



DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,  
SECOND CYCLE, 30 CREDITS  
*STOCKHOLM, SWEDEN 2017*

# **Web-based Software Reengineering**

A case study on next generation product-  
selection system

**WEIFENG LIN**

## **Abstract**

Legacy systems are usually expensive to maintain, and they quickly become outdated in a fast changing industry with new requirements and technologies. However, replacing an old system with a complete new one could prove both costly and time consuming, and therefore the method of reengineering could present itself as a beneficial alternative.

There is a lack of practice-based research in relation to the implementation of software reengineering. The main purpose of this thesis is therefore to recognize key aspects on how to reengineer web-based IT systems in a modern, easy-to-maintain and business-enhancing manner.

A case study has been carried out, a reengineering of a legacy system used for product selection at Grindex AB, a Swedish submersible pump supplier and manufacturer. The process includes three stages, firstly a thorough analysis of the legacy system has been carried out, secondly interviews were conducted in order to identify new requirements, and lastly a new system was developed by Struts 2, Spring MVC, Spring, Hibernate in Java and a relationship database of MySQL.

The author is presenting six areas of consideration – architecture, function, interface, language, data storage and algorithm – in relation to the software reengineering life cycle, and with a comparison between the legacy system and the reengineered system.

## **Nyckelord**

Reengineering, web-based system, legacy system, product selection

## **Abstract**

Legacy system är ofta kostsamma att underhålla och de blir fort förlegade i en industri i snabb förändring med nya krav och teknologier. Men det kan samtidigt visa sig kostsamt och tidskrävande att byta ut ett gammalt system mot ett helt nytt. Mot denna bakgrund skulle metoden reengineering kunna utgöra ett fördelaktigt alternativ.

Den praktiskt orienterade forskning i relation till programvaruutveckling genom reengineering är knapphändig. Det främsta målet med denna avhandling är därför att ringa in viktiga aspekter sett till hur webbaserade IT-system skulle kunna reengineras på ett modernt sätt, enkla att underhålla och företagsfrämjande.

En studie har genomförts, en reengineering av ett legacysystem som använts för produktval på Gridex AB, en svensk leverantör och tillverkare av dränkbara pumpar. Processen omfattar tre delar. Först genomfördes en grundlig analys av legacy systemet, sedan genomfördes intervjuer för att identifiera nya krav och avslutningsvis utvecklades ett nytt system med Struts 2, Spring MVC, Spring, Hibernate i Java och en relationsdatabas i MySQL.

Författaren presenterar sex områden att beakta – arkitektur, funktion, gränssnitt, språk, datalagring and algoritm – i relation till programvarureengineeringens livscykel och med en jämförelse mellan legacysystemet och det reengineerade systemet.

## **Nyckelord**

Reengineering, webbaserade system, legacysystem, produktval

# Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>1</b>
1.1	Background .....	1
1.2	Problem.....	2
1.3	Purpose .....	2
1.4	Goal.....	2
1.5	Methods.....	2
1.6	Ethics.....	3
1.7	Delimitations .....	3
1.8	Disposition.....	3
<b>2</b>	<b>Theoretic Background .....</b>	<b>4</b>
2.1	Software Reengineering.....	4
2.1.1	Reverse Engineering.....	5
2.1.2	Forward Engineering.....	6
2.1.3	Restructuring .....	7
2.2	Development of Web-based System .....	8
2.2.1	Structure .....	8
2.2.1.1	3-tier Architecture .....	8
2.2.1.2	MVC Design Pattern .....	8
2.2.2	Java Framework.....	9
2.2.2.1	Integrated S2H Framework .....	9
2.2.2.2	Integrated SSH Framework.....	11
2.2.3	Programming Language .....	14
2.2.4	Data Persistence .....	15
<b>3</b>	<b>Methodology .....</b>	<b>16</b>
3.1	Qualitative Research Methods.....	16
3.2	Inductive Research Approach.....	16
3.3	Research Methodology.....	16
3.4	Grindex AB .....	17
<b>4</b>	<b>System Reengineering Development.....</b>	<b>18</b>
4.1	Reverse Engineering .....	18
4.1.1	Characteristics of Front-end .....	18
4.1.2	Characteristics of Back-end.....	21
4.1.3	Common Characteristics .....	22
4.2	Forward Engineering.....	23
4.2.1	Requirement Analysis .....	23
4.2.1.1	Competition Analysis.....	23
4.2.1.2	Semi-conduct Interview .....	24
4.2.2	Front-end Design.....	26
4.2.3	Back-end Design .....	28
4.2.4	Tools .....	31
4.2.4.1	Language .....	31
4.2.4.2	Database Design.....	31
4.2.5	Algorithm.....	31
<b>5</b>	<b>Results.....</b>	<b>32</b>
5.1	Main aspects to consider .....	32

5.1.1	Architecture.....	32
5.1.2	Function .....	33
5.1.3	Interface .....	34
5.1.4	Language.....	34
5.1.5	Data Storage.....	35
5.1.6	Core Algorithm .....	35
<b>5.2</b>	<b>Front-end.....</b>	<b>36</b>
5.2.1	Search Field.....	37
5.2.2	Result Field .....	38
<b>5.3</b>	<b>Back-end.....</b>	<b>39</b>
5.3.1	User Management .....	39
5.3.2	Pump Management.....	41
<b>6</b>	<b>Analysis and Evaluation.....</b>	<b>44</b>
<b>6.1</b>	<b>Comparison between legacy system and new system.....</b>	<b>44</b>
6.1.1	Advantages of reengineering and new system .....	44
6.1.2	Disadvantages of reengineering and new system.....	44
<b>6.2</b>	<b>Evaluation of reengineering and new system.....</b>	<b>46</b>
6.2.1	Evaluation of user experience and functionality.....	46
6.2.2	Evaluation of technical performance .....	47
<b>7</b>	<b>Conclusion and Future Work.....</b>	<b>48</b>
	<b>References.....</b>	<b>49</b>

# 1 Introduction

## 1.1 Background

Software systems are continually evolving in order to meet ever-changing requirements. Along with the fast development of information technology, more and more tools and methods are coming up to build quicker, more secure and easier maintenance software system. Especially nowadays, the Internet era, web application is rapid and progressive diffusion in the modern society and the complexity of functions and structure is increased swiftly. A system which is longer than five years could be categorized as legacy system [1]. Canfora and Cimitile [2] proposed four strategies to deal with legacy systems, which are

- (i) to change the system entirely
- (ii) to stay at the current state and not change any longer
- (iii) to maintain with controlled cost
- (iv) to upgrade the system in a modern fashion

In order to meet new and complex requirements internally and externally, for instance new functions, user-friendly interface, robust system architecture and so forth, most companies can't stay at the current state, and instead they have to evolve their systems correspondingly. However, it usually costs highly to build a completely new system, which may take thorough analysis and a longer decision making period, and the high cost may affect the decision to cancel the rebuild project. In addition, software engineering involves not only the development and deployment of software, but software maintenance is also an important lifecycle activity, once the software is deployed, software must be maintained. In some legacy software, the maintain cost is very high due to various requirements.

Therefore, to upgrade system, especially web-based application, becomes unavoidable issue for most companies, but shorter development time and budget constraints are influencing the process of software design. As an effective method, reengineering can usually achieve the goal in a best way. To generate evolvable system based on an engineering process is the aim of software reengineering, defined by Seacord [3]. In addition to cost, software reengineering could enhance other factors at the same time such as complexity, maintainability, etc.

In the process of software reengineering, the system is disassembled and analyzed in detail and man-made knowledge or internal relationships could be extracted, and the purpose of reengineering is to create new system based on the existing materials together with new requirements. Although each company has its own business functions and processes, the reengineering effort has to customize to its individual needs and customers, however some key aspects and basic guideline when implementing reengineering can be applied in order to effectively carry on the process.

## **1.2 Problem**

The existing systems deployed in many companies were usually built on several years ago, therefore technically they are expensive to maintain, develop and run based on technology and knowledge available at the time, and meanwhile, functionally it no longer meets all the requirements from internal employees and external customers.

Current researches focus on different kinds of “-based” web application by reengineering, such as role-based [4], clone pattern-based [5], model-driven [6], etc. However, there is a small amount of literature available to guide on practical solutions when developing web application in reengineering fashion. Therefore, the problem is that it is lack of research on the main aspects to consider when implementing reengineering in practice.

## **1.3 Purpose**

The purpose is to research how to develop web-based software in a modern, easy-to-maintain and business-enhancing manner so as to support evolving business operations and requirements by reengineering. The research tends to highlight the “how” of reengineering instead of focusing on the “why” and the “who”.

## **1.4 Goal**

The goal is to find out the main aspects to consider during reengineering from a legacy system to a state of the art and reliable system. In order to reach the goal, a reengineering process is to be conducted based on a legacy product selection system at Grindex AB. To reengineer it, the legacy system needs to be analysed thoroughly and then define the new requirements, and finally develop the new system.

## **1.5 Methods**

Based on Håkansson’s [7] research methods and methodologies, this thesis is a qualitative research, and it adopts the following spectrum, namely interpretivism as philosophical assumption, applied research as research method, inductive approach as research approach. The methodologies used in this research includes: literature review, case study, interviews, and prototype design.

Specifically, this research is using development in an interpretative way to create artifacts instead of using experiments or big data sets to verify a hypothesis. The research therefore is a qualitative research in general. By exploring width, depth and complexity, interpretive assumption is to get context and phenomena and “works well in developing computer systems and

artifacts” [7]. This research will examine a set of circumstances and deliver a particular application, thus applied research method is suitable. Inductive approach is a reasoning manner to formulate propositions from particular facts, and it can be “used in the development of an artifact” [7]. The level of research methods is higher than research strategies and designs, a.k.a. methodology, based on the above research methods, this search mainly uses case study as the methodology during the entire research activity, and the empirical study is carried on at Grindex AB.

## **1.6 Ethics**

The new system is created based on the legacy system, it doesn’t involve in any patent issue. And it is independent from Grindex’s core business system, so if the project fails it won’t affect the ordinary business. Moreover, Grindex AB is a Swedish company with over 75 years history, it keep sustainable and ethics principle to make business. In addition, the group company Xylem Inc., contributes very much on sustainable development, and Ethics and Compliance Review Boards are installed in each of the regions.

