Aalto University
School of Science
Department of Industrial Engineering and Management

Merina Maharjan

Enabling Closed Loop Lifecycle Management with Information Exchange Standards

Master's Thesis submitted in partial fulfilment of the requirements for the degree of Master of Science in Technology.

Espoo, May 20, 2013

Supervisor:	Professor Jan Holmström
Instructor:	Professor Kary Främling

AALTO UNIVERSITY SCHOOL OF SCIENCE

ABSTRACT OF THE MASTER'S THESIS

Author:	Merina Maharjan
Name of the Thesis:	Enabling Closed Loop Lifecycle Management with Information Exchange Standards
Date:	May 20, 2013
Number of pages:	x + 81
Department:	Department of Industrial Engineering and Management
Professorship:	TU-22 Service Management and Engineering
Supervisor:	Professor Jan Holmström
Instructor:	Professor Kary Främling

Today, the manufacturing industry is focusing on the customer oriented service. This requires appropriate product lifecycle knowledge in order to facilitate product servicing and the design of next generation of products. A concept called Closed Loop Lifecycle Management (CL_2M) is developed as an extension to traditional Product Lifecycle Management (PLM). CL_2M enables the desired information gathering, processing and exchange throughout the whole life of an entity from beginning, through middle to end of life. A key challenge for the implementation of CL_2M is when the information is distributed; a standard becomes essential concerning the format of data and way to exchange it. The objective of this thesis is to define such a messaging standard that aids in seamless information flow and exchange of information in CL_2M.

The Quantum Lifecycle Management (QLM) messaging standard is currently being developed. The standard is derived from PROMISE Messaging Interface (PMI) developed in the PROMISE EU project in 2008. In this thesis, the power consumption monitoring application example presents the implementation of QLM messaging standard. During the study, similar standards that are already exist is found, and compared with QLM. Furthermore, two application cases have been addressed in the thesis to enable information flow visibility and exchange in a real manufacturing scenario. QLM is developed for Internet of Things (IoT) as an information exchange standards for information flow between any kinds of intelligent products, devices, users and information systems and to close the information loop in CL_2M.

Keywords: Product lifecycle, Product Lifecycle Management, Closed Loop Lifecycle Management, messaging standards, QLM messaging, PROMISE.

Acknowledgments

This Master's Thesis has been written as a partial fulfilment for the Master of Science (Technology) degree at the Aalto University School of Science, Finland. The work had been carried out being a part of the European Union project (LinkedDesign) in the Business, Innovation, Technology (BIT), Department of Industrial Engineering and Management (DIEM) at the Aalto University School of Science.

I would like to express my gratitude to all the people who have supported and encouraged me in this work. First of all, I want to thank my supervisor Professor Jan Holmström for giving me the opportunity to work under his guidance. Second, I am very grateful to my instructor Professor Kary Främling for his enthusiastic engagement in my thesis and his never ending support, motivation, and guidance throughout the whole research process.

I would also like to thank my colleagues from the LinkedDesign; Andrea, Silvia, Stefano and Jörg for helping in gathering the information for this thesis. Furthermore, I wish to thank Saara, Sylvain and the Logistic Research Group for helping me with the thesis writing. Sylvain and Prajwal should have the credit of proof reading my thesis and helping me with the structure.

I owe my deepest gratitude to my parents Prem Dass and Swarna Lata, sisters Mamata and Matina, and my best friend Supriya for giving me all the love and wisdom in every step of my life. Finally, I would also like to share my thanks to Aashish, Rajendra, Prem, Manik and the entire Nepalese community at the Aalto University for their support and never ending motivation.

Espoo, May 20, 2013
Merina Maharjan

Contents

Abbreviations and Notations

API	Application Programming Interface
BOL	Beginning of Life
CAD	Computer Aided Design
CAM	Computer Aided Manufacturing
CL_2M	Closed Loop Lifecycle Management
DIALOG	Distributed Information Architecture for Collaborative Logistics
DSS	Decision Support System
EPCIS	Electronic Product Code Information Service
EOL	End of Life
ERP	Enterprise Resource Planning
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JMS	Java Messaging Service
LCC	LifeCycle Cost
MOL	Middle of Life
oBIX	Open Building Information eXchange
PDM	Product Data Management
PDKM	Product Data Knowledge Management
PEID	Product Embedded Information Device
PLC	Product Life Cycle
PLM	Product Lifecycle Management

PMI	PROMISE Messaging Interface
PROMISE	Product Lifecycle Management and Information Tracking Using Smart Embedded Systems
QLM	Quantum Lifecycle Management
REST	Representational State Transfer
RFID	Radio Frequency Identification
SCM	Supply Chain Management
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language
URL	Uniform Resource Locator

List of Figures

List of Tables

Chapter 1

Introduction

Synopsis

The motivation for this research is discussed in this chapter. We identify the research questions, clarify the objectives and define the framework of the study. Finally, we outline the structure of the thesis.

1.1 Motivation

There is a recent trend among manufacturing companies to gain a competitive edge over their competitors by providing ‘services’ rather than ‘products’ [1] [2]. Such services are for instance after-sales services, asset management, product lifecycle management and quality management. The increasing demand on product lifecycle management and quality management means that the information about products has to be easily accessible to all the actors involved during the product’s entire life time [3]. Therefore, it has become more important for manufacturing companies to understand and track how each product is built, used and behaved to enable more intelligent services, e.g., predictive maintenance, product lifecycle cost estimation, etc. [4]. Product related data is used by many people and systems for various purposes and at different locations. One needs to gather, manage and control the product information to use it effectively, no matter wherever the data is located or whoever possesses it [5].

The management of the Product Life Cycle (PLC) signifies managing the information generated from Beginning of Life (BOL) to Middle of Life (MOL) and End of Life (EOL). However, in the traditional Product Lifecycle Management (PLM) systems there is a gap in the exchange of information and knowledge between the different PLC phases. Usually, PLM systems focus on the BOL and often neglect the information generated in MOL and EOL. Moreover, there is no transfer of the information from one phase to another despite the advantages that such transfer could provide for the development of cost-effective logistics, maintenance and service [6]. According to Frey [7], PLM was necessary for automating the manufacturing process and operations planning developed during the 1970's under the names Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM).

Since CAD and CAM origins, the planning of manufacturing has depended on Information and Communication Technology (ICT) tools. Moreover, large manufacturers have a substantial amount of information to store about their products. Unfortunately, the manufacturer's database is not regularly updated with data about the use of the product. Some actors have their own database for their own service and maintenance information, and keep their own copies of data, which they maintain themselves. As a result, different groups of users often have different copies of what should be the same data that is not fed back to the manufacturer to develop new generations of products [5].

The issues of information exchange gap in BOL, MOL and EOL have been resolved by developing an extension to traditional PLM in the PROMISE EU project[1] (PROduct lifecycle Management and Information tracking using Smart Embedded systems) [8]. The extended PLM is called Closed Loop PLM and later renamed Closed Loop Lifecycle Management (CL_2M)[2]. The focus of this thesis is understanding the concept and technology of CL_2M.

Over the last decade, the development of product identification technologies such as Radio Frequency Identification (RFID) and Auto-ID have enabled products to have embedded information devices (e.g. RFID tags and on-board computers) making it possible to gather PLC data at any time and at any place [9, 10]. Thus, the whole PLC information can become visible and possibly controllable. Then, with the emergence of intelligent products, it is possible to

[1]project from 2004 to 2008: http://www.promise.no

[2]project from 2012: http://cl2m.com/. In this thesis, from now, we refer "Closed Loop PLM" concept by CL_2M.

monitor and capture the product's environment and to generate information related to product's status and performance [11]. The intelligent product can carry its full history with it and thus helps in service, e-maintenance, and recycling. This information is fed back to the manufacturer and the gap in the product information loop is closed. However, managing such product information is a challenging task, especially during MOL and EOL phases. The main difficulty is to maintain a reliable communication link between the product and its associated information as the product moves between different organizations and users.

There are other issues related with product information management as well [5]:

- *Data Availability*: Information must be available on the product's exact configuration, at any time and in any location.
- *Definition, Content, Vocabulary*: Many companies use different definitions and names for similar data items, which leads to errors, wasted time and money.
- *Security*: Product information is valuable and competitors should not have access to them. However, it is not easy to maintain the security of data while making it available for CL_2M purposes.
- *Communication Standard*: There is lack of standards for performing seamless information exchange between PLC phases, nor an agreed upon a standard for the format of data is available.

In addition to the introduction of CL_2M, the second main focus of this thesis is the development of a new information exchange standard, referred to as the Quantum Lifecycle Management (QLM) messaging standard, to address the issue of the *communication standard*.

1.2 Research objectives and framework

When PLM is distributed over organizations and users, it is necessary to develop an information exchange standard that specifies both the format of product data and the way to perform seamless information flow between the components involved in the PLC. The QLM messaging standard is studied in this thesis and is applied on an industrial scenario that comes from the EU project called *LinkedDesign*[3] [12]. This project is developing a software tool named LEAP (Linked Engineering and mAnufacturing Platform) to boost the productivity of today's engineers by providing an integrated, holistic view on data, persons and processes across the full PLC. This software is a vital resource for the competitive design of novel products and manufacturing processes.

The research questions that guide the course of the thesis work are:

1. How CL_2M can solve the various issues in traditional PLM systems?
2. What is the QLM messaging standard and how well does it meet CL_2M expectations?
3. How is the QLM messaging standard different to comparable existing messaging standards?

The above questions were raised from the identified limitations of the traditional PLM [13], together with the PROMISE project. To address these research questions, three guidelines are followed in this work:

- *Limitation discussion*: Based on a literature review, we study the PLM issues addressed by CL_2M. The study also constitutes an interview with the industry partners, which enabled to identify the most relevant issues.
- *PROMISE project*: We study the deliverable reports of the previous EU research project, PROMISE (2004-2008) so as to understand the PROMISE Messaging Interface (PMI). This serves as a basis for the developing of the new version of the messaging interface named the QLM messaging standard.
- *Comparison of the solutions*: We also compare different standards allowing information exchange throughout the PLC with the QLM messaging standard in order to identify whether the other standards could satisfy the CL_2M requirements as well.

[3]project from 2011 to 2015: http://www.linkeddesign.eu/

1.3 Structure of the thesis

The thesis consists of six chapters.

In Chapter 2, a literature review is carried out to point out the benefits that can be reached with PLM, followed by the identification of the main issues that remain to be addressed. Then, the PROMISE project and the CL_2M concept are respectively introduced.

Chapter 3 gives insight into the development of the QLM messaging standard and provides a comparison with the two other existing standards, namely: *Open Building Information eXchange* (oBIX) and *Java Messaging Service* (JMS).

Chapter 4 describes two industrial case scenarios. It also details how the QLM messaging standards is implemented in these scenarios.

Chapter 5 provides an overview on large-scale data management and analysis, and discusses the challenges that remain to be addressed. It also suggests ideas based on the open-source software *Hadoop*, which is used to improve the reliability, scalability and distributed computing.

Chapter 6 presents the contributions of this thesis, discusses the usefulness of CL_2M concept, and emphasizes the reliability and validity of the QLM messaging standard.

Figure 1.1 gives insight into the structure of this thesis from Chapter 1 to Chapter 6.

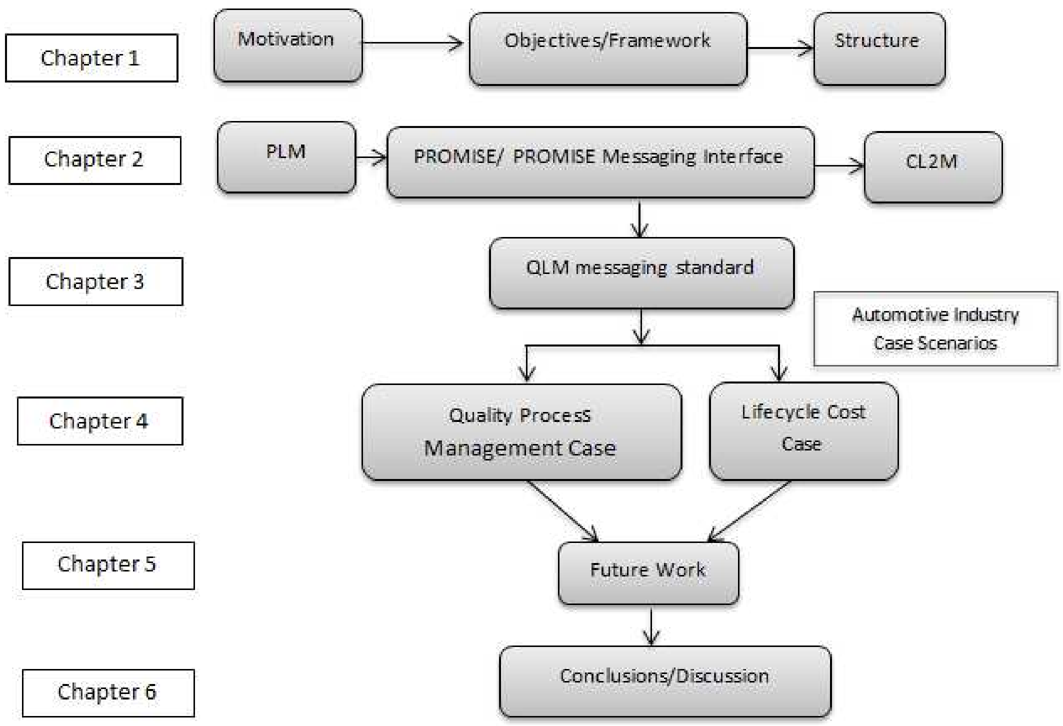


Figure 1.1: Structure of the thesis

Chapter 2

Closed Loop Lifecycle Management

Synopsis

This chapter is devoted to introduce CL_2M which has initially been defined and implemented in PROMISE project. First, based on a literature review, the benefits and issues related to the traditional PLM and the move towards CL_2M are discussed. Among the many issues in PLM, special consideration is given in this chapter to: information visibility, traceability, interoperability and communication standards. Then, the PROMISE project is introduced, followed by a brief description of the first investigations carried out on the message exchange interface. Finally, the business and technical architectures used in CL_2M are detailed.

2.1 Product Life Cycle

Product lifecycle or Product Life Cycle (PLC) may be seen in different ways according to the context of use of the product. Since 1960's, the PLC concept has been used in different areas such as product management, marketing mix, linking production processes and pricing, etc. [14], leading to different definitions according to the activity areas (Marketing, Resource, Manufacturer, User, Product involvement process) as illustrated in Figure 2.1.

In this thesis, the definition formulated by the CL_2M community in the

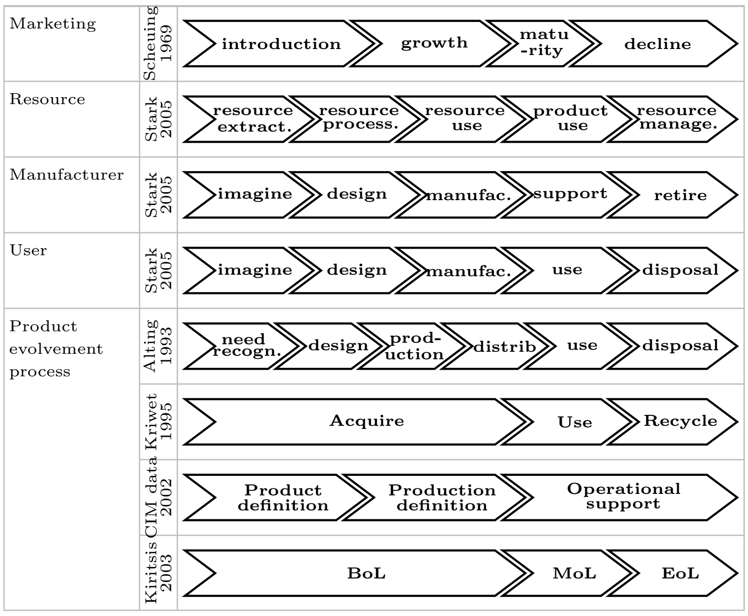


Figure 2.1: PLC definitions according to 5 activity areas

context of *Product evolvement process* is chosen, i.e., the one formulated by Kiritsis et al. [13] (see Figure 2.1).

2.2 Product Lifecycle Management

Product Lifecycle Management (PLM) is the process of managing the whole life cycle of a product beginning from generating an idea, concept description, business analysis, product design, solution architecture and technical implementation, to the successful entry to the market, service, maintenance and product improvement [1]. Thus, the main goal of PLM is to manage all the business processes and the associated data generated by events and actions of various lifecycle actors (both human and software systems) that are distributed along the PLC phases.

PLM originated from two types of management: enterprise management and product information management [15]. Enterprise management involves material resource planning (MRP), enterprise resource planning (ERP), customer relationship management (CRM), and supply chain management (SCM). Product information management involves Computer-Aided Design and Manufacturing (CAD/CAM) and product data management (PDM). However, at the end of 90's, PLM starts to evolve on system product management and product traceability to establish a new way of thinking for managing product data along the lifecycle [16]. PLM has been defined in many different ways by different vendors and scholars:

- PLM is a flow from BOL (Design, Production) to MOL (Sale, Use) and EOL (Disposal, Recycle) phases of the PLC. PLM is a means to transform information into knowledge and it improves quality, efficiency, product sustainability and service [13].
- PLM is an integrated business approach that uses information technology for enabling integrated, cooperative and collaborative information product management throughout the lifecycle [16].
- PLM is a holistic business activity addressing many components such as products, organizational structure, working methods, processes, people, information structures and information systems [17].
- PLM is a strategic business approach for the effective management and use of corporate intelligent capital [18].
- PLM is a system for managing all the activities required to deliver a product or service to the consumer from initial conception to manufacture, packaging, transportation and disposal [19].

Based on a literature review, the next section details both the benefits that can be achieved with the use of PLM systems and then, about the issues that remain to be addressed even after its use.

2.2.1 Benefits and issues

PLM is focused on "the product" and improves the activity of product development, without which a company will not survive as claimed by Stark [5]. The source of future revenues for a company is the creation of new products and services. Today, increasing competition in industries, is leading towards collaboration and information sharing with partners, customers and even competitors at the different phases of the PLC. Consequently, the ever growing amount of dispersed product data requires certain management practices and tools for successful capture, utilization and re-use of product. For this context, companies are using PLM to increase efficiency and consistency of product information throughout the PLC. Some authors argue that the product related information, data and knowledge management and sharing are the basic requirements for PLM [20, 21]. According to Stark [17]:

> *“PLM concept brings together products, services, structures, activities, processes, people, skills, ICT applications, practices, data, knowledge, procedures, standards, techniques and other activities as well as resources.”*

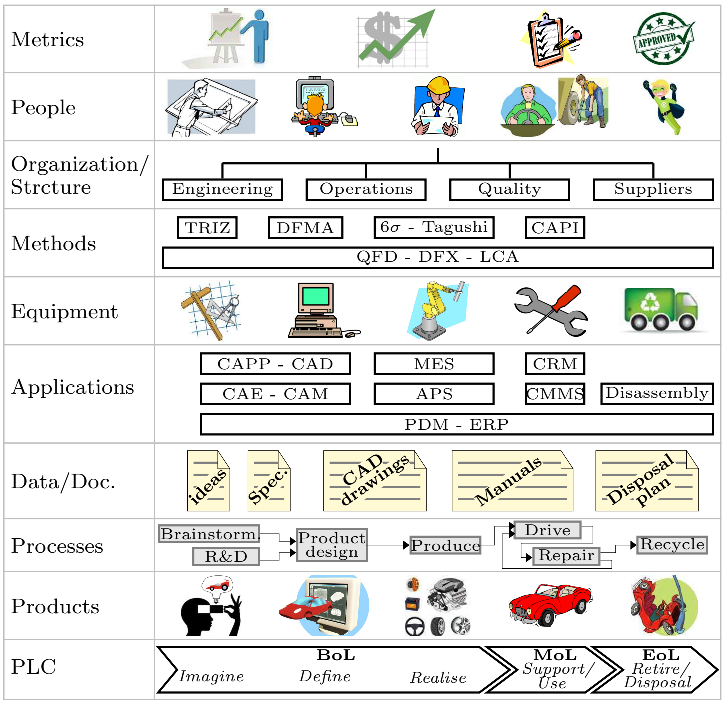


Figure 2.2: The PLM grid: adapted from [5]

Figure 2.2 provides the PLM grid with nine components (y-axis) according to each phase of the PLC (x-axis). The horizontal axis presents the phases of PLC with respect to the components on the vertical axis that have to be addressed when managing a product. This figure shows that PLM opens up a huge number of opportunities and benefits such as:

- *Valuable information*: Before PLM, people would carry out a customer survey (e.g. CAPI - Computer-Assisted Personal Interviewing as shown in Figure 2.2) to find out what customers thought of existing and future products. With the rapid expansion of Internet, wireless mobile telecommunications, RFID technologies, etc. they exchange information directly with customers who are using the product. Getting real-time data of use provides more valuable information than a survey form [22].

