sub Flows:

SUBFLOW_DISPATCHER.msgflow
SUBFLOW_SWITCH_MSG_TYPE.msgflow
SUBFLOW_ERROR_NOTIFICATION.msgflow
SUBFLOW_MONITOR_DB.msgflow

Map Flows:

MAPFLOW_GCMTOENT.msgflow

Service Flows:

SERVICEFLOW_EMAIL.msgflow
SERVICEFLOW_MONITOR_QUEUES.msgflow

5.3 Message Sets Used in This Integration

Message sets:

messageSet.mset

KONECRANES
Lifting Businesses™

6. Integration Implementation

The basic idea is to transfer a file (.xml) generated by GCM to Enterprise Application environment. The transfer procedure goes as follows:

- GCM sends an xml file to MQ (KCI.GCM.FI_HVK_ENT1.IN).
- WMB reads the file from the queue and makes necessary transformations.
- WMB send the file to MQ Queue (KCI.FI_HVK_ENT1.OUT)
- Enterprise Application reads the file into its own database

6.1 Main Flows

The integration solution uses message queues to determine the message's source system and to fetch the correct destination for the message. Both Enterprise Application and GCM uses the message queues for data interchange.
Therefore the integration solution uses at least two message queues for Enterprise Application. The queues are:

- Incoming queue from GCM to receive messages addressed to Enterprise Application.
- Outgoing queue to Enterprise Application instance to send GCM originated messages addressed to Enterprise Application.

6.2 Sub Flows

The group of message flows which are invoked from any type of flows is called sub flows. A sub flow can be invoked from all sub flows except itself. That is because of the nature of sub flows: they are copied into flow-invoker.

6.3 Map Flows

Map flows deal with Enterprise Application specific transformations. However, that is only for a one-way transformation. Thus, if an enterprise application uses two-way communication with GCM, then two map flows are required.

6.4 Service Flows

Service flows are autonomic message flows which are not directly dependent on the general framework and processing flows. Certain functionality is implemented by service flow which usually involves interaction between a message queue and external data source (database, file system, e-mail).

KONECRANES
Lifting Businesses

7. Routing Implementation

This section describes the integration specific values and implementation in DB2 database.

Integration data is stored in a DB2 database with MQ_GCM alias under ROUTE namespace. The data is used to map incoming queue name to source message type, target message type, target message queue and complimentary data (email-address, etc.).

Table INTEGRATION: specifies relations between queue and message type. Additional information like e-mail address is also stored in the table. The columns are:

Column name	Column type	Column description
KEY_INTEGRATION	INTEGER	The primary key for the record
SENDER_CODE	VARCHAR2(14)	The name of incoming queue
MESSAGE_TYPE	VARCHAR2(48)	The type of incoming/outgoing message format. Label name is created using <incoming message type>_TO_<outgoing message type>
MAIL_ADDRESS	VARCHAR2(40)	The Email address if, accordingly to business logic, the destination is e-mail, not MQ queue
APP_SPEC	VARCHAR2(48)	The logical id of an enterprise Application (FI_HVK_ENT1 for example)

Table `ROUTE` specifies many to many mapping between incoming and outgoing queues and message types. The table's columns are:

Column name	Column type	Column description
KEY_INTEGRATION	INTEGER	The ID of a row in INTEGRATION table which corresponds to incoming data
KEY_ROUTING	INTEGER	The ID of a row in INTEGRATION table which corresponds to outgoing data
QUEUE_MANAGER	VARCHAR2(48)	The name of queue manager where outgoing queue resides in
QUEUE	VARCHAR2(48)	The name of outgoing queue

View `V_ROUTE` aggregates `senders (INTEGRATION)` and `receivers (INTEGRATION)` using `ROUTE`. The view's columns are:

Column name	Column relation
MESSAGE_TYPE_INTEGRATION	SENDER.MESSAGE_TYPE
MESSAGE_TYPE_ROUTE	RECEIVER.MESSAGE_TYPE
SENDER_CODE_INTEGRATION	SENDER.SENDER_CODE
QUEUE_MANAGER	ROUTE.QUEUE_MANAGER
QUEUE	ROUTE.QUEUE
MAIL_ADDRESS_ROUTE	RECEIVER.MAIL_ADDRESS
APP_SPEC	RECEIVER.APP_SPEC

Table MAIL_MAP maps local system, country, and region to system administrator's e-mail address. It is used when GCM sends an update of company information and the update has to be sent both to message queue and to local administrator's e-mail. The table's columns are:

Column name	Column type	Column relation
LOGICAL_ID	VARCHAR2(48)	The logical id of a system (FI_HVK_ENT1 for example)
COUNTRY	VARCHAR2(48)	The country code of a local administrator
MAIL_ADDRESS	VARCHAR2(48)	E-mail address of a local administrator

8. Sequence Diagrams

8.1 IN Interfaces

No IN interfaces (messages to GCM's direction) are included in this integration.

8.2 OUT Interfaces

Sequence diagram/tables are used for giving the developer a quick view on the whole end-to-end integration chain. The diagram/tables should include following information:

- Sending system
- MQ queue (sender)
- WMB flows for message
- MQ queue (receiver)

9. Deployment Guidelines

In GCM integration, deployment into Konecranes environment follows the following configurations:

Instance	Instance value
Server-Connection Channel	ENT1_IN
Database instance	MQ_GCM
ODBC Source	MQ_GCM
Database schema	ROUTE
Database tables	INTEGRATION, ROUTE, V_ROUTE, MAIL_MAP
Execution groups	ENT_APP, GCM_MAINFLOWS

KONECRANES
Lifting Businesses™

10. Agreed Policies for Development

There are some common policies regarding the development of GCM related integration solutions. Those policies are introduced in this section.