## **1.7 Delimitations**

Due to limited time this project implements part of the legacy system, namely main framework and main functions, some independent functions will be developed in the future, for instance curve plotting. Based on other systems, budget and resources, the developing tools are defined by the company, such as programming language, database type, etc. The research focuses on simple-functioned web-based software and the overall requirements from the company are not too complex. By reengineering, the legacy system should be existed already, and it differs from developing a new system from scratch.

## **1.8 Disposition**

The thesis is organized as follows: Chapter 2 describes the theoretical background of software reengineering and development of web-based system. Chapter 3 analyzes the research methodology in detail. In Chapter 4, the procedure of system reengineering is presented in detail, which is reverse engineering and forward engineering. Then the results are displayed in Chapter 5, the analysis and evaluation is discussed in Chapter 6. Finally the conclusion and future work is proposed in Chapter 7.



## 2 Theoretic Background

### 2.1 Software Reengineering

Chikofsky and Cross [8] define reengineering as “the examination and alteration of a subject system to reconstitute it in a new form and the subsequent implementation of the new form.” Reengineering is to examine the finished product and build it again, which can add new functionality or develop a completely new system.

When it comes to the process of reengineering, Chikofsky and Cross [8] propose that reengineering includes “some form of reverse engineering followed by some form of forward engineering or restructuring”, the model based on this concept is shown in figure 2-1. Specifically, useful information is firstly to be extracted from the existing system, such as documentation, requirements, code, etc., from detailed functions to abstract models. And then new requirements are to be merged into the abstracted models, in another word the abstracted models are altered to new models. Based on the new model, detailed specifications and architecture of the system is to be designed. Finally an up-to-date system is to be developed and deployed. Wagner [6] clarifies the process of reengineering into three steps, namely reverse engineering, transformation and forward engineering.

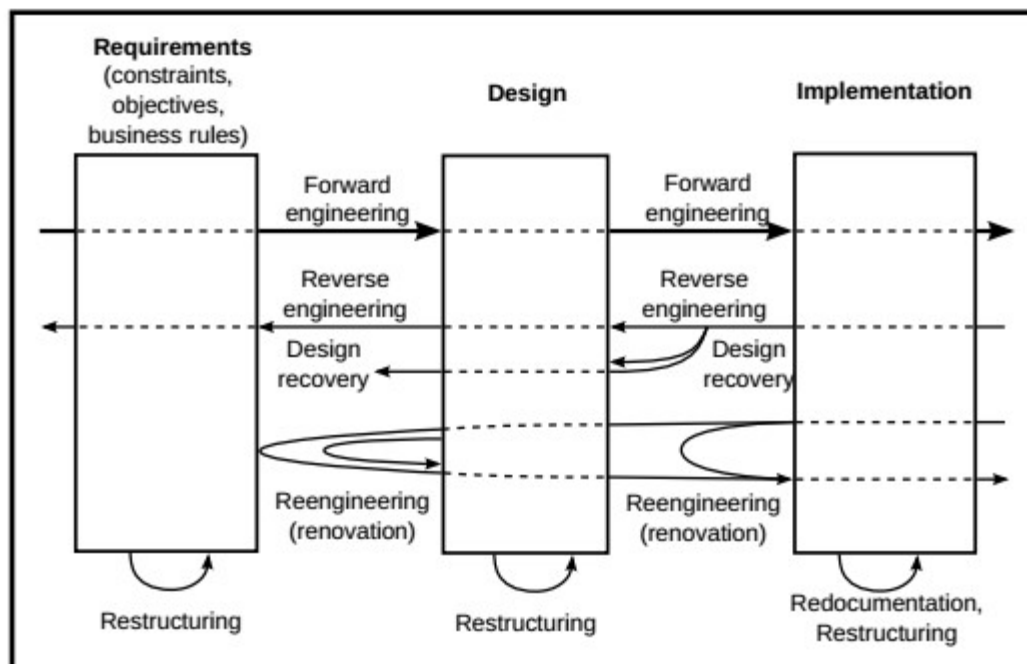


Figure 2-1: Conceptual model of software reengineering, taken from [8]

In figure 2-1, the reengineering model is according to the phases of the software development process, namely requirements, design and implementation. It means both reverse engineering and forward engineering experience the three stages, but in different ways. In the beginning, reverse engineering starts with analyzing existing system which has been implemented, to identify components and their relationships from the system, which is a backward way to understand the system, from a coded set to a model. Based on the extracted model, transformation will be carried out where new constraints, objectives and business rules will be analyzed and added so that the specifications of existing system will be altered. By following the new specifications, a forward way to design new system is carried out, and it is a restructuring from the old system. Finally implementation will be conducted; it includes coding, testing and delivery. For certain parts, restructuring could be applied if the functions will be not changed much, especially for the codes of algorithm. Forward engineering conducts on system level, it involves the whole life cycle of software development, while restructuring happens in certain phases.

By delivering improved understanding of software to customers, the life time of the software will be prolonged. Specifically, by extracting internal structure and business process, and introducing different technologies, the new system generates clear and appropriate architecture, such as logic, persistence, user interfaces, etc., and meanwhile the new system would have less dependency from the developers.

Software maintenance and software evolution are two main aspects to apply reengineering, but scope, cost and technology are different. Generally software evolution is more expensive because it requires understanding the whole system, instead of certain specific issues. Moreover, artifacts are produced during the reengineering, instead of pure program code.

### **2.1.1 Reverse Engineering**

Reverse engineering comes originally from the analysis of hardware [8], where the finished product is examined in order to understand the system and its structure. The purpose is usually either to improve own product or scrutinize competitor's product. When it comes to software, the first purpose has been mainly used for maintenance process where the maintainer could get clearly understanding of the system created by developers, whereas the second purpose could be used for clone of the original product.

Reverse engineering is finding out how a product works from the existing product, which includes structures, states, relationships among components and so on. This extraction is a higher level abstraction which may lead to modeling [6]. Chikofsky and Cross [8] concludes reverse engineering into two aspects:

1. “Identify the system’s components and their interrelationships”
2. “Create representations of the system in another form or at a higher level of abstraction”

Reverse engineering is different from reengineering, the confusion over terminology. According to Chikofsky and Cross [8], reverse engineering is part of reengineering process. Moreover, another two terms are used in the phase of reverse engineering, namely redocumentation and design recovery, which are explained as below.

- Redocumentation

Redocumentation is a simple form of reverse engineering. It is to create or revise representation at similar level of abstraction from existing system. The outcome of the redocumentation is for a better visualization for audience to understand the current representation from another angel, for instance by analyzing program code, certain data flow or data structure could be illustrated.

- Design recovery

Design recovery is a subset of reverse engineering [8]. By doing so, audience could completely comprehend the system, such as purpose, mechanisms, functions and so on. Design recovery recreates a broader collection of information, and “recreates design abstractions from a combination of code, existing design documentation (if available), personal experience, and general knowledge about problem and application domains” [24].

### **2.1.2 Forward Engineering**

Forward engineering is a development process from a high-level abstractions or concepts to build in complex and lower-level details, for example from design specifications to programming classes. In Chikofsky and Cross’s [8] model, figure 2-1, the process of requirements, design and implementation is the standard forward engineering. Moreover, Laplante [25] defines software engineering as “a systematic approach to the analysis, design, assessment, implementation, test, maintenance and reengineering of software, that is, the application of engineering to software.”

Software development process is a familiar term, which is similar as forward engineering. The core activities include requirements, design, construction, testing, debugging, deployment and maintenance. The systems development life cycle is to describe a process for planning, creating, testing, and deploying an information system in a forward way.

The term of “forward engineering” is as opposed to “reverse engineering”. Forward engineering is generally considered as a normal process for IT development, whereas reverse engineering is reflected as a creative deconstruction.

### **2.1.3 Restructuring**

Restructuring is “the transformation from one form of representation to another without changing the functionality” [8]. Restructuring could reduce complexity and improve structure, which applied for code, model, design plan, requirement structure and so on.

Under the context of requirements, restructuring plays an important role to transform from old specification to new one, where new design, structure, etc. could be added into the new specification. The restructuring is a local alteration, which supports to adjust to new environmental constraints but don't involve other parts of development.

Under the context of computer code, restructuring is similar as code refactoring which doesn't change external behavior of code, but enhance code readability. Moreover, the restructure of code doesn't require understanding the meaning, such as convert “if ... then...” statements into “case” structure, where the duplicated source code is removed semantically.

## 2.2 Development of Web-based System

### 2.2.1 Structure

3-tier Architecture is an architectural philosophy and a layered software framework which can be used in any project, whereas MVC is a design pattern which is selected according to the requirements of the project. Other design patterns are for instance Factory Pattern, Façade Pattern and so forth [10].

#### 2.2.1.1 3-tier Architecture

Based on the software design philosophy of low coupling and high cohesion, 3-tier architecture divides the system into three layers, namely User Interface Layer (UIL), Business Logic Layer (BLL) and Data Access Layer (DAL).

- User Interface Layer (**UIL**) is an interface for the user who can browse information and input data.
- Business Logic Layer (**BLL**), a.k.a. Domain Logic Layer, makes logic decisions and performs calculations. BLL is the middle layer between UIL and DAL, therefore BLL coordinates application and process commands. For example, BLL can determine how data is generated, shown, saved and altered.
- Data Access Layer (**DAL**) accesses to data directly and operates persistent storage such as relational database. Because of decoupled from BLL, the logical data model can be modified without impacting the business layer.

#### 2.2.1.2 MVC Design Pattern

MVC is the abbreviation of Model-View-Controller, and it's a software design pattern. MVC was first described in 1979 by Trygve Reenskaug, then working on Smalltalk at Xerox PARC in 1980s. By separating code into data, display and business logic, MVC organizes the code in the three parts so as to enhance the flexibility of the system. For example, while modifying the display interface, the business logic doesn't need to rewrite. MVC has been developed for mapping functions of input, processing and output in a logical structure and a graphical user interface.

- **Model** is data model and represents application's data. The technology is such as JavaBean.
- **View** is responsible to display the data model. The technology is such as Java Server Pages (JSP).
- **Controller** decides which view should render to the user based on the user's request and business logic.