- *Increase in revenue*: PLM focuses on revenue increase by developing and supporting new products, improving product structure and improving the quality of existing services [5].

- *Cost saving*: PLM provides improved product design and reduced product development costs [1].

- *Increase in efficiency*: PLM increases efficiency in all the stages of PLC, including the supply chain.

- *New market opportunities*: PLM assists the enterprise to beat the market competition with innovative product content that carries first to market advantages and drives early product sales [1].

With the adoption of PLM, enterprises can gain many benefits including mass customization [23], high quality, reduced project failure rates, increased and quick innovation, quicker delivery, higher plant uptimes, effective management and use of corporate intellectual capital, effective communication among different groups at dispersed locations, minimized manufacturing costs, less industrial and commercial waste throughout every phase of the PLC, and being more environmentally aware [14].

Even though many benefits can be identified in traditional PLM, some issues remain to be addressed, such as:

- *BOL-centric*: The traditional PLM systems often manage the design and production phase in BOL but fail to manage MOL and EOL data.

Indeed, the information flow in the traditional PLM is hard to control with the classical auto-id technologies. There are CAD and CAM for BOL but no such systems for MOL and EOL.

- *Little feedback*: Even in the early twenty-first century, there is little automated feedback from products.
- *Loss of control*: When the product information is spread over many organizations and in many locations, the complexity of gathering data and consequently the danger of loss of control also increases [5].
- *Interoperability*: Problem in integration of different applications, different data structures and different services in PLC [24].
- *Knowledge management*: PLM does not focus on knowledge representation, capture, generation and dissemination [25].
- *Standards*: Applications in PLM environment create and store product information in different ways. This complicates the access and management of product data and its transfer between applications [5].
- *Data archiving*: Implementation of innovative ICT devices for PLM such as embedded smart wireless sensors, RFID, on-board computers and so on creates huge amounts of data.

The benefits and issues of traditional PLM are summarized in Table 2.1.

Table 2.1: PLM benefits and issues

Benefits	Issues
Valuable information	Information representation
Increase in revenue	Little feedback between the phases
Cost saving	Poor knowledge management
Increase in efficiency of PLC	Interoperability issue
Enhance strategic decision making	No definition of standards
New market opportunities	Data archiving issue

CL$_2$M helps to solve some of the above mentioned issues such as better traceability, interoperability, and information visibility throughout the PLC. The next section gives detail about the concept of closing the information loop.

2.2.2 Towards a CL$_2$M concept

PLM will be used much more extensively in the future and will support a growing number of enterprise activities. Over the last decade, some scholars like Kiritsis, Främling and Terzi [13, 3, 21] emphasized the importance of closing the information gaps, which are most commonly created between the BOL and the MOL-EOL phases. Kiritsis et al. [6] claim that closing the information loops should provide needed data to producers about the methods of use, retirement as well as disposal condition of the products. Moreover, it will help service, maintenance and recycling experts to have updated and real-time data about the product usage conditions. The feedback of information to manufacturers could be used to support the product use and disposal and possibly help to develop new generations of products. Nonetheless, the concept of closed loop for the product information flow management is not totally new. This concept has been used in many different contexts such as in Supply Chain Management, Asset Management and Cradle-to-Grave.

Closed Loop Supply Chain Management is the combination of traditional forward supply chain and the activities of the reverse supply chain [26]. It includes the product acquisition (to obtain products from end-users), reverse logistics (to move products from use phase to disposal phase), reuse, re-manufacture and re-marketing of the products. In closed loop supply chain management, tracking and tagging systems are used to track the movements of assets in a closed loop [27]. This concept focuses on managing product re-manufacturing, rather than on managing product information (i.e., feedbacks are made from EOL to BOL about re-manufacturing information).

Asset Management is another concept that supports the traditional PLM. Asset Management is the process of engaging manufacturers in the after sales services and, providing better development concept [28]. This concept allows companies to integrate the management of the product development activities with the management of the actual product instances during their use and service life. The concept focuses on the communication between engineering and service teams to enhance the capability to provide maintenance services to customers. Asset management focuses on the feedback from MOL to BOL as As-Maintained Bill of Materials.

The concept of Cradle-to-Grave tracks the life of a product from the point of creation (‘Cradle’) until the disposal (‘Grave’). The idea of this type of product analysis is to determine how products respond to various situations and applications throughout the PLC. The “Cradle-to-Grave” concept literally considers the birth of the material until its death by monitoring changes in the stress and strain states and the defect features [29]. This approach serves the purpose of monitoring the entire life-cycle of the consumption of goods and services from cradle to grave. The concept is also known as Lifecycle Assessment. The data obtained from an in-depth study like a Cradle-to-Grave assessment, Closed Loop Supply Chain and Asset Management often makes it possible to enhance the product over time and to decrease the PLC cost. However, the feedback flow of information is not guaranteed.

In the PROMISE EU project, it was found that many stakeholders in the product supply and the value chain (i.e., from designers to users and recyclers) desire to enable seamless information flow, tracing and updating of information about the product or process, even after its delivery to the customer and up to the final destination or decommissioning and then back to designer and producer as shown in Figure 2.3. For this challenge the CL_2M concept was developed and implemented in PROMISE [13, 8, 30]. CL_2M integrates sensor data and real time lifecycle event data into PLM that makes all the product related information visible through the PLC, and offers the opportunity of information feedback. CL_2M is based on the seamless flow of product information, via a local connection to the Product Embedded Information Devices (PEIDs) and, finally, through a remote Internet connection to knowledge repositories in Product Data Knowledge Management (PDKM) [31]. The main data in product information flow is presented in Appendix A.

A better understanding of PLC leads to process improvements and reduction of total product costs, better product quality, improved supply chain efficiency, and better rebuilding and recycling choices. CL_2M helps to understand the PLC in a better manner than the traditional PLM by closing the information loop. Closing the product lifecycle information loops (see Figure 2.3) has the following consequences:

1. Producers can be provided with complete data about the modes of use

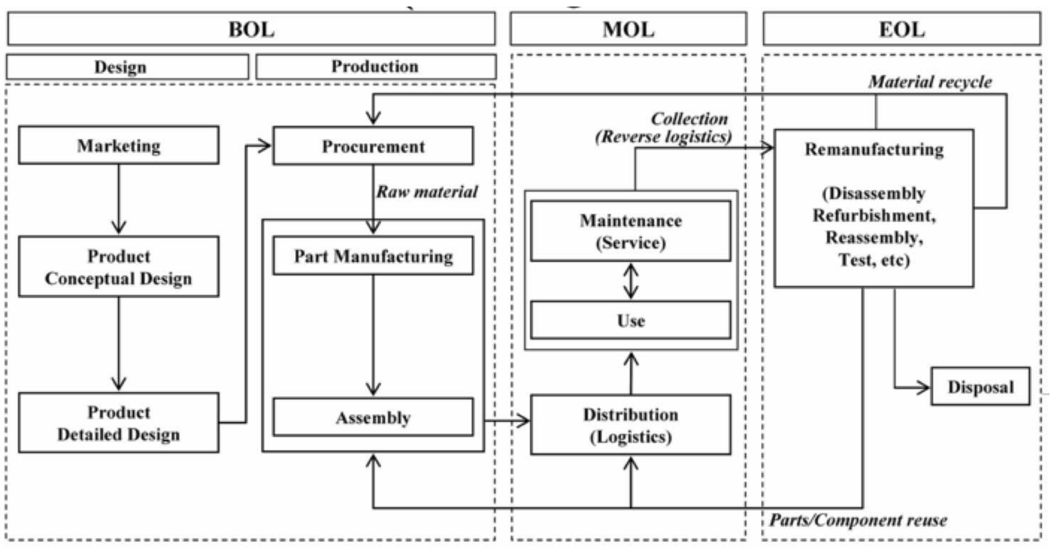


Figure 2.3: Closing information loop [22]

and conditions of retirement and disposal of their products. This can improve the quality of product design and the efficiency of production.

2. Service, maintenance and recycling experts can be assisted in their work by having a complete and always up-to-date report about the status of the product. Furthermore, predictive maintenance can also be performed by the maintenance engineers.

3. Designers are able to achieve expertise and know-how from the other actors in the product's lifecycle and can improve product designs in the direction of achieving PLC quality goals.

4. Materials recycling can be significantly improved. Recyclers and re-users are able to obtain accurate information about significant materials arriving via the EOL routes.

In short, CL_2M helps in optimizing the supply chain, calculation and optimization of the total lifecycle cost, the management of design evolutions, and the management of technological changes in the product. Figure 2.4 shows the detailed benefits of CL_2M to each phase of the PLC.

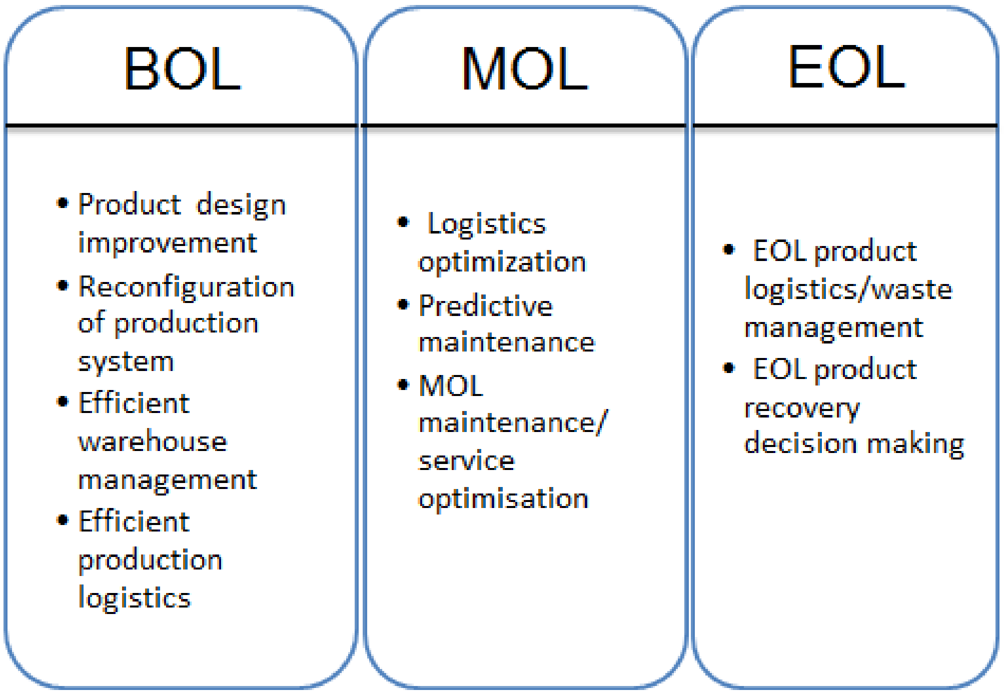

BOL
• Product design improvement
• Reconfiguration of production system
• Efficient warehouse management
• Efficient production logistics
MOL
• Logistics optimization
• Predictive maintenance
• MOL maintenance/ service optimisation
EOL
• EOL product logistics/waste management
• EOL product recovery decision making

Figure 2.4: CL$_2$M benefits in each PLC phase

2.3 PROMISE Project

The PROMISE project defined a concept where information generated during all phases of the PLC can be consolidated, closing the lifecycle information loops and which can then be transformed into knowledge [32]. The knowledge primarily aims at better PLC support for products and also creating a value in new products and services. Through this concept, the feedback of data, information, and knowledge from service, maintenance and recycling experts back to designers and producers becomes possible. To achieve such a feedback, all product related information generated during the PLC has to be managed in such a manner that it is easily accessible and usable in the other PLC phases.

PROMISE focused on developing appropriate technologies with product lifecycle models, the PEIDs associated with middleware and software components and tools for decision making based on data collected through a PLC. This enabled a seamless closed loop information flow, tracing and updating of information about a product even after delivery to the end user of the product, back to the designer and producer implementing CL_2M concept [30]. Hence, PROMISE defines CL_2M as an enterprise application that allows all the actors (managers, designers, service and maintenance operators and recyclers) to gather, manage and control product information remotely at any phase of the PLC.

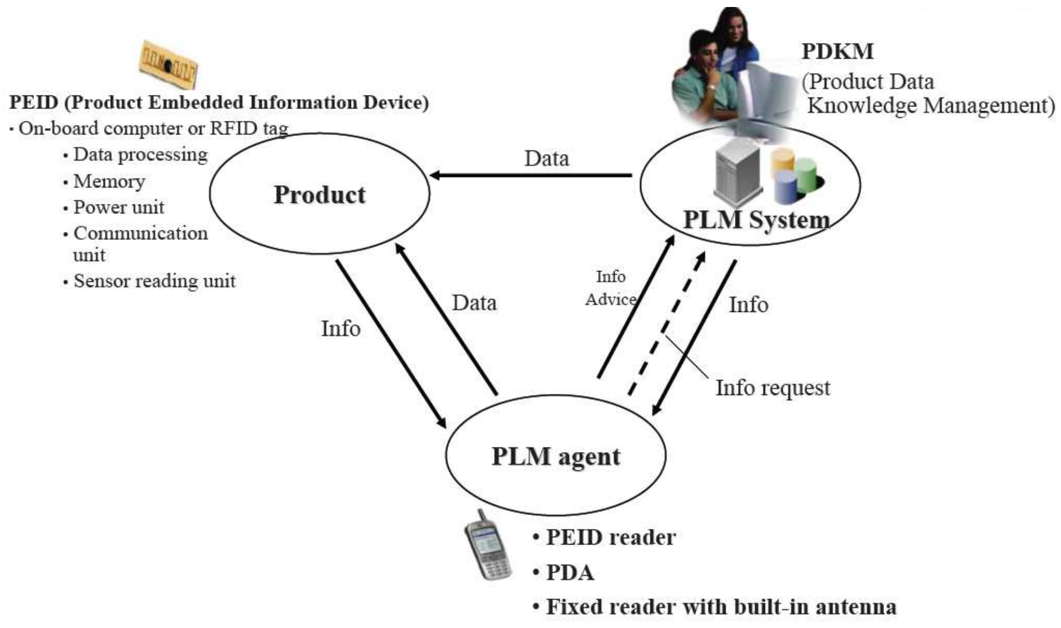


Figure 2.5: PROMISE concept [32]

The PROMISE system is based on the interaction among three components as shown in Figure 2.5 [32]:

- *Product*: PROMISE uses Product Embedded Information Devices (PEIDs) to send product information to a PLM agent.
- *PLM Agent*: The PLM agent can gather PLC information from each product at a fast speed with a mobile device like a personal digital assistant or a laptop computer with a PEID reader. The PLM agent sends information gathered on each site (e.g. retail sites, distribution sites and disposal plants) to a PLM system through the Internet.
- *PLM Server*: The PLM server provides PLC information and knowledge

created by PLM agents whenever requested by individuals or organizations.

The information flow between the three components is possible through the use of PROMISE Messaging Interface (PMI), which is described in the next section.

2.3.1 PROMISE Messaging Interface (PMI)

In the PROMISE world, information systems are grouped together under the concept of a “node” (see Figure 2.6). The messaging between the product, the PLM agent and the PLM system is done by passing messages between the nodes using a messaging interface called PMI.

The internal implementation of a node is not critical as long as it is capable of communicating using PMI. PMI is a key interface that enables a web-services based approach, permitting any PMI enabled user to exchange data with another. Depending on the complexity of any specific application, this can be achieved on a simple peer-to-peer basis if the two users are known to each other, or on a more complex wide area basis using PROMISE Data Services (middleware).

As a minimal requirement, a PEID must be able to communicate somehow with the outside world. It might be connected to PMI either through Internet connectivity or via a Device Controller (DC). A DC is required when the PEID embeds a technology that does not support TCP/IP protocol suite. For example, the RFID technology implements a simple point-to-point style communication protocol with the DC that provides the possibility to send and receive PMI messages [33, 34], as depicted in Figure 2.6. The data of interest created by PEID is sent to the back end server, which stores the data that should be further analysed by a Decision Support System (DSS). Therefore, a PMI node is able to communicate with any other PMI node as long as both implement the PMI message format.

PMI is basically a mode of exchanging structured XML messages between nodes, where the structure is defined by an XML schema. The actual XML can be transported over any low-level protocol and web service such as HTTP,

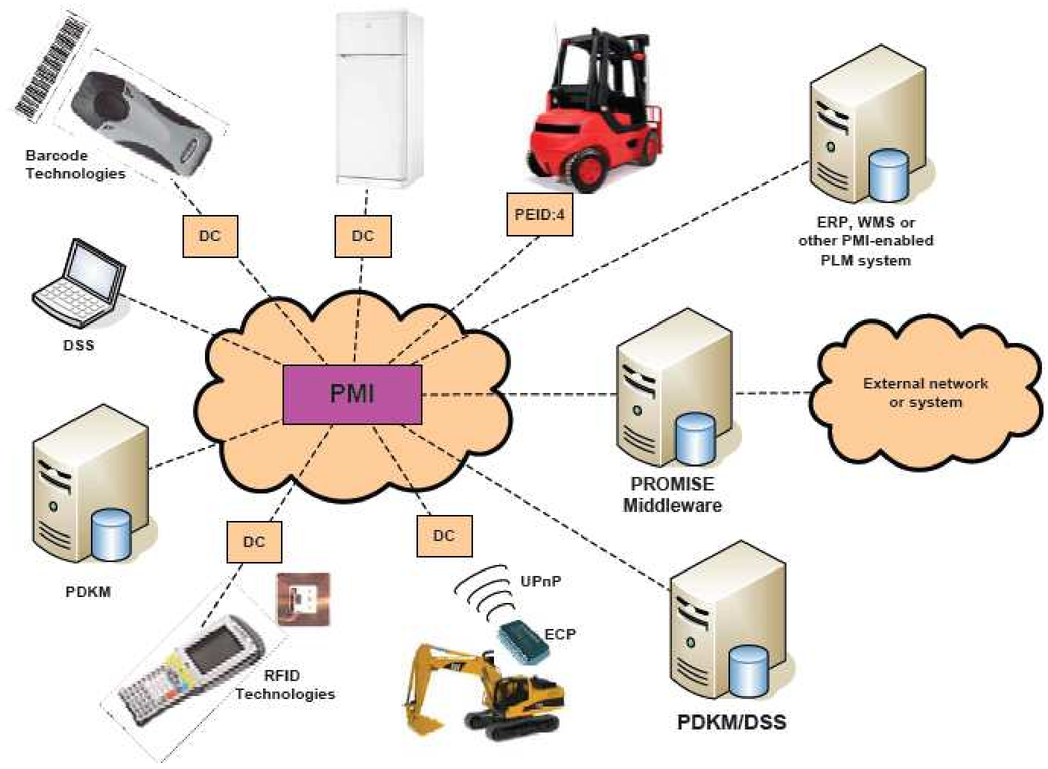


Figure 2.6: Messaging interface conceptual connectivity [8]

SMTP or SOAP. In PROMISE, the Internet is the main medium for communication between the different information sources (i.e. between PDKM, DSS, etc.).