10.1 Documentation of Development Projects

All integration development projects use this document template for documenting new integrations. This is in order to secure consistent documentation and best practices in all integration development.

10.2 Naming of MQ Queues

By default, all data interchange with WMB goes through message queues. Thus, each enterprise application deals with multiple queues. Each queue is dedicated to certain enterprise application. This means that no queues are used by multiple enterprise applications for data interchange with GCM. It is very important to specify unique and understandable names for each queue to facilitate system administration and support. In case of new enterprise application integration, at least two queues are added to WMB:

1. KCI.GCM.FI_HVK_*ENT1*.IN incoming queue from GCM to receive messages addressed to Enterprise Application
2. KCI.*FI_HVK_ENT1*.OUT outgoing queue to Enterprise Application to send GCM originated messages addresses to Enterprise Application

Besides the Enterprise Application and GCM queues, there are also queues that are used by WMB:

1. KCI.GCM.OUT message queue is used to pass messages from all enterprise applications to GCM.
2. KCI.EMAIL.QUEUE message queue is used to receive messages from WMB. Special service flow processes this queue and emails its content to system administrator.
3. KCI.DISK.QUEUE message queue is used to receive messages from WMB. Special service flow processes this queue and saves its content to the file system.
4. KCI.ERROR.QUEUE message queue is used to receive messages from WMB. Special service flow processes this queue and emails its content to system administrator.

KONECRANES®
Lifting Businesses™

11. Test Methods

If possible, include input and output file used for testing, as well as location where these files can be found. Minimum amount of test cases have been considered to be one test case per integration per use case. For CGM integration, the use cases are:

- Create a customer record.
- Create a supplier record.
- Create a record with both customer and supplier distributions.
- Add a customer distribution into existing record.
- Add a supplier distribution into existing record.
- Modify all Core tab fields of an existing record.
- Modify all Postal tab fields of an existing record.
- Modify all Hierarchy/D&B tab fields of an existing record.

12. Error Processing

Possible error causes include:

- GCM operation process error
- WMB to Enterprise Application link does not function
- WMB error
- Message format error

Available error handlers are:

- Default handler: An error message is returned using response channel (queue, web service, HTTP listener).
- Reserve handler: Used when link to Enterprise Application does not function or Enterprise Application does not support error information. In this case, any other way of communication must be used.

There is an integration monitoring solution available. Broker flows include settings for Tivoli monitoring integration specific flows.

Appendix C Definitions

Application Programming Interface (API)

An interface that enables different applications to communicate with each other.

Asynchronous Communication

Communication by which sending and receiving applications do not need to be available simultaneously.

Business Intelligence (BI)

A process for exploring and analyzing information to discern business trends or patterns, thereby drawing conclusions.

Business Process Integration-Oriented Application Integration (BPIOAI)

Approaching application integration by controlling information flow and service invocation through a business process.

Call Level Interface (CLI)

A programming interface to access several different databases.

Composite Application

A composite application has the appearance of a single application but is, in fact, composed of multiple, independently designed applications.

Data Warehouse

A storage architecture designed to hold data extracted from enterprise applications and other external sources.

Data Warehousing (DW)

The process of managing data warehouses.

Enterprise Application

A software product designed to take care of some core operation of an organization, such as sales, accounting, or manufacturing.

Enterprise Application Integration (EAI)

Technologies that allow the exchange of information between different applications within an organization.

Extensible Markup Language (XML)

A standard for defining descriptions of structure and content in documents. Provides context and gives meaning to data.

Extract, Transform, Load (ETL)

Tools for extracting data from one data store, transforming the structure and content of this data, and loading the transformed data to another data store.

Global Company Master (GCM)

Konecranes' master data application that includes information about the customers and suppliers of the company.

IBM WebSphere Message Broker (WMB)

An information broker that allows business information to flow between disparate applications across multiple hardware and software platforms.

Information-Oriented Application Integration (IOAI)

An approach to application integration where the source and target systems exchange information in real time.

Integration Hub

A middleware model that provides centralized communication method between applications.

Java Database Connectivity (JDBC)

A CLI programming interface that provides connectivity between Java platform and a range of database management systems.

Master Data

The core information for an organization, such as information about customers, suppliers, or products.

Master Data Management (MDM)

An approach to reducing the amount of redundantly managed information, and providing information consumers with master data.

Message-Oriented Middleware (MOM)

Middleware for connecting applications, most commonly through the use of message queuing.

Middleware

Software that facilitates the communication between applications.

Open Database Connectivity (ODBC)

A vendor-neutral CLI programming interface to access database management systems.

Point-to-Point

A decentralized middleware model that consists of individual communication solutions between two parties.

Portal-Oriented Application Integration (POAI)

Approaching application integration by aggregating the information contained in many back-end systems within a portal.

Remote Procedure Call (RPC)

A mechanism that extends the notion of local application procedure calls to a distributed computing environment.

Service-Oriented Application Integration (SOAI)

The process of joining applications together by allowing them to share services between them.

Synchronous Communication

A form of communication that requires the sending and receiving applications to be running concurrently.