Controller is in the middle between View and Model, the controller receives a message from the view and forwards to the model, and returns the processed data to the view. The technology is such as Servlet.

## **2.2.2 Java Framework**

### **2.2.2.1 Integrated S2H Framework**

By integrating Struts 2 and Hibernate, a lightweight framework is formed to enhance the expansibility and maintainability for an application. The Struts 2 is used for display and business logic control, and Hibernate is used for data persistence.

#### **❖ Struts 2**

Apache Struts is “a free, open-source, MVC framework for creating elegant, modern Java web applications.” Struts 1 was started in 2000 and the last version was released in 2008, in 2013 the Apache Struts Project officially announce the Struts 1 reached its end of life. Struts 2 was started in 2006 based on the technologies of Struts and WebWork, and since 2008 Struts 2 became focused framework to push forward instead of Struts 1. Struts 2 shares the same basic principles with Struts 1, but improves architecture greatly, which is highly decoupled, enrich features and APIs, as well as better maintenance.

The architecture of Struts 2 is designed based on MVC pattern. The FilterDispatcher in Struts 2 links to Controller in MVC, the Action links to Model, the Result links to View. The main work process is illustrated in figure 2-2, and specifically it goes the following six steps:

1. User sends a request – HttpServletRequest
2. The request is passed by a serial of Filters
3. The request invokes FilterDispatcher, which is the core of Controller. If there is any action, the request is passed to ActionProxy.
4. Corresponding Action Class is invoked according to the struts.xml, which is a configuration file invoked by ActionProxy. The Action Class is the business logic and connected to persistence.
5. After the Action is executed, a result will be returned, which is usually linked to a JSP page defined in struts.xml.
6. Finally, HttpServletResponse return the result and display to the user according to the configuration in web.xml.

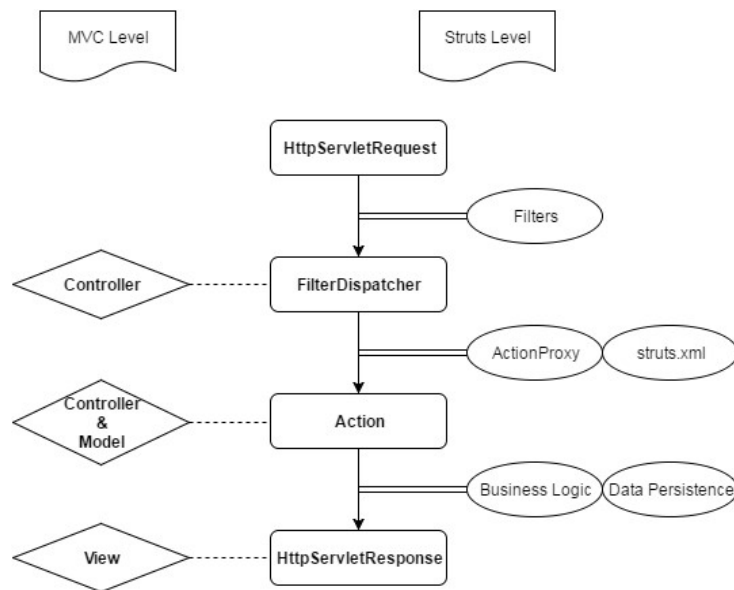


Figure 2-2: Architecture of Struts 2 framework, taken from [15]

### ❖ **Hibernate**

Hibernate is an object-relational mapping (ORM) framework which solves the matching problem between relational database and Java objects [15]. Meanwhile, Hibernate simplifies the usage for developers to access to data persistence through JDBC package especially when the data outlives process in an application.

The configuration and mapping is conducted by xml files, usually `hibernate.cfg.xml` or `hibernate.properties`. In addition, Hibernate Query Language (HQL) is Hibernate own object-oriented query language, it supports the most popular databases [15]. The architecture of Hibernate framework is shown on figure 2-3.

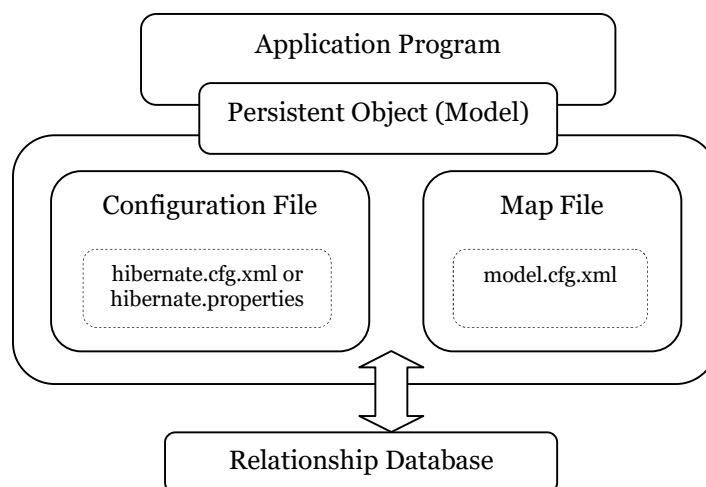


Figure 2-3: Architecture of Hibernate framework, taken from [15]

### 2.2.2.2 Integrated SSH Framework

By integrating Spring MVC, Spring and Hibernate, a powerful framework is formed to highly decouple a complex system and meanwhile keep its efficiency. Spring MVC is mainly used to process web request, Spring is in charge of the business logic, and Hibernate is used for data persistence. Hibernate is invoked by DAO interface which is managed by Spring [16]. The relationship in MVC level and application level are analyzed in table 2-1.

Table 2-1: Integration Layer and Development Step, taken from [16]

MVC Level	App Level	Analysis
Persistence Layer	Database	Link Hibernate & database
Model Layer	DAO Layer	Interact with database Operate CRUD on data
	Service Layer	Integrate business logic Implement business
Controller Layer	Action Layer	Process JSP requests
View Layer	JSP Layer	Display results in the page

#### ❖ *Spring MVC*

Similar as Struts 2, Spring MVC includes the concept of the Dispatcher servlet that interacts with the HTTP requests and delegates to the controller, view (and view resolver), and handlers. Figure 2-3 shows a diagram of Spring's implementation of the MVC pattern.

MVC pattern is obvious in Spring MVC framework, namely a Model, a View and a Handler. The handler processes the request, invokes the model and returns to the corresponding view [11]. Spring MVC separates the roles of the controller, model object, dispatcher Servlet and the handler object. Clear separation of objects and controllers makes them easier to customize.

- The front controller takes care of dispatching incoming requests to the correct handler and prepares the response, so that it can be rendered into something that the user would like to see.
- DispatcherServlet acts as the role of front controller in Spring MVC, and several components are used to fulfill its role by this servlet, furthermore the interface is the way to implement the components [11]. Besides of DispatcherServlet, there are more main components are for instance ViewResolver, HandlerMapping and so forth [12].



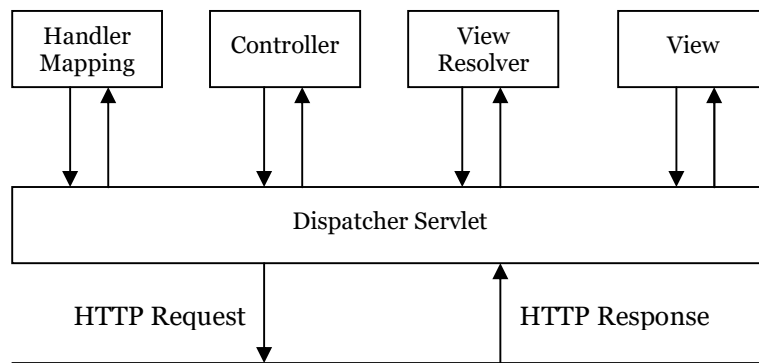


Figure 2-4: Diagram of Spring's MVC implementation, taken from [11]

### ❖ *Spring*

Spring is lightweight business logic layer framework, which can replace EJB technology [15]. Spring introduces many good features such as dependency injection, aspect-oriented programming (AOP), plain-old-Javaobjects (POJOs) [11], and RESTful web service framework and so on. Moreover, Spring is modular-based, developers can use the parts they need instead of the whole [13], for example Spring can integrate with Struts 2 only, or with Hibernate only, or with both at the same time.

The Spring framework is mainly consisted of 20 modules which are grouped into 8 sections, namely Core Container, Data Access/Integration, Web, Aspect Oriented Programming (AOP), Instrumentation, Messaging, and Test. The framework is shown in figure 2-5.

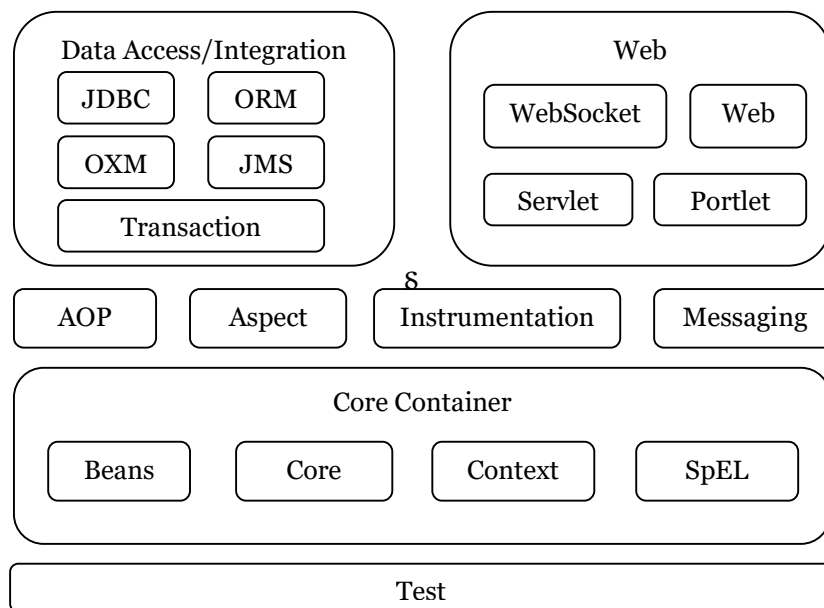


Figure 2-5: Overview of the Spring Framework, taken from [13]

In the Core Container, modules of spring-core and spring-beans provide the fundamental parts of the framework, such as features of Dependency Injection and Inversion of Control.