Currently, the research team is further developing PMI with The Open Group [1] under the name *Quantum Lifecycle Management* (QLM) messaging. QLM messaging is introduced in Chapter 3.

2.3.2 PROMISE implementation example

Ten demonstrators have been implemented in the PROMISE project that are listed in Table 2.2. Details about these demonstrators are publicly available at the PROMISE website [8].

In what follows, the outlines of demonstrator 7 are presented. A refrigerator system aims at delivering the knowledge of its behaviour for predictive

[1]QLM Open Group: https://collaboration.opengroup.org/qlm/

Table 2.2: PROMISE demonstrators [8]

No.	Demonstrators	Lifecycle phase focus
1	Monitoring End of Life Vehicles	EOL
2	Heavy load vehicle decommissioning	EOL
3	Tracking and tracing of products for recycling	EOL
4	Predictive maintenance for trucks	MOL
5	Heavy vehicle lifespan estimation	MOL
6	Predictive maintenance for machine tools	MOL
7	Predictive maintenance for Refrigerator	MOL
8	Predictive maintenance for Telecom	MOL
9	Design for X	BOL
10	Adaptive Production	BOL

maintenance purposes, thereby improving the quality of both the product and its service. As shown in Figure 2.7, the actors in the system are:

- A refrigerator DA (Digital Appliance) with the refrigerator main board as it PEID.
- An interface device SA (Smart Adapter) placed between the power cable of the refrigerator and its electric plug (Outlet).
- A wireless communication link between SA and middleware using Bluetooth.
- PDKM that stores and manages product information and a remote monitoring center where a DSS (Decision Support System) performs predictive maintenance.

The SA communicates either with a local monitoring system, e.g. PC/PDA (Personal digital assistant) via Device Controller (DC) middleware or with remote diagnostic system via Internet. Field data (statistical and diagnostic data), coming from sensors, are logged by the refrigerator control system in its non-volatile memory. These data related to appliance (e.g. energy consumption) are then sent to the SA and stored in the PDKM database. In DSS the data is analysed to find out whether eventual malfunctioning problems are going to occur on some refrigerator components. If an emerging failure is detected, an email is sent to the maintenance service company that can perform predictive maintenance actions on the refrigerator [32]. The knowledge could also help to determine in advance which spare parts are needed for the job and how to improve the scheduling of the service personnel. This gives an insight into how a refrigerator product selling activity could gradually change into a service selling activity (i.e. selling refrigeration services rather than selling the physical refrigerator itself).

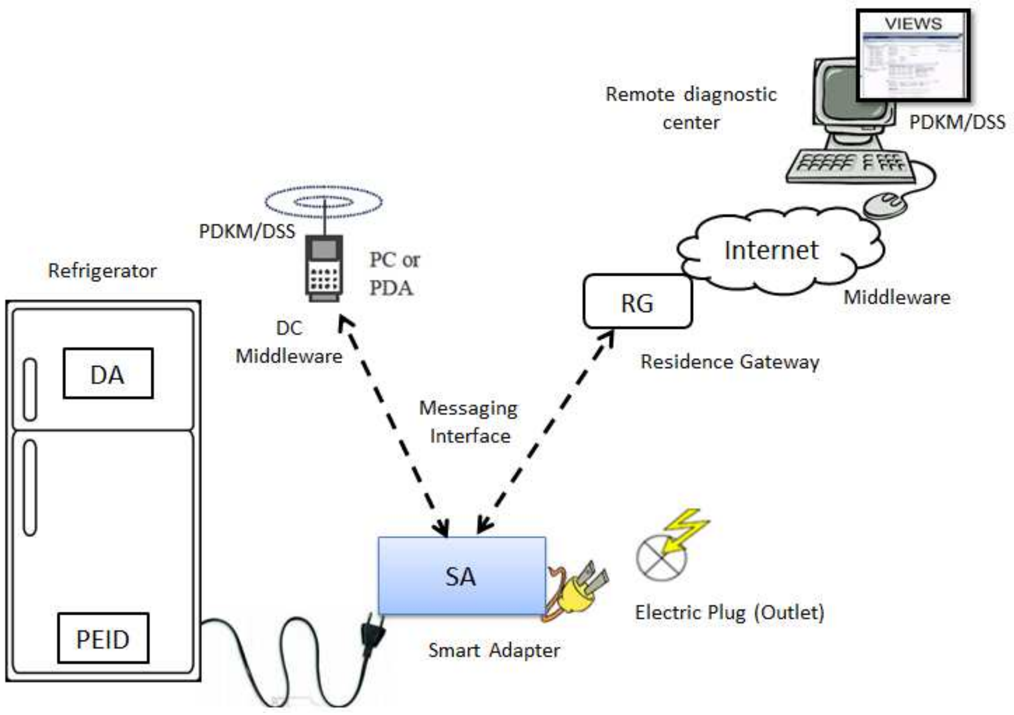


Figure 2.7: Illustration of connections and information flows

The architecture used in CL$_2$M is detailed in the next section.

2.4 CL$_2$M architecture

The CL$_2$M architecture is described based on the user requirement analysis from PROMISE project [9]. The objective of CL$_2$M is to streamline the product lifecycle operations over the whole PLC based on seamless information flows. This is achieved through a local network of PEIDs and/or through a remote Internet connection to access knowledge repositories in PDKM.

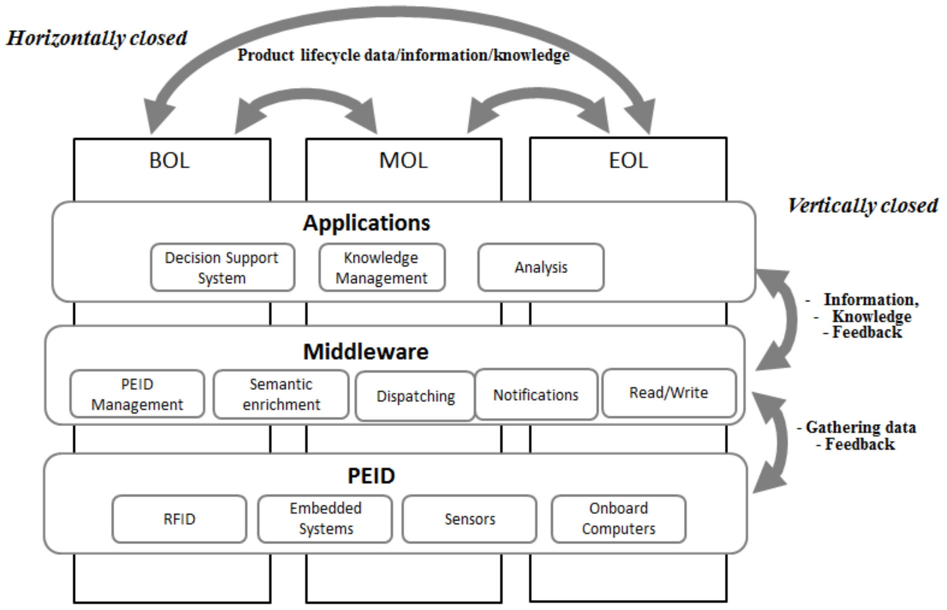


Figure 2.8: Overview of CL_2M architecture: adopted from [30]

In CL_2M, the information is gathered and controlled in the horizontal and vertical loops of PEID (hardware), middleware (software), and applications (business) processes as shown in Figure 2.8. The PLC information can be used to streamline the operations at the MOL and EOL. The PLC information also goes back to the designer and producer (BOL) so that the information flow can be horizontally closed. Information control flow is vertically closed as well. Based on the data gathered by PEID and sensors, the product related information can be analysed and decisions based on the behaviour of the products can be undertaken. The PEIDs gather product related data under specific conditions and requirements or periodically in a real-time way and then, they send the data to a PDKM. The data is exchanged or communicated between many systems through the messaging interface and the middleware. Based on received data, valuable information and knowledge are generated and stored in a PDKM. The information is used for decision making by the PLC actors. After analysis and decision making, if there is any need to update product information, the server sends an updated information to PEID directly from PEID reader or via the Internet from messaging interface.

The CL_2M architecture is further divided into business and technical architectures, which are respectively presented in Sections 2.4.1 and 2.4.2.

2.4.1 CL_2M business architecture

To coordinate and manage the business activities in CL_2M, there is a need for a business architecture [22]. The business architecture is divided into the three main phases of the PLC (BOL model, MOL model and EOL model) with a closed loop.

In BOL, designers and production engineers receive feedback from distributors, maintenance and service engineers, customers and remanufacturers about product status, usage, conditions and disposal information.

In MOL, the PEIDs gather the log data about the product history that is analysed in the lifecycle management system. Through the system such as the Internet or the wireless mobile technology, the updated information about the product status and a real time assistance is provided in MOL.

In EOL, information feedbacks improve the product design in BOL and helps to optimize maintenance and service in MOL.

Figure 2.9 shows the business architecture that describes the relevant information flows and the feedbacks from one phase to another. CL_2M not only makes the product information highly visible in the whole PLC, but also helps in decisions to reduce the inefficiencies in the lifecycle operations, optimize the lifecycle cost, manage production performance quality and gain competitiveness.

The technical architecture addresses the infrastructure that supports the business architecture. Section 2.4.2 describes which hardware and software components are required to construct a CL_2M system.

2.4.2 CL_2M technical architecture

In PLM, the various software tools, systems, and databases are often distributed over various departments and suppliers throughout the PLC. All these must be integrated so that the information can be shared promptly and correctly between people and applications [35]. This is the main objective of

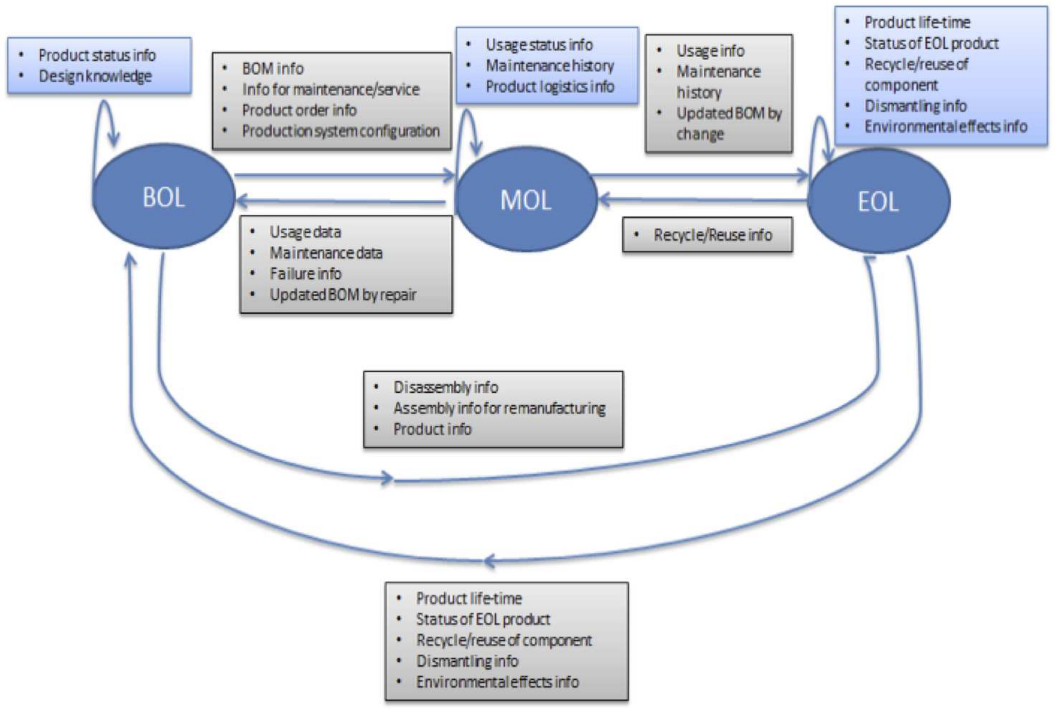


Figure 2.9: CL_2M business architecture: adopted from [22]

CL_2M. The technical architecture used in CL_2M from gathering raw data to business applications is depicted in Figure 2.10.

- PEIDs are attached to the product to gather the information from the product. An embedded software is built into the PEID hardware for controlling and managing the PEID data.

- Middleware can be considered to be an intermediate software between different applications. It connects the different software layers, e.g. between the PEIDs and the database. It is used to support complex and distributed applications and enables communication, coordination and management of data.

- The DSS is an important component in the technical architecture of CL_2M since it has the ability to transform the gathered data into necessary information and knowledge for specific applications.

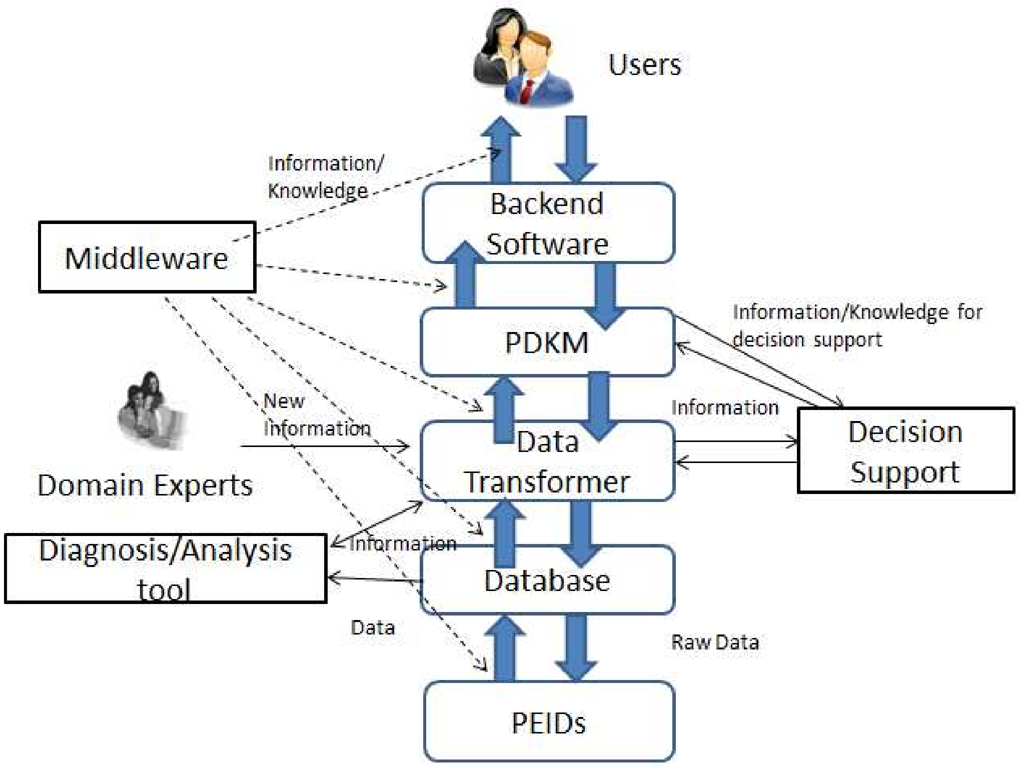


Figure 2.10: CL_2M technical architecture: adopted from [22]

- The PDKM manages the information and knowledge generated during the PLC and is generally linked with both the DSS and the data transformer. PDKM is both the process and the associated technology to acquire, store, share and secure product information.
- Users get access to the information/knowledge from the backend software they have in their premises.

Therefore, for contrast to the concept of the traditional PLM, the information flow and the control flow are horizontally and vertically closed in CL_2M. Product lifecycle data (such as usage conditions, failure, and maintenance or service events, etc.,) is gathered by the PEID embedded in each product over the whole PLC. In the technical architecture, PEID can be seen as *intelligent product* and have different level of intelligence. Intelligent products adds the capabilities of collecting product information through the PLC and reacting on it pro-actively [11]. Several definitions and classifications of an intelligent product are admitted [36, 11, 37], which is introduced in the next Chapter 3. Another main objectives of this thesis is to develop an information exchange standard to enable the exchange of any kind of information between any kind of “intelligent product”.

Chapter 3

Quantum Lifecycle Management messaging standard

Synopsis

The development of Quantum Lifecycle Management (QLM) messaging is introduced in this chapter along with its design principle. An implementation example of QLM messaging for monitoring power consumption is also presented. Then, QLM is compared with other similar existing messaging standards to highlight its potential benefits or drawbacks.

3.1 Intelligent products and Internet of Things

Intelligent products and systems generate the information that can be transformed into knowledge to better support the existing products and to increase their service value [36]. In order to understand the development of the QLM messaging standard, the concepts of intelligent product and Internet of Things (IoT) are first defined.

IoT has been described as an extension to the Internet, so that it would be possible to collect and read information about physical things over the Internet [38]. This concept is focused on product identification technologies, tracking of the product locations and the global visibility of objects [39]. The upcoming IoT has become the focus of intense research in many countries.

"IoT is an integrated part of the Future Internet and could be de-

fined as a dynamic global network infrastructure with self configuring capabilities based on standard and interoperable communication protocols where physical and virtual things have identities, physical attributes, virtual personalities and use intelligent interfaces, and are seamlessly integrated into the information network [40]".

Intelligent products are reactive actors that are capable of autonomously adapting to changes in their environment. McFarlane et al. [41] defines an intelligent product to be a physical and information-based representation of a product that has the following properties:

- Possesses a unique identification.
- Is capable of communicating effectively with its environment.
- Can retain or store data about itself.
- Deploys a language to display its features, production requirements, etc.
- Is capable of participating in or making decisions relevant to its own destiny.

Meyer et al. [11] recently provided a complete survey on intelligent products and related applications (SCM, IoT). It shows that the applications tends to be focused on product identification technologies, information storage and information exchange rather than on the intelligence of the products. Intelligent products have the means to communicate among themselves and with other information systems. They can also play an essential role in $\mathrm{CL_2M}$ by their capability of collecting information. In order to let intelligent products communicate with each other and with other information systems, an information system architecture with standardized communication interfaces needs to be created for the purpose of product tracking, product data gathering and information sharing. This was done in the PROMISE project through the creation of PMI. PMI was implemented by four partners of the PROMISE consortium. One of the implementations is based on the DIALOG (Distributed Information Architecture for Collaborative Logistics) platform, which is mainly used for testing and verifying new concepts and models for research purposes [42].

DIALOG is programmed in Java and initially exchanged Java objects of the class DialogMessage or classes derived from it. Support for PMI was added by a interface “ wrapper” that mapped PMI into DialogMessage objects that could be consumed by the DIALOG agents, and vice versa.