- Dependency Injection and Inversion of Control

Java platform provides flexible functions, and people can compose classes by various design patterns such as Factory, Builder, Decorator, etc., however, developers have to implement the fundamental components into a coherent whole by themselves. The Inversion of Control (IoC) in Spring framework fix this shortage efficiently by organizing different blocks into one application, which is robust and maintainable [13].

- Aspect-oriented programming (AOP)

In the module of spring-aop, it provides aspect-oriented programming implementation. “The goal of aspect-oriented programming is to design and develop programs by, first, separation of concerns in terms of aspects, and second, constructing the resulting applications by weaving the aspects to the basic modules that implement the skeleton of the application.” The business logic of the application can be implemented as modular concern. One basic type of AOP is called cross-cutting concern, which needs injecting several program fragments to the modules [14].

Other modules are explained briefly. Spring-aspects module is different from AOP function, this module is responsible for integration with AspectJ. Class instrumentation support and classloader are implemented by spring-instrument module. Spring-message is a new function in Spring framework 4 with key abstraction from the Spring Integration project. Moreover, annotations is supported by this module to map message to methods. The Data Access/Integration section is responsible to communicate with data persistence layer, and it consists of the JDBC, ORM, OXM, JMS, and Transaction modules.

The four modules in the Web section, namely spring-web, spring-webmvc, spring-websocket and spring-webmvc-portlet, provide web-oriented services for MVC implementation. The spring-test module provides test functions and tools, such as JUnit, TestNG, etc.

### 2.2.3 Programming Language

Both PHP and Java are widely accepted and used for web development. Some developers will use both codes, depending on what the project call for, while others have their own preference. Java is often known as the industry standard and a bit more robust, while PHP is often known as the server side scripting language.

Integration is the strength of Java. Because Java is almost “Industry Standard”, and meanwhile there are many standards implementations in Java. However, the selection of libraries is quite limited for PHP.

#### ❖ *Java*

According to the official definition by Java, Java is “the foundation for virtually every type of networked application and is the global standard for developing and delivering embedded and mobile applications, games, Web-based content, and enterprise software.” Moreover, 97% of Enterprise Desktops Run Java and 89% of Desktops or Computers in the U.S. Run Java [17].

In many industries such as banking, insurance, retail, telecom, etc., Java Enterprise Edition (Java EE) has been the platform of choice, because Java is an object-oriented language, and Java platform is robust and highly scalable, as well as suitable for distributed application. [Pro Java™ EE Spring Patterns, chap1 p1] Moreover, compiled Java code has great cross-platform advantage, because Java interpreters and runtime environments are available in most operating systems [18].

Hence, selecting appropriate technology is critical when developing Java EE-based software. These choices, backed by sound architectural and design principles, go a long way in building applications that are easy to maintain, reuse, and extend [19].

#### ❖ *PHP*

According to the definition by PHP, PHP is “PHP is a popular general-purpose scripting language that is especially suited to web development” [20]. As long as existing PHP processor in a web server, PHP can be interpreted, therefore most operating system can run PHP [18].

#### **2.2.4 Data Persistence**

Both XML documents and relational databases store and extract data. However, relational databases are better for dealing with big amount of data within a system. There are established systems to maintain large volumes of structured data efficiently. While all the information in an XML document can be transmitted from one party to another, therefor XML documents are better for self-describing.

##### **❖ XML**

XML, the abbreviation for Extensible Markup Language, is “a markup language for documents containing structured information. A markup language is a mechanism to identify structures in a document. Structured information contains content (words, pictures, etc.) and some indication of what role that content plays.” [21] Therefore, by adding markup in XML, a document can be defined in a structured way. The data and the structure are presented together, because the data in an XML file can be organized into hierarchies, the relationships between data elements thus are visually obvious.

##### **❖ Relational Database**

A relational database is a collection of data items organized as a set of formally-described tables from which data can be accessed or reassembled in many different ways without having to reorganize the database tables. The relational database was invented by E. F. Codd at IBM in 1970 [22]. In a relational database, data is readable by executing SQL queries, and it can be extracted and presented in many ways.

### **3 Methodology**

Based on Håkansson's [7] research methods and methodologies, the research spectrum is a qualitative research as a whole, and philosophical assumption is interpretivism, research method is based on applied research, research approach is inductive approach. The research methodologies include literature review, case study, interviews, and prototype design.

#### **3.1 Qualitative Research Methods**

Quantitative methodology and qualitative methodology are two major methodologies by Håkansson [7]. Quantitative methodology is based on large data set or experiments to provide a phenomenon, whereas qualitative methodology is based on investigations or development to create theory, products and inventions. In this research, a system is firstly studied and then a new system is developed based on the study result, which is an interpretative analysis instead of experiments or big data sets, therefore the research is a qualitative methodology in general.

#### **3.2 Inductive Research Approach**

Inductive approach, deductive approach and abductive approach are three major approaches proposed by Håkansson [7]. Inductive approach is a reasoning manner to formulate propositions from particular facts, and it could be used in the development of an artifact. Deductive approach is to verify or falsify hypotheses, and in most cases it's based on large data sets and used in quantitative method. Abductive approach combines inductive approach and deductive approach, so the conclusion is reasoned by both data and uses preconditions. In this research, a new system is developed based on observation and analysis of the phenomenon instead of data sets, so inductive research approach is applied.

#### **3.3 Research Methodology**

The research methodology is also called research strategy and design, and it is the guideline to carry on the research step by step. The methodologies used in this research includes: literature review, case study, interviews, and prototype design.

##### **❖ Literature Review**

Theory is an essential fundamental for a research. A literature review needs to carry out first to support the further analysis. In this research, the research focus on computer engineering and science, and includes key aspects for example, reengineering, web-based software development, programming, Java frameworks, data persistence, requirement analysis, and so on. Based on

the literature review, previous research results are studied and nowadays technologies are leant.

### ❖ *Case Study*

By an in-depth analysis, the case study is to investigate a phenomenon in a real life context. The research is carried on Grindex AB, and a case study is carried on to analyze Grindex pump selection system. The research is first to study the legacy system, and after the prototype of the new system is developed, a further comparison and discussion is conducted for this case study.

### ❖ *Interview*

A semi-structure interview could provide some guidance in the interviews but still allow interviewees for flexibility. The interview questions could guide the interview and collect the data and meanwhile the interviewee could freely bring in more information related to the interview topic. In this research, first hand data was gathered by semi-structure interviews made with Technical Support team and Marketing team at Grindex AB.

### ❖ *Prototype*

Prototype design is an early model or release of a product built to test a concept or process to be replicated [23]. In this research, a prototype of new pump selection system for Grindex AB is created, and it's a main framework with key functions. The prototype is based on the analysis results of reverse engineering of the legacy system and the interviews. Based on the prototype, the system will be improved and more functions will be added in the future.

## **3.4 Grindex AB**

The case study is carried out at Grindex AB in Sundbyberg, Sweden. Grindex is a fully owned subsidiary to Xylem Inc. which is a leading global water technology company. Grindex is specialized in designing and manufacturing electrical submersible pumps for professionals. There are over 40 types of pumps applied in various applications. Customers choose appropriate pumps according to the specific requirements, such as static head, dynamic head, flow speed, friction loss, pipe diameter, solid content, etc. Therefore, a reliable software system is important to accurately and quickly select product. At Grindex a web-based product-selection system has been run for 8 years. According to Wagner [6], Grindex adopts the second strategy, namely to freeze the current state without further changes in the past few years.

## 4 System Reengineering Development

### 4.1 Reverse Engineering

Usually the only documentation available for reverse engineering is source code, and by reading the code, document and specification are created. This process is called reverse engineering. In this research, the author read through most codes and analyzes the legacy system to identify current components and their relationships. The system is divided into front-end and back-end, and for each part, five aspects are examined, namely architecture, functions, interface, tools and algorithm. This chapter will analyze the characteristics of the legacy system from the above five aspects for both front-end and back-end, as well as common characteristics.

#### 4.1.1 Characteristics of Front-end

##### ❖ *Architecture*

The architecture of the front-end is 3-tier architecture, namely user interface layer, business logic layer and data access layer. Moreover, client-server model is the design pattern. Figure 4-1 illustrates the architecture in general and their internal relationship. The process in practice is when a client accesses pump-selection service with a web browser, the client initiates a request to Grindex's web server. The server then provides the corresponding service to the client who initiates the request.

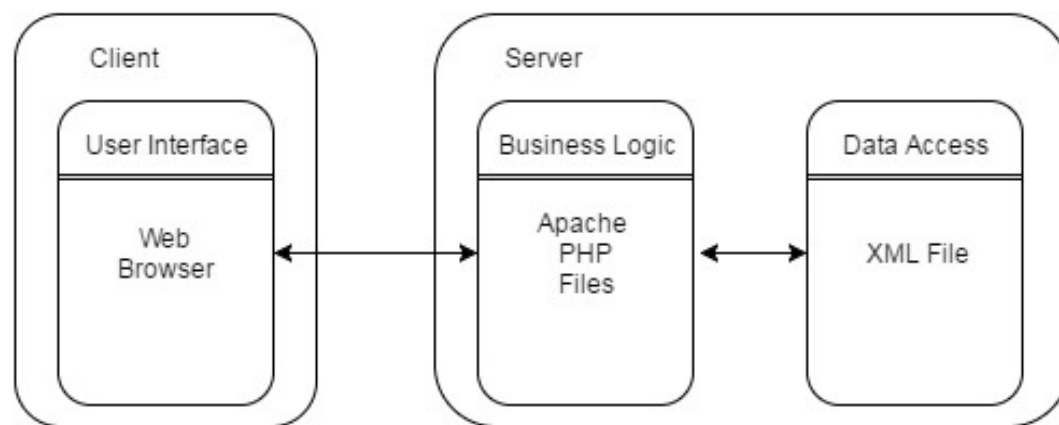


Figure 4-1: Architecture of the legacy system, Front-end

## ❖ Functions

In the front-end, users only need to retrieve data, but doesn't need to operate data, so the functions are designed based on this principle. Four main actions are considered which are to enter parameter values, to view result list, to view detailed information and to download files, see figure 4-2.