At present, the research community involved in DIALOG development is developing a messaging standard derived from the PMI called “QLM Messaging Interface”. Simultaneously, the DIALOG platform is further developed so that the communication with local and remote agents can be performed in identical ways. Software components that implement the QLM Messaging Interface can also communicate directly if they do not need services provided by DIALOG (i.e., message routing, message persistence, etc.).

In our definition of IoT, it is possible to easily query and set up information flows between any kinds of intelligent products, devices, computers, users and information systems in general. However, at present it is not possible to do so due to the lack of sufficiently generic application-level interfaces for exchanging the kind of information required by an IoT. The QLM Messaging Interface is being developed and is proposed as a standard application-level interface that would fulfill those requirements. This thesis presents the design principles and the specifications of the QLM messaging standard.

3.2 Development of Quantum Lifecycle Management messaging

PROMISE created two main specifications that fulfilled the necessary requirements for the project: the PROMISE Messaging Interface (PMI) and the PROMISE System Object Model (SOM). The Quantum Lifecycle Management group derived two standards from PMI: the QLM Messaging Interface and the QLM Messaging Format.

In the QLM system architecture (Figure 3.1), the communication between the participants, e.g., products and backend systems, is done by passing messages between the nodes using the QLM Messaging Interface. The QLM cloud is similar to the Internet cloud, as the QLM Messaging Interface is intended to play the same role in the IoT as HTTP does for the Internet (Figure 3.2).

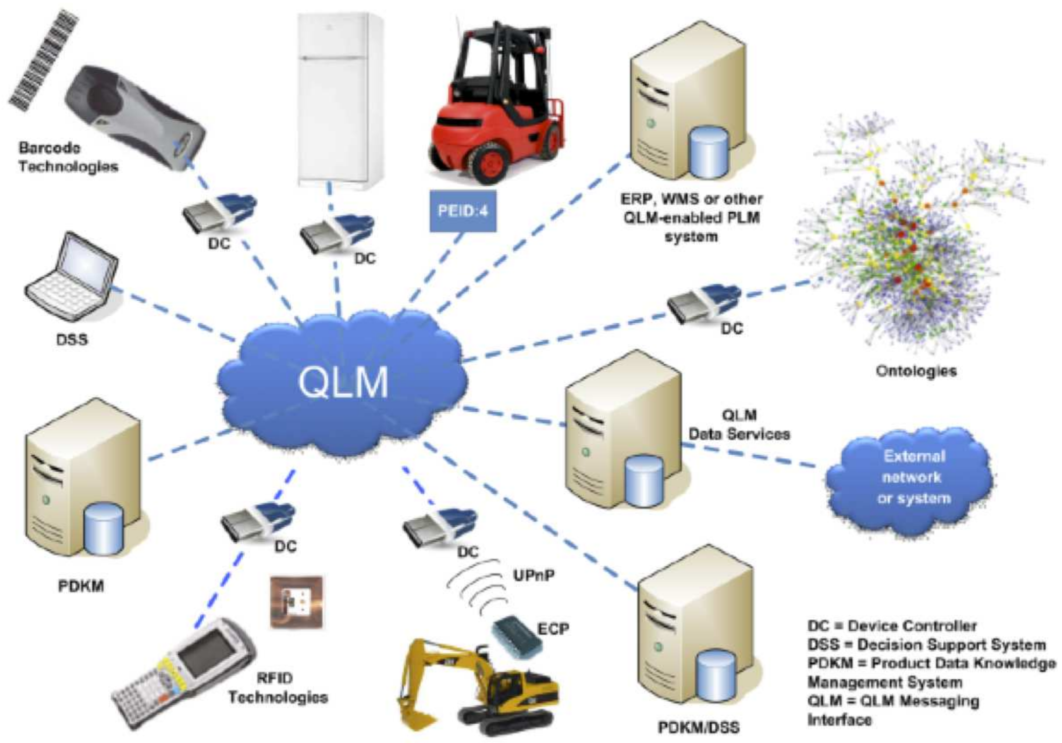

Barcode Technologies
DC
DC
PEID:4
ERP, WMS or other QLM-enabled PLM system
DSS
DC
Ontologies
QLM
QLM Data Services
External network or system
PDKM
DC
DC
UPnP
ECP
RFID Technologies
PDKM/DSS
DC = Device Controller
DSS = Decision Support System
PDKM = Product Data Knowledge Management System
QLM = QLM Messaging Interface

Figure 3.1: QLM conceptual connectivity [43]

In order to reply to the requirements for information exchange between the intelligent products in the IoT, an accompanying standard called "QLM Messaging Format" is specified. The QLM Messaging Format partly fulfils the same role in the IoT as HTML does for the Internet (see Figure 3.2).

3.2.1 QLM Messaging Interface

A QLM Messaging Interface node in the architecture is a communication end-point in a QLM network, and it manages communications for one or several devices [43]. A fundamental characteristic of the QLM Messaging Interface is that nodes do not have predefined roles, as it follows the peer-to-peer approach to communications. Hence, entities implementing the QLM interface can communicate directly with each other or with back-end servers. The QLM Messaging Interface can also be used for the server-to-server information exchange of sensor data, events, and other information. A full QLM node is capable of sending as well as receiving requests and includes both the client and

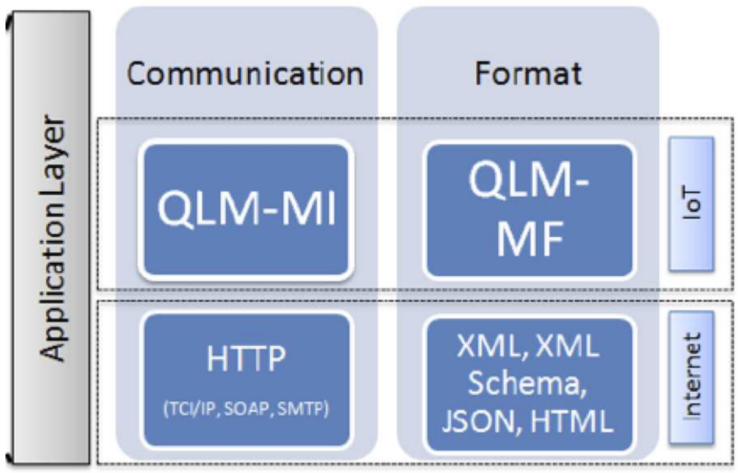


Figure 3.2: QLM messaging in the application layer of the OSI model

the server functionalities. However, a more limited node for sending messages may just implement client functionality. An example of such limited nodes are those associated with the RFID tag readers, or more generally, nodes that are unreachable from the outside because of a firewall. Such nodes periodically send product data to a product monitoring system according to a *subscription* specified when the product is installed. Subscription is the possibility of retrieving specific information automatically and periodically from a specific device during an interval time.

The QLM Messaging Interface defines different operations such as a read or a write of the value of a particular InfoItem. The InfoItems represent the actual values, such as sensor readings of a device, e.g., a refrigerator, a car, a manufacturing machine etc. The parameters for the method calls are XML strings whose structure is defined by an XML schema. In addition to reads and writes, the QLM Messaging Interface also provides callback methods for asynchronous communications. Examples of asynchronous communications include a subscription read, a call to the read method with parameters that specify that the target node should not respond directly with a value, but rather send multiple responses at a specified interval. The callback method interface also provides a mechanism for nodes to send events to each other with or without a prior subscription, subject to the particular node implementation.

Indeed, the callback method is an embodiment of the well-known *Observer* pattern applied to messaging systems [44, 3].

Developing a function that requires communication between two distributed components can be unexpectedly difficult when you consider all the possible failure scenarios. Let us consider the case of a program sending a purchase request to another program over a network. If the message is not delivered and the sender is not made aware of the delivery failure, then the purchase request will be lost. If the message is delivered more than once and the target is unaware of these multiple deliveries, then too many purchases will occur. Reliable messaging refers to the ability of a sender to deliver a message once and only once to its intended receiver [45]. The basic method for achieving reliable delivery is to send the message repeatedly to the target until the target acknowledges receipt of the message. The message must contain an identifier so that the target will discard any duplicates it receives. Also, if the sender's server goes down, the sender may lose its copy of the message and, therefore, will not be able to resend it. For this reason, senders need to record the message in a reliable store until it is definitely delivered. Furthermore, the sender frequently needs to make a record of the fact that the request has been sent.

Some systems are capable of broadcasting a message to many destinations. Others only support sending a message to a single destination. Some systems provide facilities for asynchronous receipt of messages (messages are delivered to a client as they arrive). Others support only synchronous receipt (a client must request each message) [46]. Important attributes for messages are time-to-live, priority and whether a response is required. QLM messaging is specified keeping these features of messaging services in mind.

The main properties and requirements for the QLM Messaging Interface are listed in the specification document made by The Open Group of QLM, namely [47]:

1. QLM Messaging Interface messages can be transported using most "lower-level" protocols, such as HTTP, SOAP, SMTP. It can also be transported using file storage media such as USB sticks.

2. Three possible operations: read, write, and cancel. Read is for immediate

retrieval of information from a QLM node and for placing subscriptions. Write is for sending information updates to QLM nodes. Cancel is for cancelling subscriptions before they expire.

3. QLM nodes can request for current and historical data with immediate response by the read operation.

4. QLM nodes can send data to each other at any time by the write operation.

5. Subscriptions can be made for asynchronous retrieved of data from other QLM nodes. This is done by the read operation if the interval parameter has been set. If a callback address is provided, then the data is sent using a QLM Messaging Interface response message at the requested interval. If no callback address is provided, then the data can be retrieved by issuing a new read request with the ID of the subscription (this is particularly useful if the requesting node is behind a firewall).

6. All requests and responses can specify a time-to-live. If the message has not been delivered to the "next" node before time-to-live expires, then the message is removed and an error message is returned or sent to the message originator, if possible.

7. Enable synchronous ("real-time") communication between nodes. Any response message can include a new request. It also provides a possibility to perform "client-initiated" communication with nodes that are located behind firewalls.

8. Publication and discovery of data sources, services and meta-data. Publication of new data sources, services and meta-data can be done with write operation. "RESTful" URL-based (HTTP-GET) queries (in addition to read operations) allow the discovery of them, including discovery by search engines. Remark: data source, service, meta-data etc. semantics are not specified by the QLM Messaging Interface, that is normally done by domain- or application-specific standards or APIs.

9. Allowing different payload formats both for requests and responses. A QLM Messaging Interface message can transport actual information using any format that can be embedded into an XML message.

10. All requests can specify a list of target QLM nodes. The receiving node(s) are then responsible of re-routing the request to the target QLM nodes, or sending back an error message to the requesting QLM node in case of failure.

3.2.2 QLM Messaging Format

A QLM Messaging Format structure is a hierarchy with an Objects element as its top element. The Objects element can contain any number of Object sub-elements. Object elements can have any number of properties, called InfoItem, as well as Object sub-elements (see Figure 3.7). The resulting Object tree can contain any number of levels. InfoItems can contain three optional sub-elements:

- *Display*: Text intended mainly for human user interfaces that explains what the InfoItem is,
- *MetaData*: Sub-element that provides meta-data information about the InfoItem, such as value type, units and similar information,
- *Value*: Arbitrary number of values for the InfoItem, possibly with timestamps.

3.2.3 Communication protocol for QLM Messaging Interface

The QLM Messaging Interface messages can be exchanged with many protocols such as, HTTP, SOAP, SMTP, FTP, etc. The most appropriate protocol to use depends on the application. Different protocols provide their own security mechanisms that might be important when choosing the one to use. The chosen communication protocol for this research is HTTP with the POST functionality.

HTTP Interface

QLM Messaging Interface messages can be communicated using plain HTTP communication. If so, requests and callback messages should be sent using HTTP POST messages. Figure 3.3 shows how a minimal (payload) QLM Messaging Interface message can be sent with HTTP POST using the Unix *curl* utility, where the URL of the QLM node is http://dialog.hut.fi/qlm/. If it is received correctly, then a reply similar to that in Figure 3.4 should be received [1].

```
curl http://dialog.hut.fi/qlm/
--data msg="<?xml version="1.0" encoding="UTF-8"?>
<qlm:qlmEnvelope version="1.0"
ttl="10"><qlm:read><qlm:msg></qlm:msg></qlm:read>
</qlm:qlmEnvelope>"
```

Figure 3.3: Example of QLM Messaging Interface message sent using the Unix "curl" utility

```
<?xml version="1.0" encoding="UTF-8"?>
<qlmEnvelope version="1.0" ttl="0">
        <response>
                <result>
                        <return returnCode="200"></return>
                </result>
        </response>
</qlmEnvelope>
```

Figure 3.4: Typical minimal response to a QLM request

RESTful Interface

The QLM Messaging Interface has been designed to be RESTful whenever possible. For the publication and discovery of data sources QLM nodes implement a URL-based (HTTP GET) mechanism for retrieving a list of available

[1]ttl: The time to live (in seconds) for the QLM Messaging Interface requests. The time is counted from the time the QLM Messaging Interface request was received by the QLM node. The value "0" signifies that a response must be provided while the connection is active (for instance, in the case of HTTP). The value "-1" signifies "forever" [47].

information in a hierarchical way. The actual semantics being used in the URLs depends on the domain and will normally be defined by the same (or similar) XML schema as is being used for the message payloads. The QLM Messaging Format provides a generic payload format.

```
wget http://dialog.hut.fi/qlm/Objects/
```

Figure 3.5: Issuing HTTP GET request to QLM node at "http://dialog.hut.fi/qlm/" for getting the available information about the data source "Objects"

```
<Objects>
        <Object>
                <id>Refrigerator123</id>
        </Object>
        <Object>
                <id>HeatingController321</id>
        </Object>
        <Object>
                <id>WeatherStation651</id>
        </Object>
</Objects>
```

Figure 3.6: QLM Messaging Format to request for Objects list

HTTP GET requests can be performed directly from a browser's address line. Another possibility is to use for instance the Unix "wget" utility as shown in Figure 3.5. This request would return just an XML structure with "Objects" as the root element and all the first-level properties of the Objects element as shown in Figure 3.6. Then the elements of the retrieved XML structure can be used for drilling further down into the QLM Messaging Format object hierarchy, where for instance, the URL (http://dialog.hut.fi/qlm/Objects/Refrigera -tor123/) would return the list of properties (called InfoItems) and the possible sub-objects of the object "Refrigerator123".

Contrary to the many RESTful specifications, the QLM Messaging Interface does not use PUT and DELETE methods because that would create an explicit link between the QLM Messaging Interface and the HTTP protocol, which would go against the functional requirements set out for the messaging interface.

```
<?xml version="1.0" encoding="UTF-8"?>
<qlm:qlmEnvelopettl="-1" version="0.2">
       <qlm:writemsgformat="QLM_mf">
              <qlm:msg>
              <Objects>
                     <Object>
                            <id>Refrigerator123</id>
                            <InfoItem>
                            <id>FridgeTemperatureSetpoint</id>
                                   <value>3.5</value>
                            </InfoItem>
                     </Object>
              </Objects>
              </qlm:msg>
       </qlm:write>
</qlm:qlmEnvelope>
```

Figure 3.7: Example of QLM write message using QLM Messaging Format semantics

3.3 Wattson monitor implementation example

In this section an implementation example of the QLM messaging standard is presented. This example was implemented at the BIT research center [2] of Aalto University in 2012 to demonstrate the use of QLM Messaging Interface and QLM Messaging Format for a power consumption monitoring application.

This example is about monitoring the real time power consumption of the power socket and calculating the cost of the power consumption. The implementation works with:

- Wattson power meter device: It is a wireless portable energy monitor

[2]http://www.bit.tkk.fi/

showing the real-time electricity consumption. This device is attached with a sensor clip around the electrical wire that is measured.

- WiFi router: It is a wireless router, which was used as a QLM node.
- DIALOG: *It is a generic software in the sense that it provides protocol and interface neutral message passing mechanisms with message persistence functionality, security mechanisms etc. that are abstracted away from the business logic itself, implemented by agents* [44, 33]. The open-source implementation is based on the DIALOG platform.

The information flow from Wattson power meter to the user's system is shown in Figure 3.8.

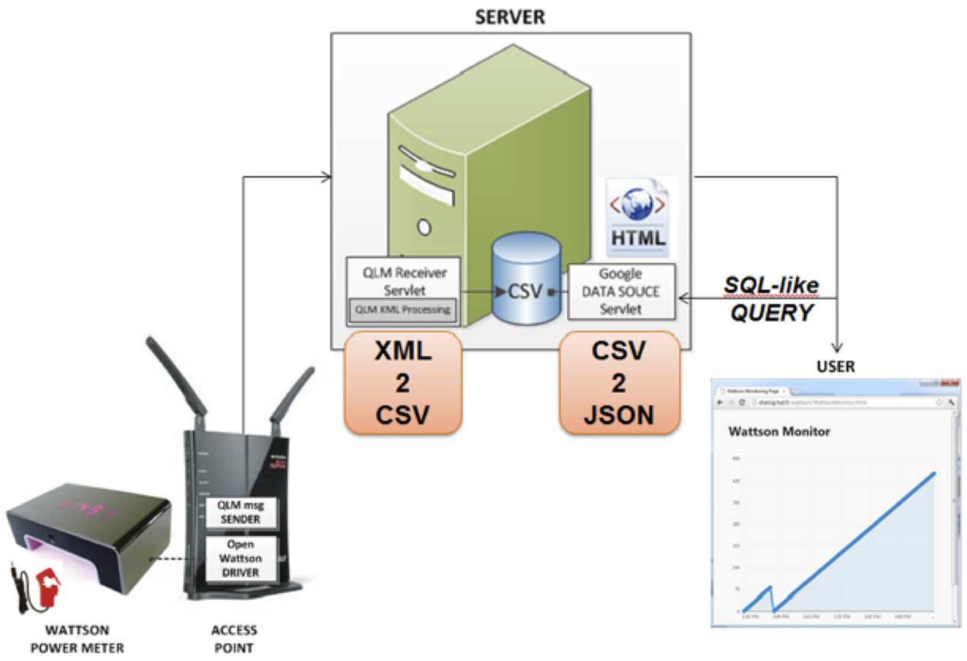


Figure 3.8: QLM messaging implementation example, showing the information flow from intelligent product to web browser in internet

In this implementation, DIALOG was used as middleware for creating a connection between one QLM node (access point) and another QLM node (a user's laptop to display the power consumption), which could also further transmit information to other systems using the QLM Messaging Interface.

A QLM Messaging Interface message is sent using HTTP POST. The curl command is used in the QLM sender script in WiFi router. The QLM XML message with a QLM Messaging Format payload is sent from the WiFi router to the QLM receiver servlet shown in Figure 3.9. This XML message is converted into CSV (Comma Separated Value) format and then sent to the Google data source servlet. Google chart API (Application Programming Interface) is used for the visualization of the power consumption in the chart form (http://dialog.hut.fi/wattson/WattsonMonitor.html).

```
<?xml version="1.0" encoding="UTF-8"?>
<qlm:qlmEnvelopettl="-1" version="0.2">
       <qlm:writemsgformat="QLM_mf">
              <qlm:msg>
              <Objects>
                     <Object>
                            <id>WattsonMonitor22334411</id>
                            <InfoItem>
                                   <id>PowerConsumption</id>
                                   <value>43</value>
                            </InfoItem>
                     </Object>
              </Objects>
              </qlm:msg>
       </qlm:write>
</qlm:qlmEnvelope>
```

Figure 3.9: QLM XML message for the power consumption

The Wattson implementation example shows how QLM Messaging Interface and QLM Messaging Format are used to transport information from an intelligent product to the user's system. The information exchange from both sides was not implemented in this example. Only sending information with the write operation was implemented. The setup of the entire implementation is shown in Figure 3.10.

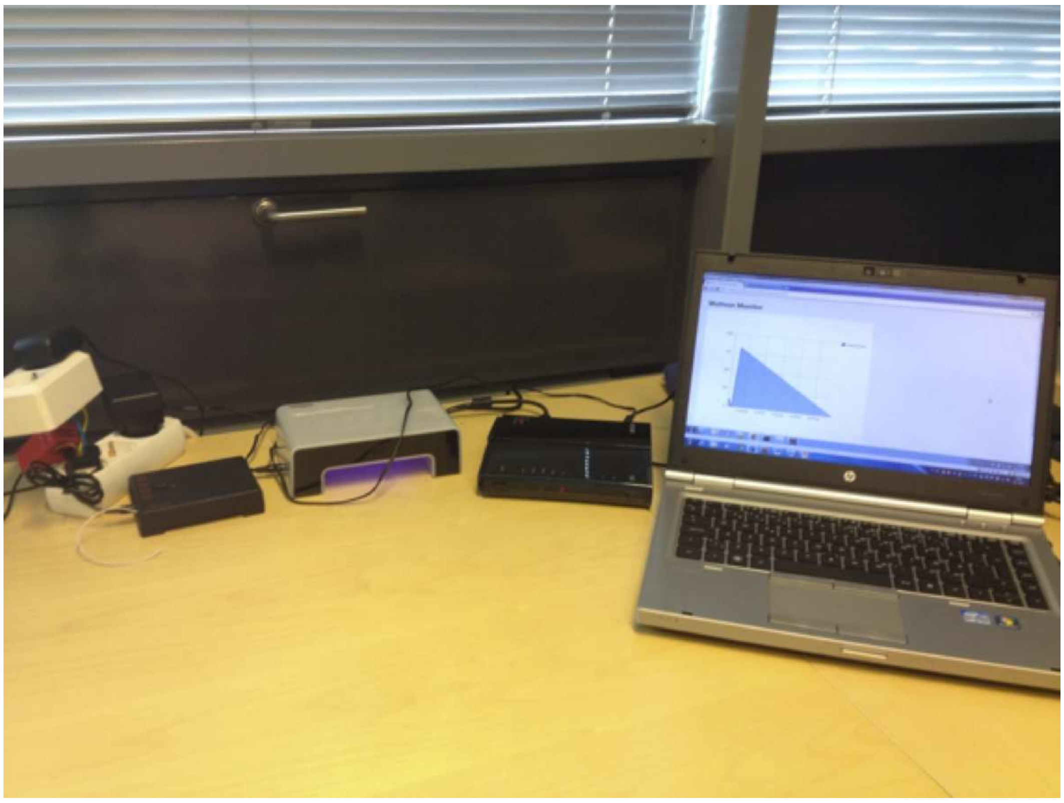


Figure 3.10: Photo of the implementation setup

3.4 Comparison with existing messaging standards

Other standards exist in the information services area that are potentially useful to provide inter-organizational information exchange functionality, such as EPC (Electronic Product Codes) Information services (EPCIS) developed by EPCglobal network Auto-ID consortium, Simple Object Access Protocol (SOAP/WSDL) from the World Wide Web Consortium, Open Building Information eXchange (oBIX) from Technical Committee of OASIS and Java Messaging Service (JMS) from Sun Microsystem Inc.

EPCIS is the format for exchanging information related to the static properties of a group of products, (e.g., name of the product), the data about a product item, (e.g., date and place of manufacture). It also defines how to exchange supply chain information such as locating where serialized products are stored and who has custody of the product [48]. As such, EPCIS uses a server based philosophy, it does not currently support the transmission of sensor values, and is very focused on logistics information.

SOAP is the protocol used in encoding the desired action and parameters when a client calls a web service. The SOAP interface is described using the Web Service Definition Language (WSDL) that describes what kind of XML encoded SOAP messages can be sent over the HTTP POST method (W3C SOAP). However, SOAP could also be used as transport protocol for QLM.

This thesis is mainly focused on comparing the QLM messaging standard with oBIX and JMS that are quite similar to QLM. Still there are some glaring differences, which are listed in the following sub sections.

3.4.1 oBIX

The oBIX web site[3] states that the purpose of the OASIS oBIX Technical Committee (TC) is to define a standard web services protocol to enable communications between building mechanical and electrical systems, and enterprise applications.

The oBIX architecture is based on the following principles:

- *Object Model*: a concise object model used to define all oBIX information.
- *XML Encoding*: a simple XML syntax for expressing the object model.
- *Binary Encoding*: a simple binary encoding for constrained devices and networks such as 6LoWPAN sensor networks.
- *URIs*: URIs is used to identify information within the object model.
- *REST*: a small set of verbs is used to access objects via their URIs and transfer their state via XML.
- *Contracts*: a template model for expressing new oBIX “types”.
- *Extendibility*: providing for consistent extendibility using only these concepts.

[3]www.obix.org

The current specification defines how these request/responses can be implemented between a client and server using HTTP and SOAP. Objects are the fundamental abstraction used by the oBIX data model. Object properties are defined using the XML attributes with a limited number of simple data types.

oBIX can be considered to use a weakly-typed model, where objects with any structure and contents can be freely exchanged. This standard is a suitable communication interface for data acquisition and control of devices of different kinds. It also provides an easy and light-weight discovery mechanism, e.g., for dynamically creating graphical user interfaces that automatically include all available devices and other information sources.

The oBIX XML for a smart house example is shown in Figure 3.11, retrieved from the URI "http://server/obix/myhome/thermostat".

```
<obj href="http://myhome/thermostat">
<real name="spaceTemp" unit="obix:units/
fahrenheit" val="67.2"/>
<real name="setpoint" unit="obix:units/
fahrenheit" val="72.0"/>
<bool name="furnaceOn" val="true"/>
</obj>
```

Figure 3.11: oBIX XML example: extracted from [49]

oBIX versus QLM

The most significant differences between oBIX and QLM messaging are listed below:

1. oBIX is based on synchronous client-server communication. Therefore it lacks relevant features such as time-to-live and other features that are essential, e.g., for message persistence, which is a CL_2M requirement. QLM has these features in its messaging interface schema.

2. There is no support for defining other destination(s) for messages than the current server that receives the message. In many practical IoT systems, messages may need to be routed between several different networks before reaching their final destination.

3. There is no support for the callback functionality; the subscription mechanism implemented by oBIX "Watches" always require the client to poll for the results of subscriptions. This is a limitation for server-to-server communication or machine-to-machine communication in general. This is a major handicap for implementing CL_2M information flows.

4. Only HTTP and SOAP can currently be used for oBIX, while QLM Messaging Interface messages can be transmitted in different ways.

5. It is not possible to set up time-limited subscriptions. oBIX "Watches" (subscription) always have to be removed by the client. The only exception is that it is possible to provide a timeout, after which the watch is removed if the client has not polled for values before the timeout expires. However, time-limited subscriptions, potentially with different sampling intervals for different kinds of data, are essential (e.g., for remote error diagnostics applications).

6. oBIX does not define a devices service for getting a list of available information sources as shown in Figure 3.11. QLM provides the list of available information sources through Object elements and InfoItems in QLM Messaging Format as shown in Figure 3.6.

7. oBIX does not provide an extension mechanism for using external XML Schema when non-oBIX data needs to be transmitted. Only oBIX Contracts may be used for doing so. With the QLM Messaging Interface, it is possible to include oBIX information as message but it is not clear how and if oBIX could be used for transmitting data that is not oBIX formatted.

8. The stated scope of the OASIS oBIX Technical Committee is limited to Building Automation. The oBIX specification utilizes web services for the exchange of information about the mechanical and electrical systems in commercial buildings. Therefore, it seems unlikely that it would be feasible to introduce all CL_2M requirements and features into oBIX.

Despite these weakness of oBIX, the oBIX objects *can* be used as a payload format in QLM Messaging Interface messages. It can be used instead of the QLM Messaging Format, for instance, in building related information systems that have the support for handling oBIX object structures as shown in Appendix C (Figure C.1).

3.4.2 JMS

Java Messaging Service provides messaging features to support the messaging needs of distributed enterprise applications. JMS defines a Java API for messaging that enables loosely coupled, asynchronous and reliable communication. A JMS provider can deliver messages to a client as they arrive; a client does not have to request messages in order to receive them. Higher levels of reliability are available for applications that cannot afford to miss messages or receive duplicate messages. If there is no receiver for a message while sending the message, the message will be stored in the JMS Server and will be delivered to the receiver(s) when it comes up next time [46]. Clients use the message implementations supplied by their JMS provider. A major goal of JMS is that clients have a consistent API for creating and working with messages independently of the JMS provider.

The JMS messaging products can be broadly classified as either point-to-point (PTP) or publish-subscribe (Pub/Sub) systems. PTP products are built around the concept of message queues. Each message is addressed to a specific queue; clients extract messages from the queue(s) established to hold their messages. Pub/Sub clients address messages to some node in a content hierarchy. Publishers and subscribers are generally anonymous and may dynamically publish or subscribe to the content hierarchy. The system takes care of distributing the messages arriving from a node's multiple publishers to its multiple subscribers.

JMS messages are composed of an header, properties and a body. All messages support the same set of header fields, which contain values used by both clients and providers to identify and route messages. In addition to the standard header fields, messages provide a built-in facility for adding optional header fields to a message such as application-specific properties, standard properties and provider-specific properties. JMS defines several types of message body, which cover the majority of messaging styles currently in use.

The JMS message model has the following goals [50]:

- Provides a single, unified message API.
- Provides an API suitable for creating messages that match the format used by existing, non-JMS applications.
- Supports the development of heterogeneous applications that span operating systems, machine architectures, and computer languages.
- Supports messages containing Java objects.
- Supports messages containing XML pages.

The whole specification just describes what Java classes and methods to call and use. However, it does not provide information about the underlying protocol.

JMS versus QLM

The most significant differences between JMS and QLM messaging are listed below:

1. JMS is an API based on Java whereas QLM is messaging standard that can be implemented with any language supporting text processing.
2. In JMS, message generation is hidden from the developer as it provides interfaces for creating message through an API. Here the developers have less flexibility while creating the messages. On the other hand, QLM developer have full control for creating and implementing message transfer for any products or systems because messaging interface does not provide the API but defines the standard.
3. QLM messaging can provide different read, write and cancel operations and it specifies the services of the Objects and its sub elements. The JMS API does not specify such operations or services.
4. JMS is "tightly coupled" with Java, and notably J2EE. This implies it can only run on server-class machines. This might present difficulties for

implementing interoperability concept in CL_2M. The QLM Messaging Interface allows QLM nodes to have any role (either server or client) and communicate in a peer-to-peer manner.

5. In JMS, the format of the message is not known beforehand. As a developer, the API provided by JMS must be used to create the message and to perform the necessary computation. In contrast, the QLM Messaging Format and the QLM Messaging Interface are known in advance and its structure is validated by their corresponding schema files. One can use any language (as implementing tool) for creating and parsing the message.

6. In JMS, a message does not dictate what the recipient should do and sender does not wait for a response. This means that JMS does not acknowledge the request whereas QLM has the capability of sending response message (see Figure 3.3 and Figure 3.4).

7. JMS messaging provides guaranteed delivery via the once-and-only-once delivery semantics of persistent messages. QLM supports it implicitly with read/write, callback, ttl, etc functionality. However, QLM does not really specify what guaranteed delivery mechanism to use; that is mainly up to the implementation. So, different QLM Message Interface implementations may provide different mechanisms according to application requirements.

8. JMS does not define a schema of systems messages (such as delivery notifications) for request acknowledgement. CL_2M has feedback functions and requires delivery notification of messages.

9. The conceptual framework used for the JMS architecture is the Publish-Subscribe design pattern. QLM on the other hand uses the Observer design pattern, which means that a QLM node can add itself as an observer of events that occur at another QLM node (See Figure 3.12). For many applications the Observer and the Publish-Subscribe models can be used in quite similar ways. However, the Publish-Subscribe model

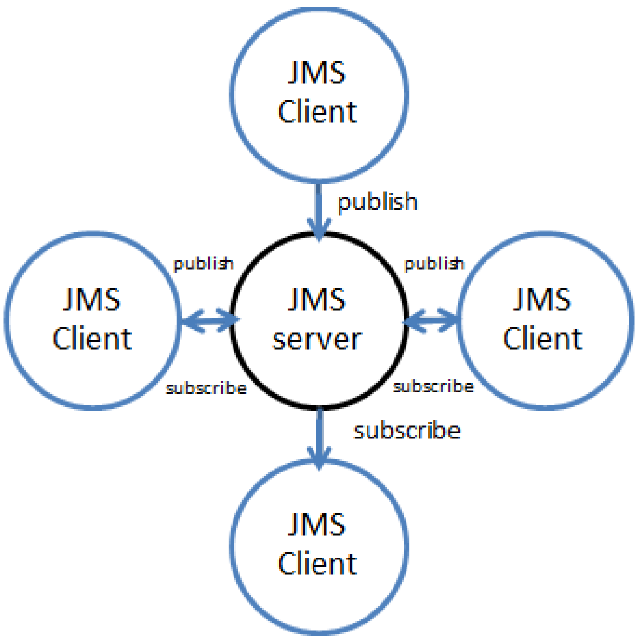
usually assumes the usage of a high-availability server, which the Observer pattern does not. In that regard, the Observer model is more suitable for IoT applications where products might communicate with each other directly.

From the above presented list, it can be noted that there are substantial weaknesses in both oBIX and JMS, which make them unsuitable for the requirements of IoT and CL_2M. QLM on the other hand provide all the functionalities needed for IoT application. QLM is developed focusing for IoT and intelligent products considering the features of CL_2M such as feedback, information visibility, traceability, information persistence and interoperability. JMS is developed mainly for integrating distributed enterprise systems, which does not entirely cover the needs of CL_2M.

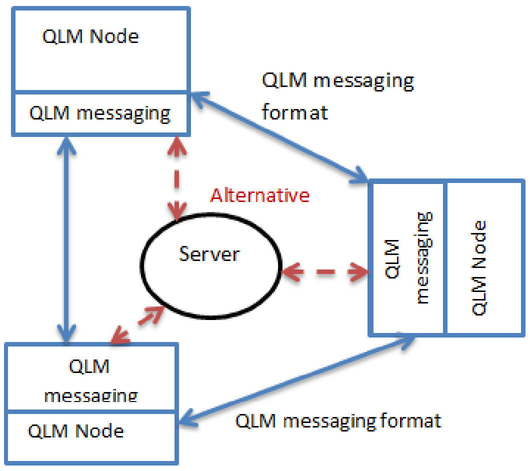
3.5 Assessment of QLM messaging

In CL_2M, information such as sensor readings, alarms, assembly, disassembly, shipping events, and other information related to the entire PLC need to be exchanged between several organizations. A suitable standard for exchanging such information is required. The QLM Messaging Interface and QLM Messaging Format specifications are proposed to meet the CL_2M requirements of PLC information exchange. It is possible to implement QLM messaging standard for any kind of information systems, including embedded and mobile systems. Figure 3.13 shows how information about the things and instances are exchanged using the QLM messaging standard.

In Figure 3.13, the ‘thing’ has been illustrated in the same way for the different PLC phases such as an idea, a set of CAD drawings and so on. During the design phase, the thing is a collection of ideas, design documents, etc. that may even be spread over several organizations. In the manufacturing phase, the thing is a set of parts and sub-assemblies that may be manufactured by different organizations. The thing that the consumer buys and uses is then the tangible result of all the previous phases and of its usage history. The corresponding product information tends to be spread over different organizations, geographical locations and information systems. Thus, the QLM messaging



(a) Publish-Subscribe design pattern in JMS



(b) Observer design pattern in QLM messaging

Figure 3.12: a. Design pattern in JMS and b. Design pattern in QLM messaging

standard can be used for the communication between these different information systems, accessing product information in different organizations and enabling all kinds of intelligent products for information exchange in ad hoc

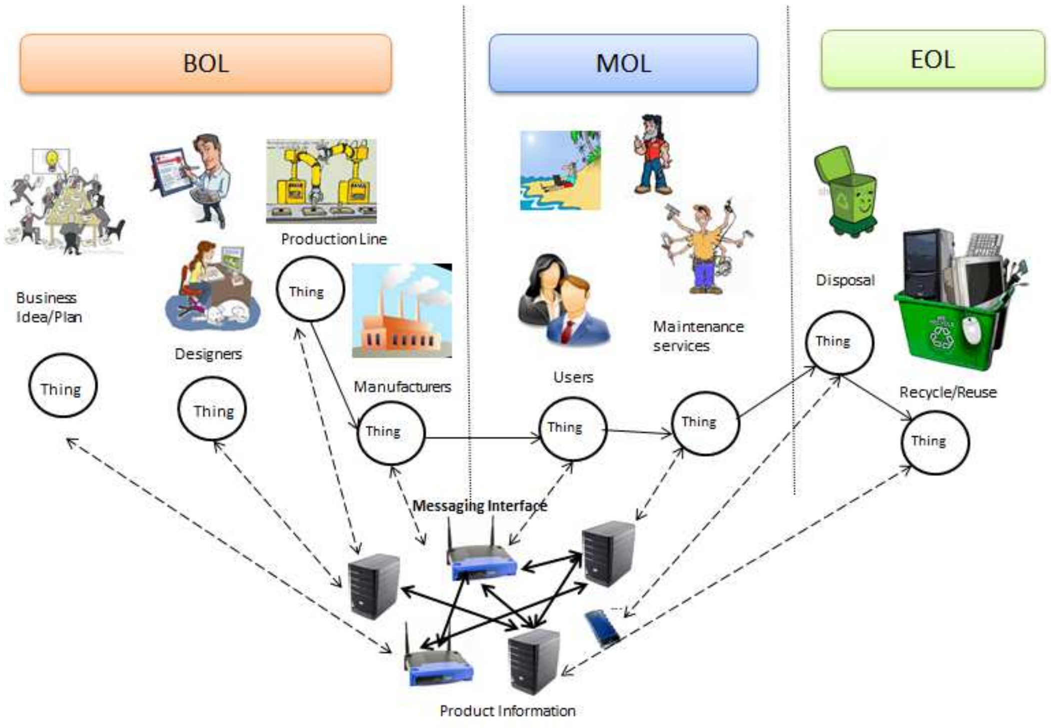


Figure 3.13: QLM messaging for CL_2M in manufacturing industry: adopted from [51]

and loosely coupled ways. Hence, the requirement of seamless information flow and feedback between PLC phases in CL_2M can be fulfilled by QLM messaging standard.

Chapter 4

Application examples

Synopsis

We present two application examples in this chapter. We will take both the case scenarios from automotive manufacturing industry. The first case scenario is for improving product quality and process management in the production line. The second case scenario is for optimizing the Lifecycle Cost (LCC) of the assembly line. These examples illustrate the application of QLM messaging standards for information visibility and flow in manufacturing facilities.

4.1 QLM in automotive industry

There are designers, manufacturers, distributors, service providers, and recyclers involved in the PLC. Each has her/his own existing information applications that serve different value chains. Each application defines its own data presentation and software integration, which may be incompatible with each other. Therefore, to share the information derived in all organizations, a common information exchange standard is necessary. As shown in Figure 4.1, QLM can be used as a common information exchange standard, which makes it possible to exchange messages in a common format that is understood by the different systems and organizations involved in the PLC. Using QLM the manufacturer can track the product information in a distributed network and can share meaningful information about it and its relationship with other products. This means, for example, that one is able to find out the information about the users involved with a specific product throughout the whole PLC.