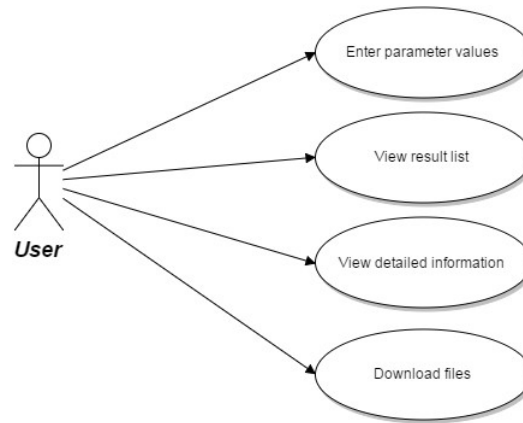


Figure 4-2: Use case of the legacy system, Front-end

Specifically, the sequence for the process is described in figure 4-3. User first inputs certain values from the interface and submits to server, the server calculates the results based on algorithms and fetch data from database, and finally returns and display the result to the client interface.

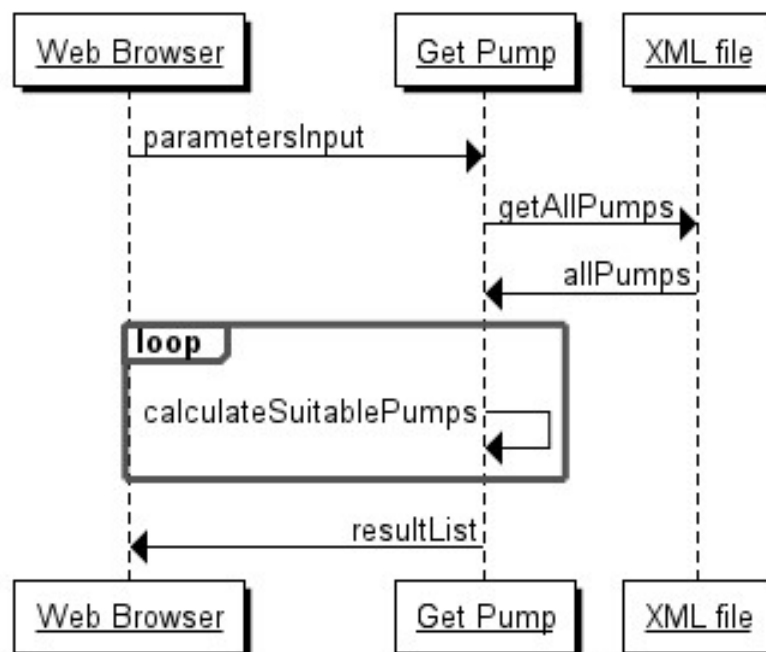


Figure 4-3: Pump selection sequence diagram of the legacy system, Front-end



## ❖ Interface

The front-end user interface of the legacy system is simple but well functioned. The pump selection interface is an html-based page which is embedded in the company's website by <iframe> tag. The page is a PHP page with CSS and JavaScript at client side, run by PHP at server side. In figure 4-4, the rectangle area with black dash line is the pump selection field. The area of green oval is the input field with different parameters. The two areas of orange ovals are the outputs with a list of possible pumps and a curve with one selected pump in the list.

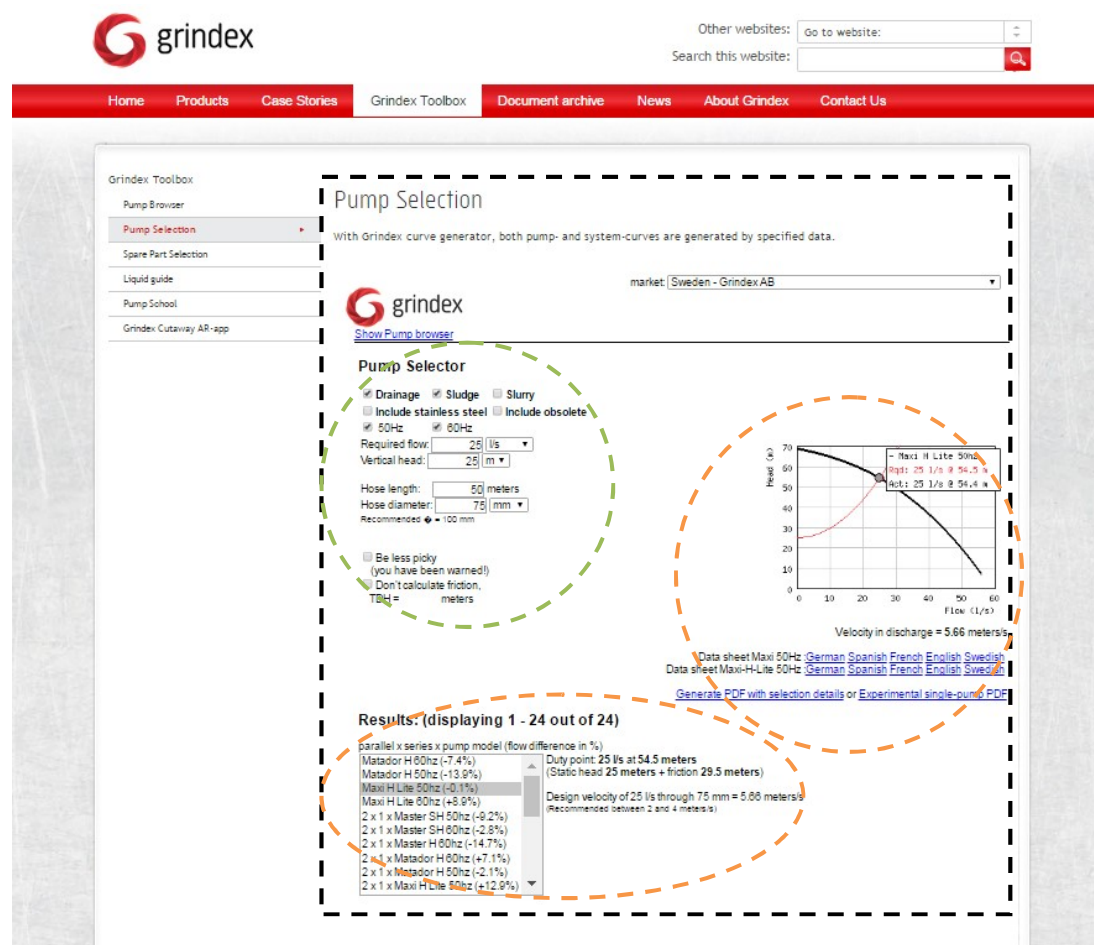


Figure 4-4: User interface of the legacy system, Front-end

### 4.1.2 Characteristics of Back-end

#### ❖ *Architecture*

The architecture of the back-end is similar as the front-end architecture, which is 3-tier architecture with client-server model, see figure 4-1.

#### ❖ *Functions*

The functions of the back-end are designed for administrator who needs to operate data in different ways, such as create, read, update and delete (CRUD). The use case is illustrated in figure 4-5.

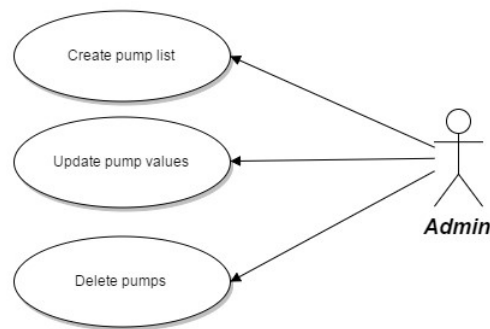


Figure 4-5: Use case of the legacy system, Back-end

#### ❖ *Interface*

The interface of the back-end is not easy to navigate and not user-friendly, moreover the process is complicated to follow. Due to sensitive information of the company, one example of the interface is shown in figure 4-6.

Available pump models in the curve data base:  
[Add new curve](#)  
[Generate pump data base file](#)  
[Change pump data base file in use](#)

I've got one of those PECU files with junk in the beginning and columns in the w

Pump model	options	modified (red means
Bravo-20-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20080725 - 16:30:37
Bravo-20-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20091204 - 15:40:25
Bravo-200-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20100607 - 17:10:46
Bravo-200-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20140217 - 15:10:40
Bravo-200-Ex-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20101209 - 08:33:33
Bravo-200-Ex-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20150921 - 12:37:32
Bravo-30-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20080725 - 16:30:37
Bravo-30-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20091204 - 15:43:18
Bravo-300-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20100607 - 17:11:55
Bravo-300-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20100607 - 17:12:10
Bravo-300-Ex-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20101209 - 08:34:05
Bravo-300-Ex-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20101209 - 08:34:23
Bravo-40-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20080725 - 16:30:38
Bravo-40-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20091207 - 11:26:58
Bravo-400-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20150921 - 12:32:49
Bravo-400-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20150921 - 12:33:16
Bravo-400-Ex-50hz	<a href="#">Edit</a> <a href="#">Delete</a>	20150921 - 13:18:31
Bravo-400-Ex-60hz	<a href="#">Edit</a> <a href="#">Delete</a>	20150921 - 13:19:03

Figure 4-6: Example of user interface of the legacy system, Back-end

### **4.1.3 Common Characteristics**

#### **❖ *Language***

The interfaces are programming by HTML, CSS and JavaScript. The server side functions are programming by PHP. There are around 50 files in PHP, and around 15 files in HTML, CSS and JavaScript. The total amount of codes is over 15,000 lines, and all files are saved in one same folder.

#### **❖ *Data storage***

The data points are used for pump selection, plotting corresponding curves, generating datasheet and so forth. The format of data storage is XML documents. All data for pump selection is stored in one XML file with many backup files along with time. In addition, each pump has several text documents (.txt) to record their detail information according to languages.

#### **❖ *Algorithm***

There are several algorithms to implement different tasks, such as pump selection algorithm, curve generation algorithm, etc. Due to confidential information, the algorithm is not explained in detail. The algorithm is coded in the program, but there is no corresponding document or flowchart.

## 4.2 Forward Engineering

After the reverse engineering for the legacy system, the stages of transformation analysis and forward engineering will be conducted in order to design a new specification for Grindex pump selection system. This section will explain the overview of the system in detail.