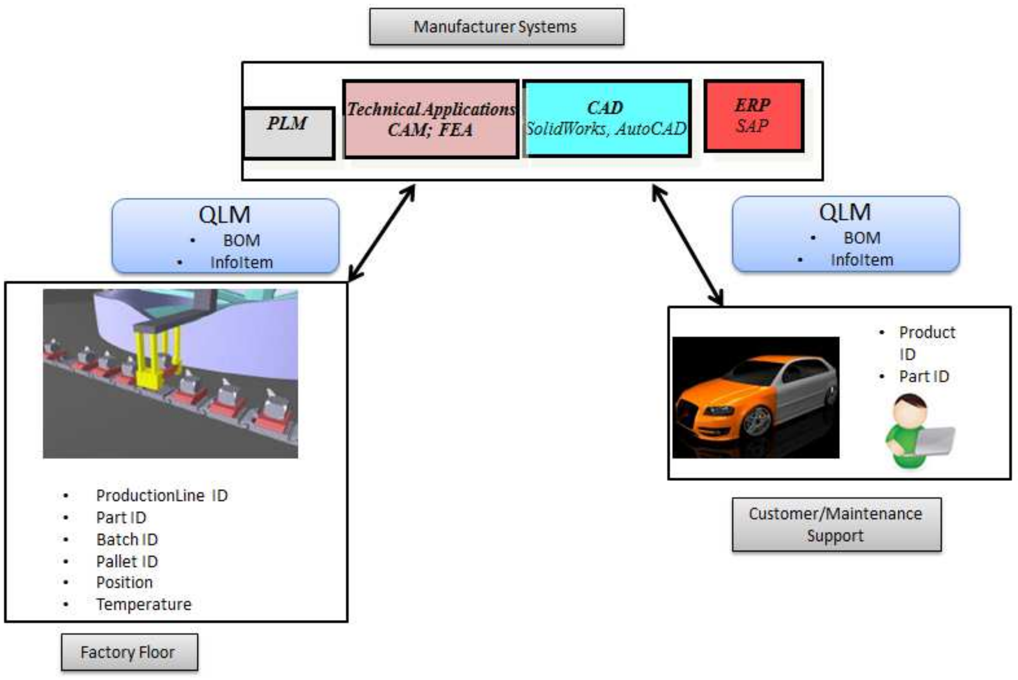


Figure 4.1: QLM Messaging Interface for the integration of different PLC systems

In the automotive industry it is usually the manufacturers' responsibility to organize the lifecycle management strategy. The environment in which the product will have its lifecycle is largely determined already in the manufacturing stage. Although the application cases we present in this thesis have an initial BOL phase focus, they are also concerned with increasing the feedback of MOL and EOL lifecycle data. The information from BOL is useful for the automobile's customers in MOL and for the dismantling markets in the EOL phase.

In the first application case, the defect in products during the production process costs more to the case company, as they may have to throw away an entire batch. Thus, to manage the quality and the performance, a better traceability of the information generated during the production process is required.

In the second application case, to determine the LCC of the assembly line, the case company wants to collect and analyse the information produced in all the PLC phases of an assembly line, including the information from robotic machines.

4.1.1 Application 1: quality and process management

The first case company is a European car manufacturing company. The company wants to improve production quality control. To do so, they allow traceability of process parameters and correlation of that data with fault detection for process optimization. They even want to reuse the knowledge and to exploit it for the design of future tools or for optimizing the design of work pieces.

In this application example, the focus is given on the improvement of the product quality from the hot stamping process (Figure 4.2). The hot stamping process lines produce large number of parts every day, which places high requirements on quality and process stability.

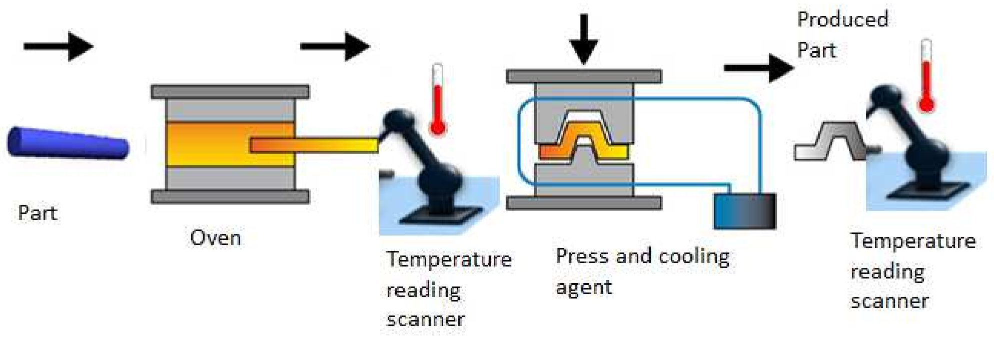


Figure 4.2: Hot stamping process

There are various user groups responsible for the production process. Product information flow between them is challenging and unmanaged. The responsibilities of user groups in hot stamping process are as follows:

- Shop floor operator: He pin points the error and remedies the error causing the defects in pieces.
- Shop floor manager: He checks the current production status and the

historic view. The shop floor manager performs decisions regarding the change of production parameters in order to improve the process quality.

- Production manager: He checks Key Performance Indicators (KPIs).
- Part designer: He works on the CAD design of work piece.
- Quality manager: He reports the total number of good and bad parts during production.

The shop floor operator is currently not able to detect the error. To determine the source of errors he wants to get the real time parameters of the process in order to remedy the error immediately. Accordingly, the operator wants to get a text message or an email message when a defect occurs or when the temperature is below the minimum value allowed after the oven and above the maximum value allowed after the press. Then, the shop floor manager and the quality manager want historical data displayed in a graphical way to make decisions on how to avoid the defects.

Furthermore, in the production line, the information gained during the process is not collected within a single database and is not evaluated automatically. This means that merely statistical samples are evaluated and the part quality cannot be assessed in-line with the production process. Then, the main errors that occur during production are errors in the geometry of the final part. Moreover, errors might occur that render the product useless. The testing procedures for this task becomes a time-intensive procedures and causes defect in a whole batch of products because the error cannot be solved instantaneously. This might be prevented through an improved exploitation of all the available process information.

According to the interviewee from the company, the causes can be identified by correlating defects (the quality measurements) with the actual production process (which machines are used, oven temperature, dwell time, pressure, etc.) and design/engineering information (geometry of the part to produce). To gather the data, the company uses scanners after every process and a geometrical measurement measuring machine from Trimek [1]. For the communication between scanners to responsible user's systems, machines to systems and

[1] http://www.trimek.com/en/

systems to systems, the QLM messaging standard seems to be appropriate. The information flow from production line to the manufacturer's or engineer's systems is illustrated in Figure 4.3.

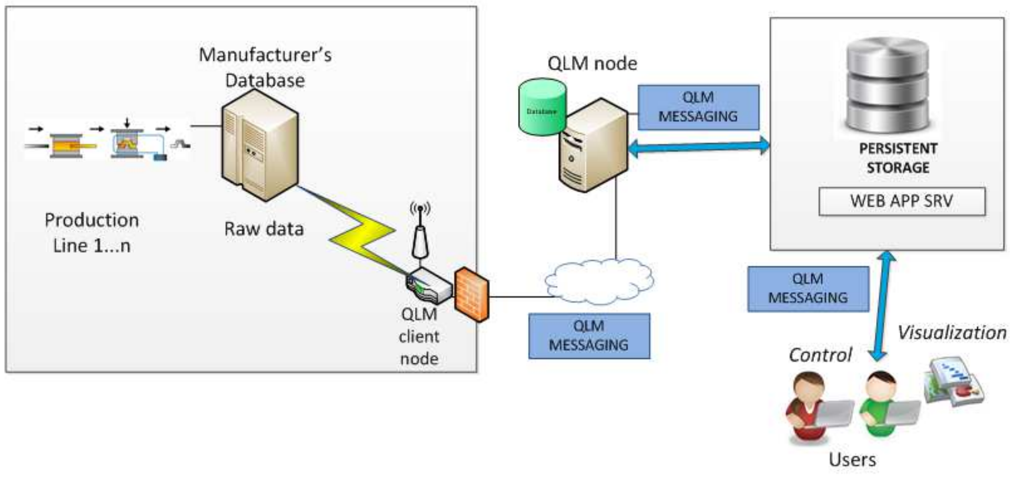


Figure 4.3: Information flow from factory floor to the users

The data generated by the scanner is stored in the manufacturer's database. The raw data can then be accessed by QLM client nodes. The QLM client node works as the interface of the manufacturer's network with the outer world. The communication between the QLM client node and the QLM node uses QLM messaging. In addition, the message passing from the QLM node to the web application service and from the web application service to the visualization tool also uses the QLM messaging. Based on the information shown in the visualization, the users could make decisions and control the defects in the hot stamping process line.

4.1.2 Application 2: estimating lifecycle cost

The second case company is a global supplier of industrial automation systems and services mainly for the automotive manufacturing sector. The company is organized into five business units which are; Body Welding and Assembly, Powertrain Machining and Assembly, Robotics and Maintenance Services, Aerospace Production Systems and Adaptive Solutions.

The case that we are working on is automotive Body Welding and Powertrain plants that are composed of several automatic lines, each of which is used for performing several operations on the part to produce. The life of each automatic line must be carefully planned because it will produce an entire series of products, representing ten or fifteen years of operation. The proper evaluation of the Life Cycle Cost (LCC) is essential for manufacturers in order to choose the best solution regarding their production activities. LCC is the sum of all cost factors over the expended life of product machinery. These cost components can be divided according to the classification given in Figure 4.4.

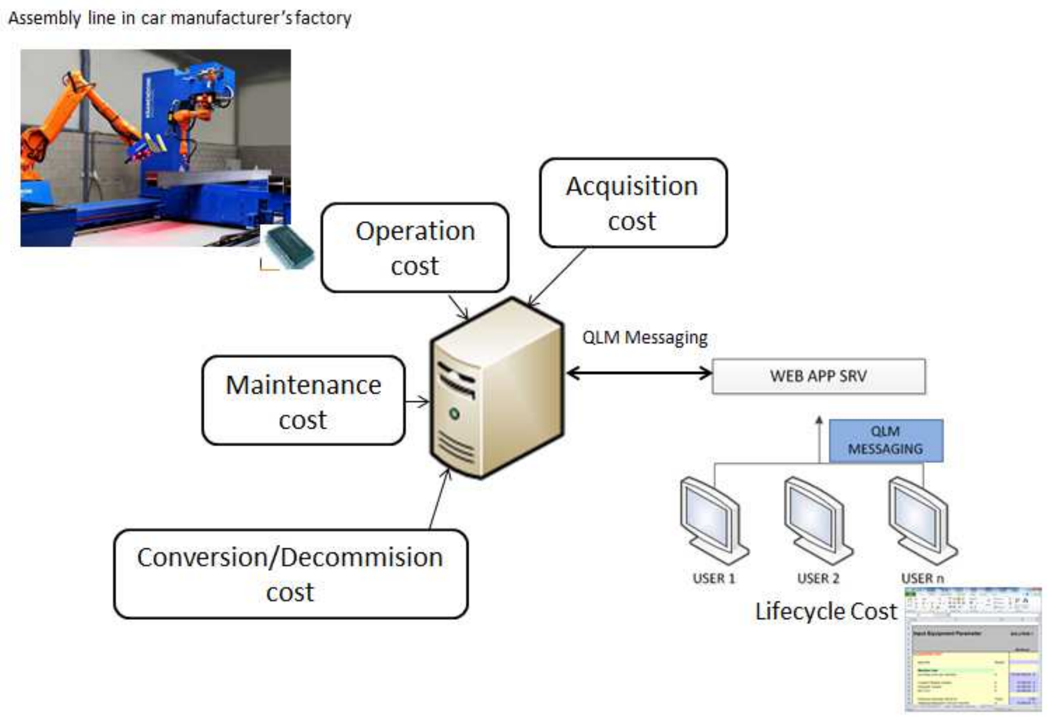


Figure 4.4: Assembly line LCC estimation

The objective of this case is to calculate and optimize the LCC of automatic assembly lines. Applying the CL_2M concept means identifying and summing all costs associated with the system's life cycle. According to the interviewee from the company, up to 95 % of the total LCC is determined by decisions made during the concept and design phases. Hence, an application of LCC analysis is more effective in the product's early design phase to optimize the basic design approach. However, it should be used during the subsequent phases of the PLC to optimize other engineering decisions and to facilitate the efficient allocation of resources.

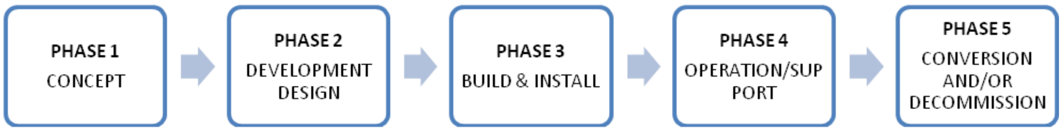


Figure 4.5: Phases of manufacturing Machinery and Equipment Life Cycle

The added value can be found in each PLC phase (see Figure 4.5). During the Concept phase, which is very short but the most strategic one, a tight collaboration with the end user allows to specify the base solution. In the Proposal phase, it is possible to show more competitive solutions to win the competition. The Design phase improves the design quality since parameters are taken into account, which are not normally considered. From Build and Install phases, the collection of data starts and the feedback is a good indicator to evaluate the data. At the end of the Operation phase, a continuous comparison with analyses done during preceding phases updates the database from field data acquisition. According to the case company, 50 % of the total LCC is covered by operation and support.

The calculation of the LCC is performed considering the desired design of the production plant and the analysis of the historical data related to the reliability of each single component or subsystem with the support of proper computer simulation activities. To perform such an evaluation, an appropriate database of consistent data, the history and lessons learned must be consulted. It is particularly important that these data come from real field experiences, but this is particularly difficult because of:

1. Missing statistical information regarding device reliability.
2. Incoherence in the data retrieved from the field.
3. Large database and hardware required.
4. Difficult access to data.

QLM can be used for problems 1 and 2 listed above. In order to have efficient operational management, the use of intelligent products and QLM messaging may turn out to be suitable solutions [52]. Problem 3 is presented as a future research topic in this thesis. Problem 4 is an organizational issue rather than a technical issue and, thus, is not covered by QLM messaging.

The case company provides assembly lines that consist of machines and equipments from different organizations and suppliers. As each system implements its own messaging format, making the transfer of data between the various equipments is very hard if not impossible and it would be a very challenging task to manage all these data at receiver's end. In order to solve this issue, the QLM messaging is implemented in each system (designer system, operation department system, maintenance system, etc.) as shown in Figure 4.6. This solves the data inconsistency problem and makes the communication between different systems possible. The preprocessed data after collecting from all the systems are stored in the company's database. As discussed in the previous application case, the QLM client node is used as the interface between the company's network and the outer world in this case as well. The middleware then transmits the information to the web application service, which provides all the information required to calculate the LCC.

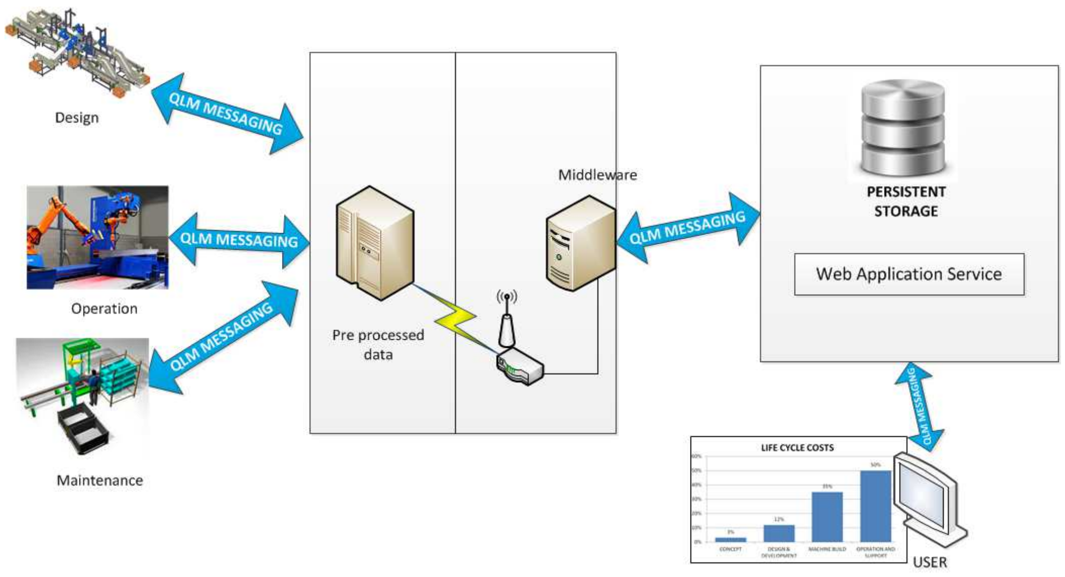


Figure 4.6: QLM Messaging Interface for interoperability scenario

4.2 Current implementation of QLM messaging

The PLC information from BOL to MOL and to EOL can be studied with the help of the QLM Messaging Interface, the Internet and the web application services. The engineers, manufacturers and service providers who want to study the product information may develop their own way of visualization tools (graph, charts, images) using different APIs. The architecture is illustrated in Figure 4.7. The QLM Messaging Interface exchanges all the product related information through the PLC with the web application service and the Internet using HTTP request and response messages. The users can have various visualization forms of the product information that they are interested on their computers.

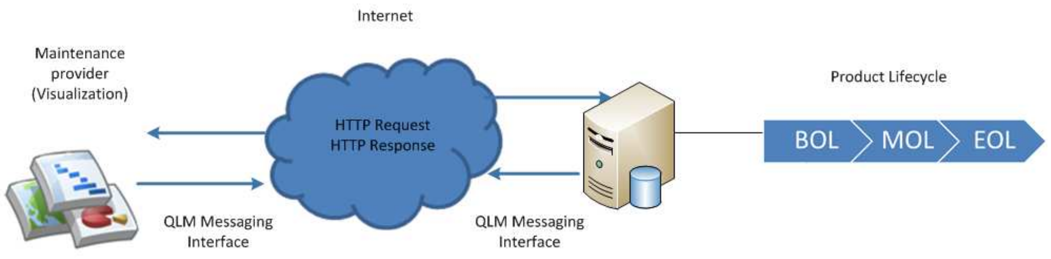


Figure 4.7: QLM messaging visualization of product related information

This section presents the current implementation of the QLM Messaging Interface and the QLM Messaging format for the case examples we have presented earlier. The current implementation is done mainly for the application example 1. As mentioned in Section 4.1.1, there are many factors that can cause defects in the production line. In our implementation the temperature data is displayed, which may be one variable for causing the defect in the part after hot stamping process.

For the current implementation, QLM Messaging Interface is not implemented at the client's system. So, there is no QLM client node. Instead, the case company provided us the processed temperature data in a USB drive. The temperature data of the production line machines was extracted from the SQL database, which was provided in an Excel file. The scanners data was provided in a text file with the path of the image of the parts after the hot stamping process. The data is sent to the web services from the server where the QLM Messaging Interface is implemented with a payload adhering to the QLM Messaging Format. The examples of the QLM messaging implementation are presented in Figure 4.8 and Figure 4.9 (see more Appendix B, Figure B.1).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<qlm:qlmEnvelope xmlns:qlm="QLM_mi.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="QLM_mi.xsd QLM_mi.xsd" version="0.2" ttl="-1">
<qlm:write msgformat="QLM_mf.xsd">
        <qlm:msg>
        <Objects xmlns:xsi="http://www.w3.org/2001/
        XMLSchema-instance" xmlns="QLM_mf.xsd"
         xsi:schemaLocation="QLM_mf.xsd QLM_mf.xsd">
        <Object>
                <id>ProductionLine1</id>
                <InfoItem class="CurrentTemperatureValues">
                        <display>Realtime Production Line Temperatures</display>
                        <value type="CSV">
                                238,64,738,745,792,805,846,884,917,924,
                                745,32,35,29,24,456,598,605</value>
                </InfoItem>
        </Object>
        </Objects>
        </qlm:msg>
</qlm:write>
</qlm:qlmEnvelope>
```

Figure 4.8: QLM write message for real time monitoring of temperature data

The development of the User Interface (UI) for the current implementation is shown in Figure 4.10. This UI shows the visualization of the real time temperature information of the hot stamping production line. The information that can be visualized from the web page in Figure 4.10 are as follows:

- The real time temperature data is displayed on the page, which is updated every 5 seconds.

- The picture of the scanner temperature can also be generated and en-

```xml
<?xml version="1.0" encoding="UTF-8"?>
<qlm:qlmEnvelope xmlns:qlm="QLM_mi.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="QLM_mi.xsd QLM_mi.xsd" version="0.2" ttl="-1">
<qlm:write msgformat="QLM_mf.xsd">
       <qlm:msg>
       <Objects xmlns:xsi="http://www.w3.org/2001/
       XMLSchema-instance" xmlns="QLM_mf.xsd"
        xsi:schemaLocation="QLM_mf.xsd QLM_mf.xsd">
              <Object>
                     <id>ProductionLine1</id>
                     <InfoItem class="LowPartTemperature">
                            <display>Minimum Temperature from
                            Scanner1 with position</display>
                            <value dateTime="2001-10-26T15:33:21"
                     type="CSV">201,x,y,z,FilePathforImage</value>
                     </InfoItem>
              </Object>
       </Objects>
       </qlm:msg>
</qlm:write>
</qlm:qlmEnvelope>
```

Figure 4.9: QLM write event message for temperature data with file path of part image

larged for inspection.

- The color scale bar can be used to indicate the temperature in the picture (blue for cold and progressively red for hot).
- Based on historical data, if some values are out of range with a given percentage it will blink while displaying a message indicating the associated reason.
- The production line statistics widget allows the user to retrieve historical data for within a time period statistical analysis. Clicking on the button "export data" downloads all the data regarding the production line within the time period selected. The data downloaded will be in an appropriate input format for use in statistical analysis applications, such as R, Matlab, etc.

Data visualization is another widget, as shown in Figure 4.11. It shows

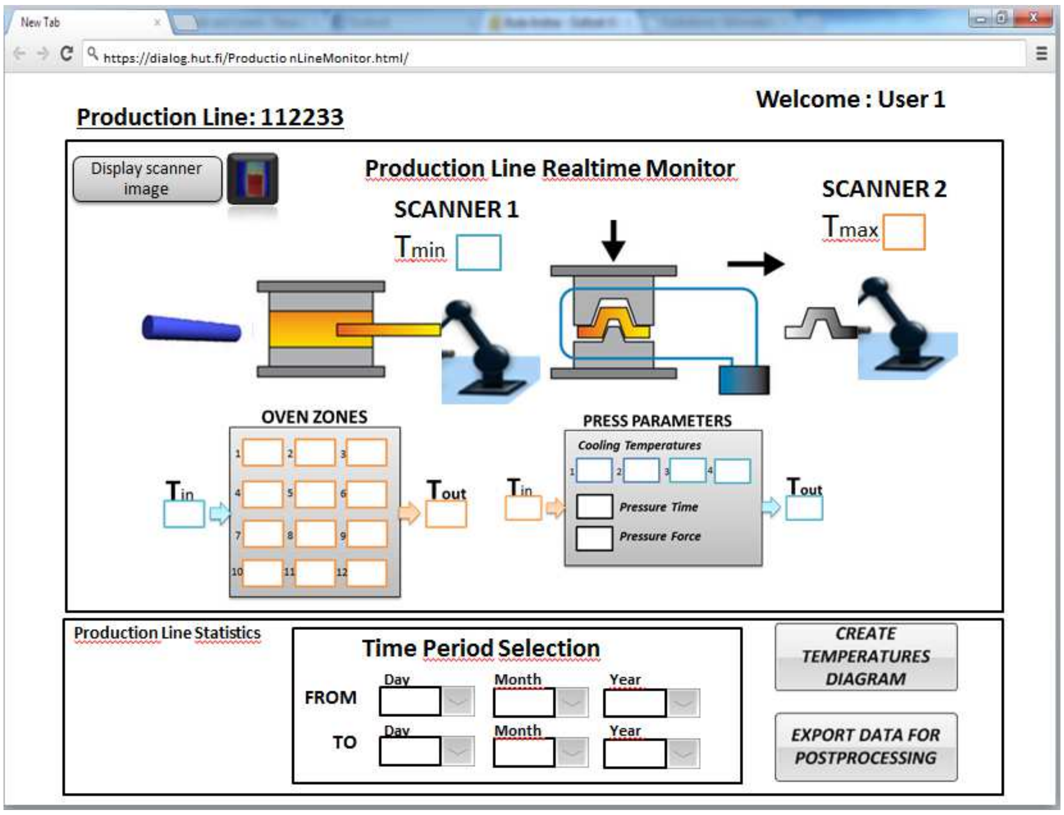
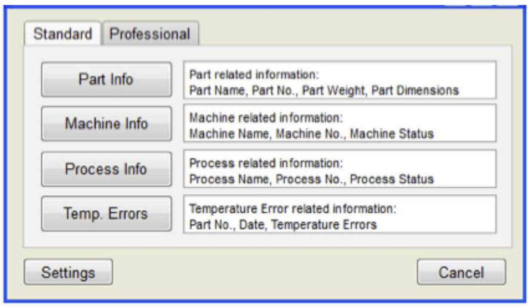


Figure 4.10: Web page example for the visualization of data from the production line

the graphical representation of the scanner's data and also displays the oven or press graphical data when an event occurs. In the ordinary case, a line plot with two curves are displayed; one of maximal temperature of Scanner 2 (behind the press) and one of minimal temperature of Scanner 1 (behind the oven). If there is a fault (maximal temperature above limit or minimal temperature below the limit) the corresponding curve section is highlighted (e.g., change line color from black to red). Pop ups with the scan image are displayed. A line plot of the oven zone temperatures before and during the fault could be displayed in a pop up as well. A message could also be sent via SMS to all directly influenced persons on the shop floor. In addition to that an email could be sent after the occurrence of failure. The write operation of QLM Messaging Interface is used for exchanging the temperature data for all purpose as shown in Figure 4.12.

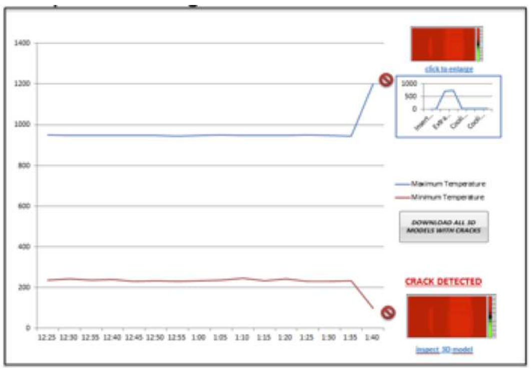


Standard | Professional

Part Info	Part related information: Part Name, Part No., Part Weight, Part Dimensions
Machine Info	Machine related information: Machine Name, Machine No., Machine Status
Process Info	Process related information: Process Name, Process No., Process Status
Temp. Errors	Temperature Error related information: Part No., Date, Temperature Errors

Settings | Cancel

(a) Historical data



(b) Graphical representation

Figure 4.11: Example widgets in the web page

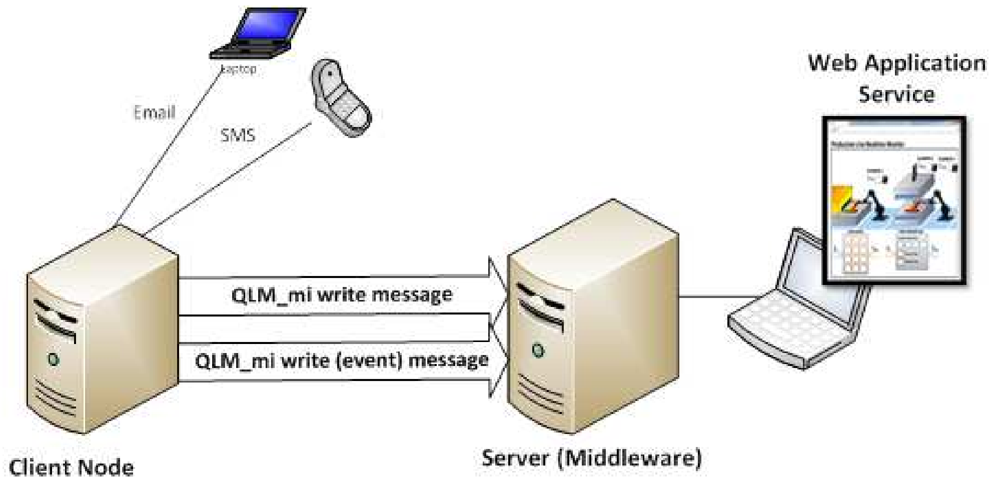


Figure 4.12: Getting real time data in the web application using QLM write operation

The user interface developer can develop a web page with many widgets and information displayed in a single page in a similar way as iGoogle. Information from the design phase, such as CAD documents, can also be transferred into the same page using QLM Messaging Interface. The data collected during the MOL (data of usage and maintenance) can also be transferred by QLM messaging and be used for creating various charts such as performance graphs and LCC charts. The page can also display EOL information (disposal or recycling) so that it is helpful in the development of new product, which can have smaller maintenance requirements.

Chapter 5

Future research

Synopsis

The issues found during the implementation and application of intelligent products is that the data generated in every few seconds will result in big data in the future. Big data is a collection of large and complex data sets that are difficult to query and process using traditional data processing applications. Then, for storing the data for future subscription and historical data, the database needs to be huge and this will be expensive. Therefore, future research on cloud based approaches might make it easier to access, process, query and search information from big data. In our research, we looked at the cloud based approach called Hadoop to resolve big data issues for the first application case example. This approach is discussed in this chapter.

5.1 Big data solution: Hadoop integration

CL_2M will boost the productivity of today's engineers by providing a holistic view on data, persons and processes across the full PLC as a vital resource for outstanding competitive design of novel products and manufacturing processes. However, the issue of big data is found during the application of PEIDs and intelligent products. Big data is a large and complex data that is challenging to store and process for traditional database systems.

The cloud based approach called Hadoop might enhance CL_2M's flexibility by allowing the processing (data analysis, querying and storing functions) of big data gathered during the PLC. With IoT, the collection of a large amount of data from different objects is expected to be commonplace. New opportunities to store the data include the use of cloud services and novel no-SQL database technology. If there is a need for storing and querying big data, Apache Hadoop's HDFS (Hadoop Distributed File System) and MapReduce can be used. There is also an Apache project called Apache Hive, which is a datawarehouse framework that can be used in querying file attributes using SQL. If the data is large and a fast query time is required, such as microsecond time, then Solr Index server is used as the search engine. Solr is a popular, highly scalable and fast open source enterprise search platform from the Apache Lucene project.

Hadoop is an open source software platform for distributed processing of large data sets across clusters of computers using a simple programming model. It is an Apache Software Foundation project written in Java and originally created by Doug Cutting [53]. Apache Hadoop, was designed to solve a different problem: the fast, reliable analysis of both structured data and complex data. Many enterprises deploy Hadoop alongside their legacy IT systems, which allows them to combine the old and the new data sets in powerful new ways. Hadoop is actually a load balancing architecture for cloud computing used to distribute the application data, and to parallelize and manage application execution across computers. It is widely used in finance, telecom, media and entertainment, government, research institutions and other markets with significant amounts of data [54]. With Hadoop, enterprises can easily explore complex data using custom analyses tailored to their information and questions. It complements existing data management solutions with new analyses and processing tools to:

- Retrieve and organize big data sets regarding the usage of production line components.
- Post process big data sets in order to extract meaningful statistical information.
- Maintain data integrity using a relational database for long term storage.
- Multiple computers instead of single computer.

- Fast processing and sorting of large data.
- Off-line batch processing.
- Load balancing.
- Can be used in projects where in future there will be growth of data.

Hadoop provides a reliable shared storage and analysis system. The storage is provided by HDFS, and the analysis by Hadoop MapReduce. So, HDFS and Hadoop MapReduce are the most commonly used sub-projects of Hadoop. Hadoop MapReduce is used for processing and extracting knowledge from large data sets on computer clusters [55]. It is an application framework that allows programmers to write their own map and reduce functions to process their data. MapReduce inputs typically come from input files loaded onto a processing cluster in HDFS. These files are evenly distributed across all nodes. Nodes in the cluster have mapping tasking running on them. When the mapping phase is completed, the shuffling and sorting process is done before sending the data to reducer. The reduce tasks are spread across the same nodes in the cluster as the mappers. The reducer gets the sorted and shuffled data. Then, the reducer process generates the outputs, which can be stored locally.

Hadoop is open source and has an active community so that developers can freely modify Hadoop to add custom features and patches and it provides a flexible framework for running distributed computing algorithms with a relatively easy learning curve.

5.1.1 Application example of Hadoop

Hadoop can be used in projects where the amount of data grows rapidly. For performing data analysis it uses MapReduce. For example, in BOL we can have CAD design locations listed in a text file, map can process the existing cluster to check the required design for a particular problematic car. Then reduce gives the final output file that accumulates all those locations of the CAD design that are relevant. The CAD designs can be stored in HDFS so that it is easy and fast to download the required designs locally.

Hadoop can be used for CL_2M in manufacturing industry for analysing the big datasets generated from production process. Manufacturers have different machines in the production factory floor so they have huge amounts of data that need to be processed. One example of such data is the data generated from scanners in Application Case 1 explained in Section 4.1.1, where the scanners collect the data of each part that has been heated and pressed in the hot stamping process. A simple program can be developed using the MapReduce paradigm in Hadoop. This program gives a number of space-delimited text files containing the machine ID, maximum temperature, dwell time, the number of times error occurred in the product production, the number of times the error in the production process was maintained, and the number of times the error was neglected. The final output files are then accumulated and copied to a server where a web service makes the data available for visualization.

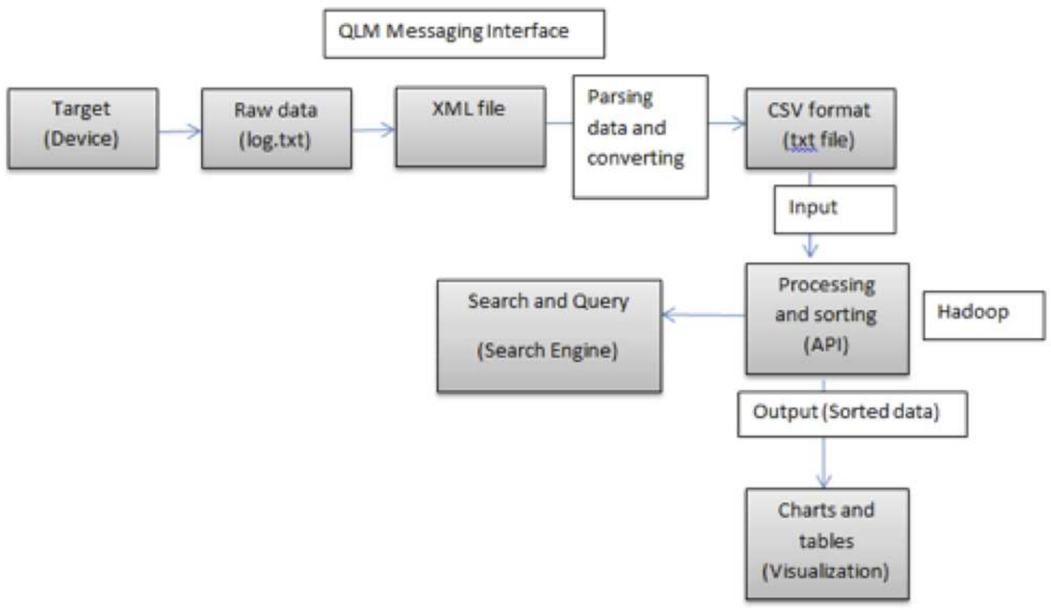

QLM Messaging Interface
Target (Device)
Raw data (log.txt)
XML file
Parsing data and converting
CSV format (txt file)
Input
Processing and sorting (API)
Hadoop
Search and Query (Search Engine)
Output (Sorted data)
Charts and tables (Visualization)

Figure 5.1: QLM Messaging Interface and Hadoop for retrieving, exchanging, processing, searching and visualizing product lifecycle data

The product lifecycle quality and cost optimization is performed considering the desired design of the production plant and the analysis of historical data related to the reliability of each single component or subsystem with the support of computer simulation. To perform such an evaluation, an appropriate database of consistent data on the behaviour, the history and lessons learned must be consulted. Hadoop provides an easy solution to problems such as finding missing statistical information on device reliability, incoherence from the data retrieved from the field, large database and hardware requirement and difficult access to data.

Therefore, Hadoop can be a part of future work in this area of research where the growth in data size is evident. It can be used for fast processing of large amounts of data. Hadoop can generally be used as a database, which processes the big data and stores the data in its masternode (locally) only. This data can then be used globally by using other APIs.

However, in our first application example, the manufacturers are mainly interested in receiving real time information from the production line. Hadoop might not be a good solution for this approach. Hadoop is used for off-line data processing and not for real time because minimum wait time will be 15 seconds to 10 minutes (depending on size of data from few megabytes to terabytes). When there is an error in the production line, then the error information should be received by the responsible person immediately in order to remedy the error and save the batch. If waiting time is less than 10 to 15 seconds, then Hadoop cannot be used. It takes some time to initialize all processes for map reduce and MapReduce is an expensive process. Whereas, in the second application case, the company needs the product information to calculate and optimize the LCC. The company has big data from the production site and is facing the problem of storing and processing the data. In the interview, the interviewee mentioned that when they search a certain information in their system, the processing time is very long and sometimes they even have to restart the system. For this reason, Hadoop might be a solution.

Chapter 6

Conclusions

Synopsis

This chapter provides a summary of findings and discusses the implications of these findings. Reliability and validity of the research are then evaluated.

6.1 Summary of findings

This thesis concentrated on studying product information exchange standards to enable Closed Loop Lifecycle Management (CL$_2$M). The particular interest was to study CL$_2$M and to evaluate a messaging standard that meets the requirements of CL$_2$M.

In the first part of the thesis, we explored the CL$_2$M concept and how it addresses the issues in traditional Product Lifecycle Management (PLM). In traditional PLM, it is found that there is a gap in the product information flow, with no feedback from Middle of Life (MOL) and End of Life (EOL), and lacking information flow between lifecycle phases in general after Beginning of Life (BOL). There are systems such as CAD and CAE for BOL but there are no such systems to collect and transfer the data, information and knowledge produced during MOL and EOL. Therefore, in the EU project PROMISE, the CL$_2$M concept was developed as an extension of PLM by closing the information flow loop. CL$_2$M enables the desired information gathering, processing and exchange throughout the Product Lifecycle (PLC) from the beginning, through the middle to the end of life. CL$_2$M helps to understand the PLC in a better manner than the traditional PLM, as CL_2M focuses on product individuals rather than on product types. However, it requires a communication standard to query and set up information flows between any kinds of products, devices, computers, users and information systems.

The second part of the thesis described the work on developing such a messaging standard to enable seamless information flow and exchange of information in a CL_2M system. This new messaging standard called Quantum Lifecycle Management (QLM) is derived from the PROMISE Messaging Interface (PMI), developed in 2008. In the current specification, QLM messaging is divided into two standards; QLM Messaging Interface and QLM Messaging Format. These standards are independent entities. The QLM Messaging Interface is a communication interface, while the QLM Messaging Format is a format level specification. The standards are applied in the example of monitoring appliance power consumption to illustrate the implementation of QLM. Furthermore, two automotive industrial application cases were studied for the evaluation of QLM messaging standard to enable seamless PLC information flow.

The first application case consists in subscribing for information from the production lines for better quality management and for optimizing production cost. QLM messaging is implemented in the production line systems and provides the information to different users with different responsibilities. The user can then use production information to reduce the error immediately and to optimize cost and quality in production facility. The real-time statistical data, historical data, text messages and alarm emails are exchanged with the implementation of QLM messaging in the manufacturer's systems. Similarly, the second application case consists in subscribing to information from assembly lines to calculate and evaluate Lifecycle Cost (LCC). In the current implementation of QLM messaging, we are able to monitor the production line information using the QLM Messaging Interface and the QLM Messaging Format.

In order to assess the technical features of QLM messaging, we also performed a survey of comparable information exchange standards. The most relevant standards identified are Open Building Information Exchange (oBIX) and Java Messaging Service (JMS). The third part of the thesis compares the

QLM messaging standards with oBIX and JMS standards. oBIX is limited to building automation and JMS is for integrating distributed enterprise systems. In both standards, some limitations and issues were found, which are covered by the QLM messaging standards. QLM messaging standards can be implemented for any kind of product instances as independently of the application domain as possible in a peer-to-peer style, ad hoc and loosely coupled way.

6.2 Implications of the research

Work carried out in this thesis has allowed to assess the suitability of the QLM messaging standard in the framework of CL_2M, and more exactly in the framework of industrial applications.

Close contacts with Industrial companies made it possible to support and supplement the literature statements regarding the CL_2M concept, and especially the fact that a communication standard is required to get the most out of this concept, thus enabling better visibility of product-related information. QLM messaging standards can provide the needed communication interface and message description model in order to exchange real-time statistical and historical data.

The application cases proposed in this thesis show that it is possible to create, gather and exchange the needed information in the manufacturing industry. Moreover, the implementation of QLM can be extended to improve the closed loop information flow and get the data in the desired form. Users on the factory floor can now be provided with the right data, in the right place, at the right time. These studies showed that the implementation of QLM for production facilities offers better data interoperability between products and with all information systems that consume or provide relevant information during the PLC. This helps in increasing product value, and in improving quality and performance of existing and future products. Despite the focus on product lifecycles, it is our intention that the QLM messaging standard would be applicable to lifecycles of anything, such as humans, services, projects, electronic documents, etc.

The second main contribution of this thesis is the comparison of the QLM messaging standard with other existing relevant standards, namely with JMS and oBIX. A major difference is that QLM is developed using the Observer design pattern, which does not use a server-based philosophy. The QLM node may have both server and client functionality. Whereas, JMS uses a server-based publish-subscribe model with heavy Java-based server implementations that make it difficult to realize machine-to-machine communication for low-range computing hardware. Similarly, the oBIX specification is intended to be used for the exchange of information about the mechanical and electrical systems in commercial buildings. oBIX is specified only for HTTP and WSDL/SOAP protocols and it does not have support for callback that is needed for instance with real-time notifications. JMS and oBIX also lacks other features of QLM messaging. For instance, QLM provides a functionality to exchange information even behind the firewall, which is not possible with JMS and oBIX.

Therefore compared to existing standards, QLM is a generic application level standard that can fulfill the IoT requirements to query and set up information flow between intelligent products, information systems, computers and devices. When there is a need for event based access to different information QLM can be used. It also supports document related events such as “document created”, “document updated”, “document deleted” and so on.

In the future work section of the thesis, the issue of big data related to IoT is also addressed. The cloud-based approach called Hadoop is briefly explained as an approach for solving the big data issue.

6.3 Reliability and validity of the research

CL_2M can be the extension of the traditional PLM with feedback but this concept currently does not deal with the organizational issue of not allowing to access the information generated from stakeholder’s systems. There are different stakeholders, suppliers and organizations involved in the PLC, with different data sources and databases. However, the technical issues of integrating different applications and systems and exchanging information can be solved by the QLM messaging standard. Then, issues such as feedback from customers (customers not willing to involve in CL$_2$M), access to stakeholder databases and big data still needs to be considered.

The QLM messaging standard has functionality to request the information behind the firewall by embedding the request in the response. This specification is not applied in this thesis because of the IT rules and regulations in the application case companies factory floor. They do not allow any connection with the outer world. So, we tested the data given to us in external storage device (USB drive). However, the design principle of QLM messaging standard for the information flow in factory floor is well described in the thesis which gives the general scenario of usability of QLM.

The QLM messaging standard is still in the development phase. There are many demonstrators implemented in the PROMISE project, where QLM's predecessor PMI has been used. Then, the QLM Messaging Interface and QLM Messaging Format is derived from the PMI. However, QLM is still not fully developed; the complete specification documents of both Messaging Interface and Messaging Format (as Data Model) will be published in The Open Group only in the middle of the year 2013. Our research is into new types of solutions, so it takes time for the key theoretical concepts to become well defined and problem contexts, solution proposals, and descriptions to become well structured.

Thus, this research has mainly qualitative results. However, while presenting the descriptive studies, the relevant concepts and standards were outlined in the description to some extent. This thesis works as the foundation of the development of QLM messaging standard and its usability to enable CL$_2$M expectations.

Bibliography

[1] V. Gecevska, P. Chiabert, Z. Anisic, F. Lombardi, and F. Cus. Product lifecycle management through innovative and competitive business environment. *Journal of Industrial Engineering and Management*, 3(2):323–336, 2010.

[2] O. K. Mont. Clarifying the concept of product–service system. *Journal of cleaner production*, 10(3):237–245, 2002.

[3] K. Främling, T. Ala-Risku, M. Kärkkäinen, and J. Holmström. Design patterns for managing product life cycle information. *Communications of the ACM*, 50(6):75–79, 2007.

[4] J. Anke and K. Främling. Distributed decision support in a PLM scenario. In *Proceedings of Product Data Technology Europe 14th Symposium*, pages 26–28, 2005.

[5] J. Stark. *Product lifecycle management: 21st century paradigm for product realisation*. Springer, 2011.

[6] D. Kiritsis, V. K. Nguyen, and J. Stark. How closed-loop PLM improves knowledge management over the complete product lifecycle and enables the factory of the future. *International Journal of Product Lifecycle Management*, 3(1):54–77, 2008.

[7] M. Frey. Closed - loop product life cycle management using smart embedded systems. PROMISE Interregional Coordinating Partner, 2011.

[8] PROMISE. PROMISE - product lifecycle management and information tracking using smart embedded systems. http://www.promise.no/, 2004-2008.

[9] H. B. Jun, J. H. Shin, D. Kiritsis, and P. Xirouchakis. System architecture for closed-loop PLM. *International Journal of Computer Integrated Manufacturing*, 20(7):684–698, 2007.

[10] H. B. Jun, D. Kiritsis, and P. Xirouchakis. Closed-loop PLM. *Advanced Manufacturing. An ICT and Systems Perspective: An ICTand Systems Perspective*, page 79, 2007.

[11] G. G. Meyer, K. Främling, and J. Holmström. Intelligent products: A survey. *Computers in Industry*, 60(3):137–148, 2009.

[12] LinkedDesign. LinkedDesign. http://www.linkeddesign.eu/, 2012.

[13] D. Kiritsis, A. Bufardi, and P. Xirouchakis. Research issues on product lifecycle management and information tracking using smart embedded systems. *Advanced Engineering Informatics*, 17(3):189–202, 2003.

[14] W. Liu, Y. Zeng, M. Maletz, and D. Brisson. Product lifecycle management: A review. In *Proceedings of the ASME 2009 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference. Presented at the ASME Design Engineering Technical Conferences, San Diego, USA. Retrieved December*, volume 4, page 2009, 2009.

[15] S. G. Lee, Y. S. Ma, G. L. Thimm, and J. Verstraeten. Product lifecycle management in aviation maintenance, repair and overhaul. *Computers in Industry*, 59(2):296–303, 2008.

[16] M. Garetti and S. Terzi. Product lifecycle management: definition, trends and open issues. In *International Conference on Advances in Production Engineering*, 2004.

[17] J. Stark. Product lifecycle management - paradigm for 21st century product realization, 2004.

[18] E. Subrahmanian, S. Rachuri, S. J . Fenves, and S. Foufou. Product lifecycle management support: a challenge in supporting product design and manufacturing in a networked economy. *International Journal of Product Lifecycle Management*, 1(1):4–25, 2005.

[19] Lifecycle management - a business guide to sustainability. United Nations Environment Programme, 2007.

[20] M. G. Marchetta, F. Mayer, and R. Q. Forradellas. A reference framework following a proactive approach for product lifecycle management. *Computers in Industry*, 62(7):672–683, 2011.

[21] S. Terzi, A. Bouras, D. Dutta, and M. Garetti. Product lifecycle management–from its history to its new role. *International Journal of Product Lifecycle Management*, 4(4):360–389, 2010.

[22] H. B. Jun, D. Kiritsis, and P. Xirouchakis. Research issues on closed-loop PLM. *Computers in industry*, 58(8):855–868, 2007.

[23] B. J. Pine and S. Davis. *Mass customization: the new frontier in business competition*. Harvard Business Press, 1999.

[24] M. Garetti, P. Rosa, and S. Terzi. Life cycle simulation for the design of product–service systems. *Computers in Industry*, 2012.

[25] F. Ameri and D. Dutta. Product lifecycle management: closing the knowledge loops. *Computer-Aided Design & Applications*, 2(5):577–590, 2005.

[26] V. Daniel R. Guide, T. P. Harrison, and Luk N. Van Wassenhove. The challenge of closed-loop supply chains. *Interfaces*, 33(6):3–6, 2003.

[27] J. Holmström, R. Kajosaari, K. Främling, and E. Langius. Roadmap to tracking based business and intelligent products. *Computers in Industry*, 60(3):229–233, 2009.

[28] P. Edward and M. L. Eric. Asset management to support product lifecycle management. IBM Product Lifecycle Management, 2009. New York, USA.

[29] M. F. Horstemeyer and P. Wang. Cradle-to-Grave simulation-based design incorporating multiscale microstructure-property modeling: Reinvigorating design with science. *Journal of computer-aided materials design*, 10(1):13–34, 2003.

[30] C. C. Røstad, O. Myklebust, and B. Moseng. Closing the product lifecycle information loops. In *Proceedings of 18th International Conference on Production Research, Italy*, 2005.

[31] D. Kiritsis. Closed-loop PLM for intelligent products in the era of the Internet of Things. *Computer-Aided Design*, 43(5):479–501, 2011.

[32] PROMISE. The information exchange for closed loop lifecycle management. http://cl2m.com/system/files/private/PROMISE AS Volume 3 Architecture Reference PMI.pdf, 2009.

[33] J. Nyman. Product data gathering using a distributed software architecture and product embedded information devices. Master's thesis, Helsinki University of Technology, 2008.

[34] J. Nyman, K. Främling, and V. Michel. Gathering product data from smart products. *ICEIS (1)*, pages 252–257, 2008.

[35] Z. Yang, D. Djurdjanovic, and J. Ni. Maintenance scheduling in manufacturing systems based on predicted machine degradation. *Journal of Intelligent Manufacturing*, 19(1):87–98, 2008.

[36] M. Kärkkäinen, J. Holmström, and K. Främling, K.and Artto. Intelligent products –a step towards a more effective project delivery chain. *Computers in Industry*, 50(2):141–151, 2003.

[37] O Ventä. *Intelligent products and systems: Technology theme-final report.* VTT Technical Research Centre of Finland, 2007.

[38] K. Främling and J. Nyman. Information architecture for intelligent products in the Internet of Things. *Beyond Business Logistics proceedings of NOFOMA*, pages 224–229, 2008.

[39] L. Atzori, A. Iera, and G. Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[40] P. Guillemin and P. Friess. Internet of Things: Strategic research roadmap. *CERP-IoT Project*, 2009.

[41] D. McFarlane, S. Sarma, J L. Chirn, CY Wong, and K. Ashton. Auto id systems and intelligent manufacturing control. *Engineering Applications of Artificial Intelligence*, 16(4):365–376, 2003.

[42] M. Kärkkäinen, T. Ala-Risku, and K. Främling. Efficient tracking for short-term multi-company networks. *International Journal of Physical Distribution & Logistics Management*, 34(7):545–564, 2004.

[43] QLM. QLM: Standards for Quantum Lifecycle Management. https://collaboration.opengroup.org/qlm/index.php, 2012.

[44] K. Främling, M. Harrison, and J. Brusey. Globally unique product identifiers-requirements and solutions to product lifecycle management. In *Proceedings of 12th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, pages 17–19, 2006.

[45] F. Curbera, F. Leymann, T. Storey, D. Ferguson, and S. Weerawarana. *Web services platform architecture: SOAP, WSDL, WS-policy, WS-addressing, WS-BPEL, WS-reliable messaging and more.* Prentice Hall PTR, 2005.

[46] M. Richards, R. Monson-Haefel, and D. Chappell. *Java message service.* O'Reilly Media, Incorporated, 2009.

[47] K. Främling. Standards for Quantumn Lifecycle Management. The Open Group, 2012.

[48] EPC EPCglobal. Information services (EPCIS) Version 1.0. 1, specification, epcglobal, september 2007.

[49] P. Ehrlich and T. Considine. Open building information exchange (oBIX) version 1.0. oasis committee specification, december 2006.

[50] M. Hapner, R. Sharma, J. Fialli, and K. Stout. JMS specification. *Sun Microsystems Inc*, 4150, 2002.

[51] K. Främling, M. Harrison, J. Brusey, and J. Petrow. Requirements on unique identifiers for managing product lifecycle information: comparison of alternative approaches. *International Journal of Computer Integrated Manufacturing*, 20(7):715–726, 2007.

[52] J. Holmström, K. Främling, and T. Ala-Risku. The uses of tracking in operations management: Synthesis of a research program. *International Journal of Production Economics*, 126(2):267–275, 2010.

[53] HadoopMapReduce. http://wiki.apache.org/hadoop/.

[54] Cloudera. What is Hadoop?, 2012. http://www.cloudera.com/what-is-hadoop/.

[55] T. White. *Hadoop: The definitive guide*. O'Reilly Media, 2012.

[56] D. Kiritsis. Product lifecycle management and embedded information devices. *Springer Handbook of Automation*, pages 749–765, 2009.

Appendix A

Main data of information flow

Table A.1: Main data of information flows in the closed loop life-cycle management [56]

Information flow	Information Category	Main data
BOL to MOL	BOM	Product ID, product structure, part ID, component ID product/part/-component design specification, etc.
	Maintenance service	Spare part ID list, price of spare part, maintenance/service instruction etc.
	Production	Assemble/disassemble instruction, production specifications production history data, production routing data, production plan, inventory status, etc
BOL to EOL	Product	Material information, BOM, part/component cost, disassemble instruction, assembly information for remanufacturing, etc.
	Production	Production date, Iot ID, production location, etc.
MOL to EOL	Maintenance history	Number of breakdowns, components ID in problem, installed date, maintenance engineers ID, list of replaced parts, aging statistics after submission, maintenance cost, etc.
	Product status	Degree of quality of each component, performance definition, etc.
	Usage environment	Usage condition(e.g., average humidity, internal/external temperature), user mission profile, usage time, etc.
	Updated BOM	Updated BOM by repairing or changing parts and components etc.
MOL to BOL	Maintenance and failure	Ease of maintenance/service, reliability problems, maintenance date, frequency of maintenance, Mean time between failure,, Mean time to repair, failure rate, critical component list, root causes, etc.
	Technical support	Customer complaints, customer profile, usage time, etc.
	Usage environment	Usage condition(e.g., average humidity, internal/external temperature), user mission profile, usage time, etc.
EOL to MOL	Recycling component	Reuse part or component, remanufacturing information, quality of remanufacturing part or component, etc.
EOL to BOL	EOL product status	Product/part/component lifetime, recycling rate of each component or part, etc.
	Dismantling	Ease to disassemble, reuse or recycling value, disassembly cost, remanufacturing cost, disposal cost, etc.
	Environment effects	Material recycle rate, environment hazard information, etc.

Appendix B

QLM messaging example