### 4.2.1 Requirement Analysis

One important way to transform from legacy system to new system is to conduct requirement analysis. By two main methods, the author collects the requirements so as to design the specifications. Based on the analysis, new constraints, objectives and business rules will be analyzed and combined with the legacy specifications.

#### 4.2.1.1 Competition Analysis

A brief competition analysis is conducted with regarding to target customers, functions, user interface and so on, shown in table 4-1 and table 4-2. The competitors are for instance xylect.com, grundfos.com and wilo.com. From the observation, many competitors' systems are suitable for experts with complicated parameters and functions. In addition, the user experience is various, some ones are fancy whereas others are poorly designed, but most of them are not easy to use as a new user. The format is mostly an internet-based software tool that allows users to select pumps without installation.

Table 4-1: Comparisons of different suppliers of pump selectors

Company	Format	Registration	Interface	Functions
Xylem	Online	No	Professional designed, but not easy to navigate	Full functions for expert
Wilo	Online	No	Different sections are separated clearly but too spare	Many functions for both normal user and expert
Grundfos	Online	No	Professional designed, and good UI	Full functions for expert
ABSEL (Sulzer)	Online	No	Similar as Wilo	Similar as Wilo
Pumpex (Sulzer)	Software on disc	-	Complicated to navigate	Full functions for expert
Franklin Electric	Software to download	Yes	Not accessible	Not accessible

Sulzer	Online	Yes	Not accessible	Not accessible
Tsurumi	Online	Yes	Not accessible	Not accessible
KSB	Online	Yes	Not accessible	Not accessible
Gorman-Rupp	Online	Yes	Not accessible	Not accessible

Table 4-2: Links of different suppliers of pump selectors

Company	Link for pump selector
Xylem	<a href="http://www.xylect.com">www.xylect.com</a>
Wilo	<a href="http://www.wilo.com/select">www.wilo.com/select</a>
Grundfos	<a href="http://product-selection.grundfos.com">http://product-selection.grundfos.com</a>
ABSEL (Sulzer)	<a href="https://absel.sulzer.com/ApplRange.aspx">https://absel.sulzer.com/ApplRange.aspx</a>
Tsurumi	<a href="http://www.tsurumi-global.com/products">www.tsurumi-global.com/products</a>
KSB	<a href="http://www.ksb.com/fluidfuture-en/selection">www.ksb.com/fluidfuture-en/selection</a>
Gorman-Rupp	<a href="http://www.grpumps.com/pumpfinder/index">www.grpumps.com/pumpfinder/index</a>
Sulzer	<a href="http://www.sulzer.com/en/Resources/Online-Tools">www.sulzer.com/en/Resources/Online-Tools</a>

#### 4.2.1.2 Semi-conduct Interview

Several semi-conduct interviews are conducted at Grindex Technical Support team and Marketing team. The interview questions and answers are shown in table 4-3. According to the interviews, the target users for Grindex pump selection system are mainly ordinary clients, distributors and internal technical support team, in another words, most of them are not experts. Therefore, the user interface in the front-end should be simple input parameters, and functions and logic should be easy to understand.

The interviewees are satisfied with the layout of the current front-end interface, so the layout and business logic keep the same in the new system. However, with regarding to the back-end, a new interface is highly required, because the current interface is difficult to maintain, whereas the basic functions of the back-end keep the same. Moreover, a database is required to secure the security and flexibility of the data.

Table 4-3: Interview questions and answers regarding requirements

Q1: Who are the users for the software?	Customers and distributors, if possible for expert with extra options.
Q2: Which situations/Why do you use the software?	I need to select possible pumps for the customers according to their requirements. In addition, for new pumps I need to add them into our system for selection.
Q3: How often do you use the software?	About 2-3 times per week
Q4: How do you feel about the design?	I feel the layout is fine, and I use to this style. But it doesn't look fresh and beautiful. It's not modern for our customers.
Q5: How do you feel the existing functions?	I satisfied with the selection functions (front-end) which are suitable for our team and our customers.
Q6: What design & functions do you want to add or delete?	When I want to check more details, some options for selection are expected to add, such as flow deviation, head deviation, number of parallel and serial pumps. The back-end is complicated to use, I'd like to have a new one which I could add and modify easily. The data storage is not safe and flexible, a database is required.
Q7: How about competitors' software?	We have different target customer groups, so their software is not suitable for us.
Q8: What pre-knowledge do customers have?	Ordinary customers don't have such professional knowledge, but they may know some concepts such as flow, head, hose length, diameter, and so on. If they don't know, we could help them to estimate.

### 4.2.2 Front-end Design

#### ❖ *Architecture*

MVC is the separation of model, view and controller, and it's simply a paradigm. Based on Java EE, the framework of the front-end is adopted by Struts 2 and Hibernate. The system is design as figure 4-7.

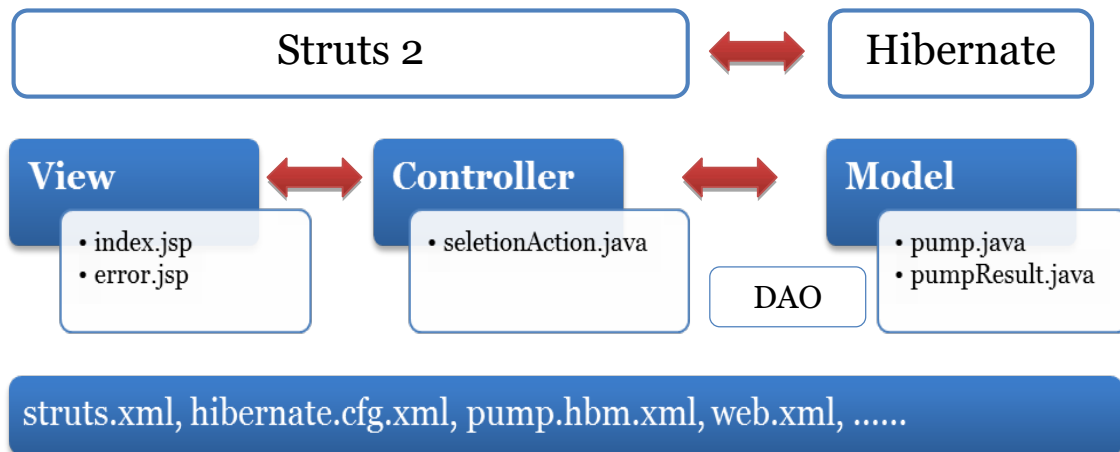


Figure 4-7: Architecture of the new system, Front-end

As illustrated in figure 4-7, the structure of the front-end is

View – Controller – DAO (Interface) – Model

Struts 2 is responsible for coordinating View and Controller, and Hibernate is mainly communicating with Model, the database.

## ❖ Functions

Based on the legacy system, the core functions and corresponding use case are similar, shown in figure 4-8. However, the pump selection sequence is different due to the new architecture, and the new sequence diagram is illustrated in figure 4-9.

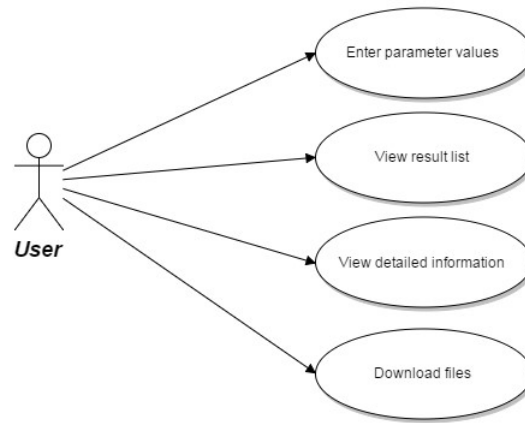


Figure 4-8: Use case of the new system, Front-end

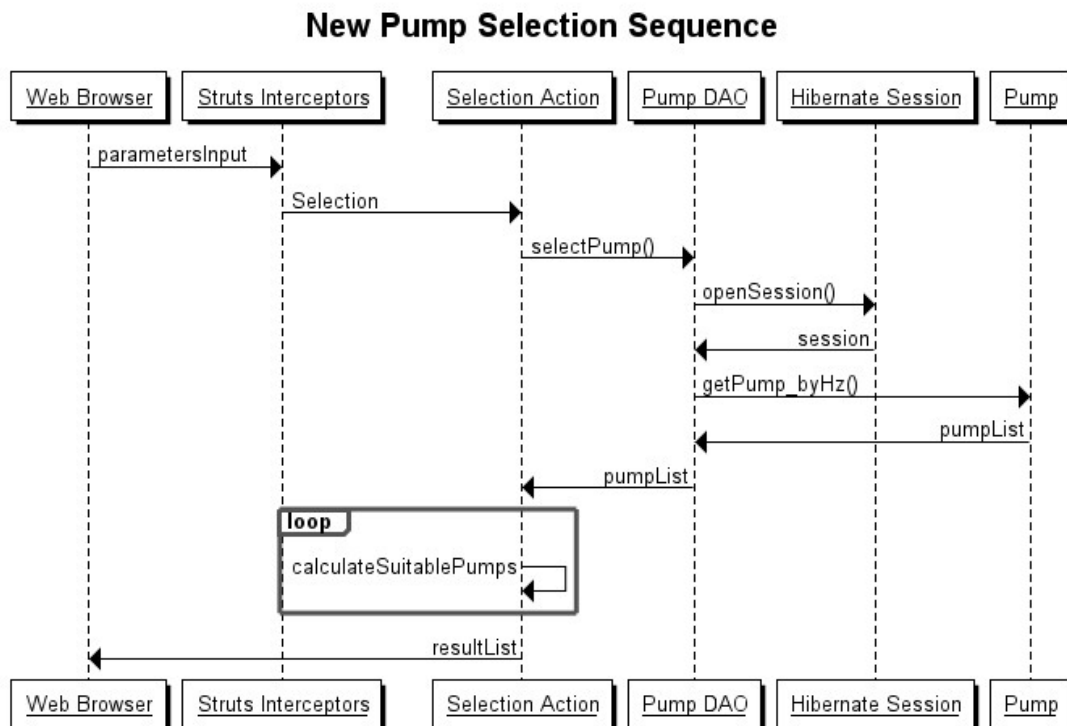


Figure 4-9: Pump selection sequence diagram of the new system, Front-end



### ❖ *Interface*

The layout of the interface is similar as the legacy system, but the user experience is improved very much, most elements, such as checkbox, text field, table, etc., are changed by new technology. Moreover, new features are added such as basic search and advanced search, sortable table, etc. Technically, the webpage is designed by Bootstrap (HTML, CSS and JavaScript). The results are displayed in Chapter 5.