```
<qlm:qlmEnvelope
xmlns:qlm="QLM_mi.xsd"
xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
xsi:schemaLocation="QLM_mi.xsd QLM_mi.xsd"
version="0.2" ttl="-1">
<qlm:write msgformat="QLM_mf.xsd">
<qlm:msg>
<Objects xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="QLM_mf.xsd" xsi:schemaLocation="QLM_mf.xsd QLM_mf.xsd">
       <Object>
<id>ProductionLine_213</id>
       <InfoItem class="Production_Batch_ID">
              <value>PL_213_3.1.2013.12.20</value></InfoItem>
       <InfoItem class="Pallet_IN_ID">
              <value>SRIN_123_98</value></InfoItem>
       <InfoItem class="Pallet_OUT_ID">
              <value>SROUT_123_98</value></InfoItem>
       <Object>
<id>Oven</id>
       <InfoItem class="TemperatureIN">
              <value>29</value></InfoItem>
       <InfoItem class="TemperatureOUT">
              <value>890</value></InfoItem>
       </Object>
       <Object>
       <id>StampingMachine</id>
       <InfoItem class="PressForce">
              <value>2989</value></InfoItem>
       <InfoItem class="CoolingTemperature">
              <value>378</value></InfoItem>
       </Object>
       </Object>
</Objects>
</qlm:msg>
</qlm:write>
</qlm:qlmEnvelope>
```

Figure B.1: QLM message example for Application1 case example

Appendix C

oBIX payload

```
<!-- Example of a simple reply message using
oBIX payload, with namespace and schema
declaration. -->
<qlm:qlmEnvelope
xsi:schemaLocation="QLM_mi.xsd QLM_mi.xsd" version="0.2" ttl="10">
  <qlm:response>
    <qlm:result msgformat="obix.xsd">
     <qlm:return returnCode="200"/>
     <qlm:requestId>REQ0011212121212</qlm:requestId>
     <qlm:msg xsi:schemaLocation="http://obix.org/ns/schema/1.0obix.xsd">
       <obj href="http://myhome/thermostat">
       <real name="spaceTemp" unit="obix:units/fahrenheit"val="67.2"/>
       <real name="setpoint" unit="obix:units/fahrenheit"val="72.0"/>
         <bool name="furnaceOn" val="true"/>
       </obj>
    </qlm:msg>
    </qlm:result>
  </qlm:response>
</qlm:qlmEnvelope>
```

Figure C.1: QLM message example with oBIX as payload