#### 4.2.3 Back-end Design

### ❖ *Architecture*

The functions of the back-end are more complex than the front-end, so the architecture is adopted by framework of Spring MVC, Spring and Hibernate.

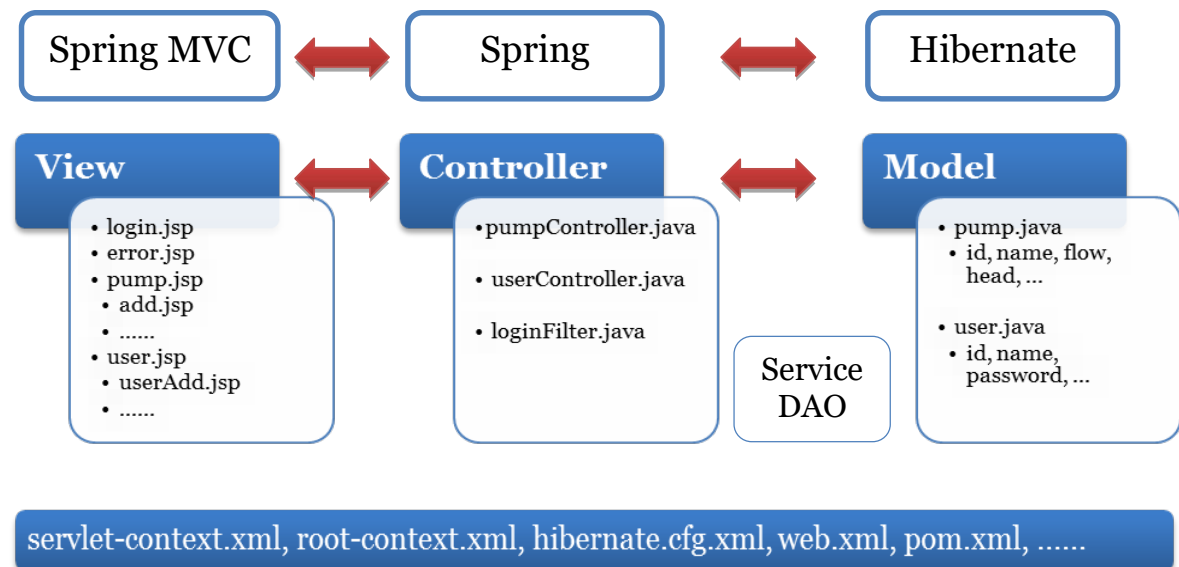


Figure 4-10: Architecture of the new system, Back-end

As illustrated in figure 4-10, the structure of the back-end is

View – Controller – Service (Interface) – DAO (Interface) – Model

It is together with configuration files such as hibernate.cfg.xml, web.xml, servlet-context.xml, root-context.xml, decorators.xml, pom.xml, etc. Moreover, it's applied RESTful style for the URL. Specifically, four main views are considered, namely pump management, user management, login page and error page. Two controllers work corresponding to pump and user, and through Service layer and DAO layer, models are connected to controllers. The procedure is explained as below and it takes “add pump” as an example.

## ❖ Functions

Although the basic functions such as add, delete, update and view are similar to the front-end, but the methods to implement are totally different, the new use case is shown in figure xxx. Two main modules are designed in the back-end are explained as below.

- User Management Module

For security reason, the system needs login function, moreover the administration can add and delete a user, and the user can update his or her information, such as password.

- Pump Management Module

All users have the same authorization to add, delete, update and view pump information from database. Input validation is applied to certain parameters.

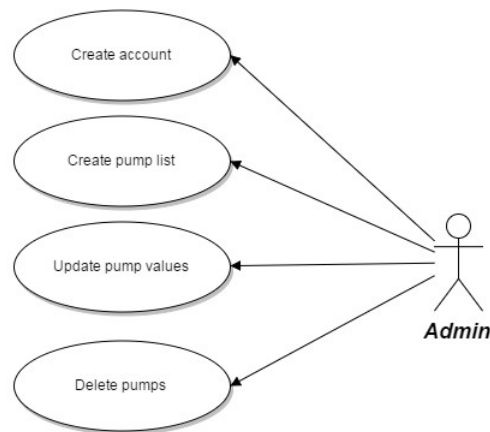


Figure 4-11: Use case of the new system, Back-end

However, the pump selection sequence is different due to the new architecture. The example of how to add a new pump is described in detail as below, the first step is to view the “add” page by GET request, and after type in all values it’s to submit the value and save into the database by POST request.

When a user clicks “add pump” link in order to view “add” page by GET request

1. A GET request arrives at Dispatcher Servlet in the servlet-context.xml, which defines this servlet's request-processing infrastructure.
2. Dispatcher Servlet finds a handler mapping file, which finds the current URL request to which controller to deal with.
3. Dispatcher Servlet sends the GET request to PumpController.
4. The PumpController then adds the attribute of new pump object and return add view

5. ViewResolver resolves the view selected for rendering by @PumpController.
6. The final view is returned to add.jsp resources in the /WEB-INF/views directory.

When a user click “add pump” button in order to add a new pump in the database by POST request

1. A POST request arrives at Dispatcher Servlet in the servlet-context.xml, which defines this servlet's request-processing infrastructure, together with certain values from “add” view page.
2. Dispatcher Servlet finds a handler mapping file, which finds the current URL request to which controller to deal with.
3. Dispatcher Servlet sends the POST request to PumpController.
4. The PumpController then interacts with Service layer, DAO layer and a series of operations to finally access pump model.
5. Through the pump model, results are saved into database and return list view.
6. ViewResolver resolves the view selected for rendering by @PumpController.
7. The final view is returned to list.jsp resources in the /WEB-INF/views directory.

### ❖ *Interface*

The legacy back-end is difficult to use and not user-friendly. Based on the feedback of the interview, the interface of the back-end needs to be re-designed, but the basic functions are similar as the legacy system. Therefore, a totally new interface is designed by Bootstrap (HTML, CSS and Javascript). The results are displayed in Chapter 5.

#### **4.2.4 Tools**

##### **4.2.4.1 Language**

Java is the main programming language in the upgraded system, JSP as web interface. In addition, Bootstrap is applied for webpage design together with CSS and JavaScript. Some plugins are applied such as Sitemesh, etc.

The general technologies are JSP as web interface, Java Enterprise Edition as platform, MySQL as database, and Glassfish as Server. Some plugins are applied such as Bootstrap, Sitemesh, etc.

##### **4.2.4.2 Database Design**

Since the existing data is storage in XML documents, and the new system is designed to store in a relational database, therefore MySQL is adopted, moreover, data transfer from XML to database is conducted. Two tables are designed for the system, namely user table and pump table.

- User Table

User table stores all information related to the users who can operate the pump table in the back-end. The user information is stored such as name, password, email, type, etc.

- Pump Table

Pump table is used for both back-end operation and front-end search. Therefore, every operation for the data from back-end will apply to the front-end when client search for certain pumps at the same time. The pump table stores all the information related to pumps, such as name, family, frequency, flow, etc.

#### **4.2.5 Algorithm**

The current algorithms can achieve the functions and they don't affect the goal of this project, therefore the algorithms are kept as most as possible from the legacy system. Moreover, the flowchart for implementing Linear Interpolation is drawn, but due to confidential reason, the chart and other algorithms are not explained in detail in this report.

## 5 Results

Reengineering is practical and important for nowadays company. According to the methodology of reengineering, namely requirement, design and implementation, the author first extracts the useful information from the existing system, such as documentation, code, etc., and then designs the new structure of the system based on the new requirements and the new technologies, and finally implements the restructuring to an up-to-date system. However, there is less research on practical aspects for web-based system reengineering, based on the case study, the author proposes two suggestions for future research, one is main aspects to consider when reverse engineering and design new system, the other one is an integrated framework for simple functioned enterprise web-based system.

In this chapter, the main aspects to consider in a reengineering project will be discussed, and moreover, the layout, functions and technologies of the new system will be explained in details according to front-end and back-end.

### 5.1 Main aspects to consider

By reengineering, the new system is developed quickly because many aspects can be reused, such as function logics of the system, the algorithms, data points and so on. Reverse engineering and forward engineering are main stages for reengineering. For each stage, several aspects for web-based system reengineering should be considered, these aspects are applied though the whole reengineering life cycle, from analyse of legacy system to design of new system. Therefore, the author identifies six main aspects to consider during reengineering a software application, namely architecture, function, interface, language, data storage and algorithm.

#### 5.1.1 Architecture

Development framework is the first aspect to consider when it comes to design the software. In order to achieve low coupling and high cohesion, 3-tier architecture and MVC pattern become very practical for nowadays web-based system, and different combination of frameworks can efficiently develop the system. For enterprise system, if there are limited requirements in front-end, for instance when user only acquires data from database without further operation, “struts 2 + hibernate” is sufficient, because it is easy to build and maintain, whereas if there are more actions for the data and more functions, “Spring MVC + Spring + Hibernate” is preferred to deal with such complex architecture. The integrated frame for simple functioned enterprise web-based system is shown in figure 5-1.

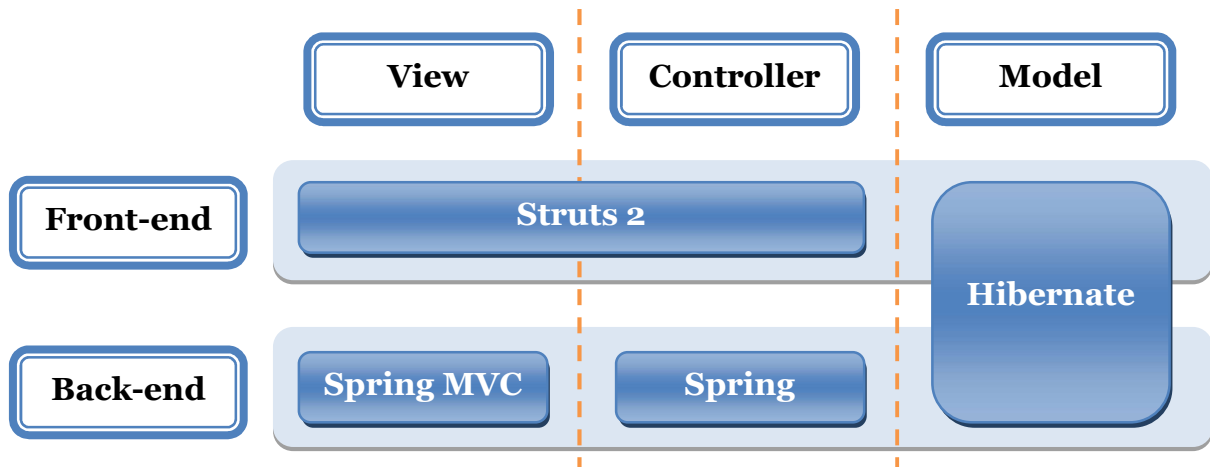


Figure 5-1: The integrated frame for simple functioned enterprise web-based system

### 5.1.2 Function

The aspect of new functions is obvious to consider when reengineer a legacy system, and it is usually the fundamental part to upgrade a system. The requirements are proposed to meet nowadays business demands from both internal employees and external customers. Three methods are proposed and applied for this project to determine new functions, namely direct demands, interview and peer study. The relations among the three factors are shown in figure 5-2.

First of all, the new functions are assigned by the product owner who points out which functions need to add to the new system. Secondly, interview is another way to perceive requirements from users. During the conversations, users may propose new ideas according to their daily behavior and obstacle while using the legacy system. These ideas are collected and discussed with the product owner one by one, and then certain functions are defined and prioritized. Based on the scope, budge and time, new functions are finally confirmed and integrated into the whole system design. Thirdly, peer study is a method to understand nowadays trend in the same industry. By analyzing systems of the peer companies, there may have certain functions which are important but neglect by our users. All in all, by the integrated methods the new functions could be defined quickly and thoroughly.

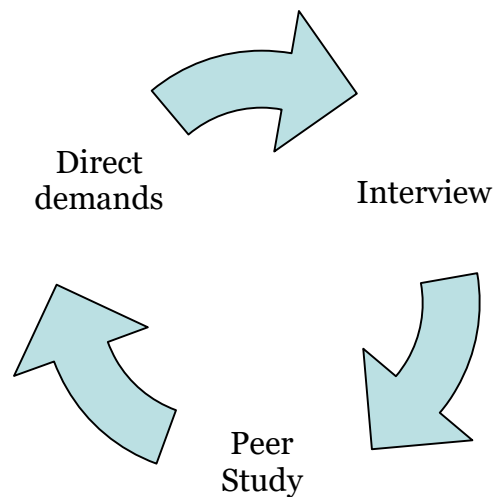


Figure 5-2: Three methods to explore user requirements

### 5.1.3 Interface

Along with the new technologies, the interface becomes more and more responsive and beautiful, many plugins and templates make the interface more user friendly and easy to programming. Moreover, mobile version is convenient to develop. The core technologies are still HTML5, CSS and JavaScript, but some tool could be considered for fast development, such as Bootstrap which is HTML, CSS, and JS framework and focusing on responsive and mobile first interface.

Besides of technology, the style and layout of the interface should be discussed with the stakeholders. The methods to design the new interface are similar as function design in figure 5-2.

### 5.1.4 Language

Programming language is an important aspect to consider, either to keep the original language or to change a new language, and if change which language will use. Because language is related to other parts of the system, such as architecture, in this case SSH framework is developed by Java, so if Java is chosen the architecture is applied. Otherwise if the language keeps PHP as in the legacy system or the language will use C#, the framework will be reconsidered correspondingly.

The strategy to choose language is mainly based on the company demand which may related to other facilities such as database. In this project, Java language is applied due to easy to maintain, wildy used for industry, quick to develop with lightweight framework.

### **5.1.5 Data Storage**

Regarding data storage, it could be to change a format to store data or to improve the current performance. In this research, it focus on legacy system which developed many years ago, data storage could be inconvenient and unsafe, such as XML file. Therefore, a relationship database is introduced to a new system. When it comes to choose a proper database, several factors are considered, such as budget, functions and so on. MySQL database is recommended to small or medium company with limited budget and simple usage for data.

### **5.1.6 Core Algorithm**

Reengineering is good at abstracting the current algorithm from the legacy system, and a flowchart could be drawn based on the existing code. If the algorithm will keep similar in the new system, the code logic will keep similar whereas replaced by a new language, in such case the algorithm part of the new system will be developed very fast. Another advantage is even the programmer doesn't understand the business logic and the algorithm, and as long as the code logic is abstracted, the programmer can still develop the system based on the existing code and flowchart.



## 5.2 Front-end

Based on reengineering philosophy, the new system is developed. It is low coupling and high cohesion built on advanced technology, such as Java, MySQL, HTML5, etc., and modern architecture like MVC is adopted. And meanwhile the main aspects to implement reengineering during the reengineering are discussed in this chapter. The front-end of the new system will be explained in this section, the overall page is shown in figure 5-3.

### Pump Selection

With Grindex curve generator, both pump- and system-curves are generated by specified data.

[More »](#)

#### Basic Search

Family ☒ Drainage ☒ Sludge ☐ Slurry

Feature ☒ Include Stainless ☐ Include Obsolete

Frequency ☒ 50Hz ☒ 60Hz

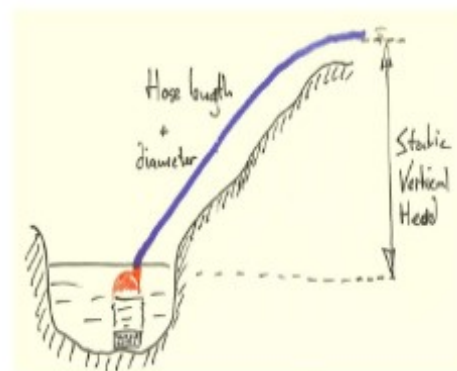
Required Flow  l/s

Vertical Head  meters

Hose Length  meters

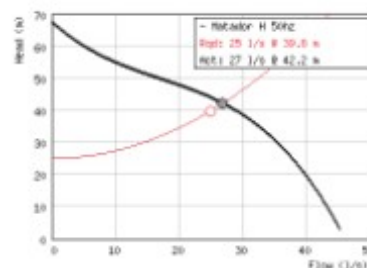
Hose Diameter  mm

[Advanced Search](#)



#### Possible Pumps

Name	Parallel	Serial	Flow (l/s)	Flow Difference	Velocity Discharge (meters/s)	Pump+Motor Efficiency
Matador-H-60Hz	1	1	44.6	15.2%	6.52	60.0%
Matador-H-50Hz	1	1	42.2	7.9%	6.11	59.0%
Master-H-50Hz	2	1	39.9	0.6%	5.7	53.0%
Master-H-60Hz	2	1	42.3	6.3%	6.13	51.0%
Matador-N-60Hz	1	1	35.5	-15.6%	4.78	36.0%
Magnum-N-50Hz	1	1	44.0	13.4%	6.42	35.0%
Maxi-N-60Hz	1	1	45.4	17.5%	6.65	32.0%
Inox-Majon-N-50Hz	1	2	35.5	-15.4%	4.79	30.0%



Selected pump is **Matador-H-60Hz**

Duty point: 25.0 l/s at 39.8 meters

(Static head 25.0 meters + friction 14.8 meters)

Design velocity: 6.88 (meters/s) through 75.0 mm

(Recommended between 2 and 4 meters/s)

[Data sheet for download](#)

[German](#) [Spanish](#) [French](#) [English](#) [Swedish](#)

[Generate PDF with selection details](#)

Figure 5-3: The overall page of front-end

### 5.2.1 Search Field

The search function is used to calculate the possible pumps from the database according to the given parameters. In order to meet the requirement of simple input interface, the search function is divided into two parts, namely Basic Search and Advanced Search, see figure 5-4. For “Basic Search”, user first checks expected family, feature and frequency, and secondly the user enters the required values of flow, vertical head, hose length and hose diameter, and meanwhile chooses the corresponding units which is according to the standards in different counties. In addition, certain validation criteria are set, for instance, in the type of family or frequency, each one is required to check at least one option, and the value texts must be filled in.

#### Pump Selection

With Grindex curve generator, both pump- and system-curves are generated by specified di

[More »](#)

The screenshot displays the 'Pump Selection' interface. At the top, a header 'Pump Selection' is followed by a descriptive line: 'With Grindex curve generator, both pump- and system-curves are generated by specified di'. Below this is a 'More »' link. The main form is titled 'Basic Search' and contains several sections: 'Family' with radio buttons for 'Drainage', 'Sludge', and 'Slurry'; 'Feature' with checkboxes for 'Include Stainless' and 'Include Obsolete'; 'Frequency' with checkboxes for '50Hz' and '60Hz'; and four input fields for 'Required Flow', 'Vertical Head', 'Hose Length', and 'Hose Diameter', each with a corresponding unit dropdown menu (l/s, meters, mm). Below the 'Basic Search' section is an 'Advanced Search' section, which is currently collapsed. It contains input fields for 'Max flow deviation up to (%)', 'Min flow deviation up to (%)', 'Max head deviation up to (%)', 'Min head deviation up to (%)', 'Max parallel pumps' (with a default of 2), and 'Max serial pumps' (with a default of 2). A 'Search' button is located at the bottom of the form.

Figure 5-4: Search field, Front-end

For “Advanced Search”, it is designed for professional user who has specified requirements, for instance, the default numbers of pumps in parallel and in serial are 2 separately, whereas the user can give different values in the advanced search. A collapsed panel is designed for the function in order to enhance user experience.

Regarding the meanings of the parameters, an explanatory figure is placed in the right of the search frame. User could give correct values so as to calculate comparatively reasonable results according the figure.

## 5.2.2 Result Field

The possible results are calculated based on the Grindex pump selection algorithm. The results will be displayed in a table, see figure 5-5, where certain important parameters will be available, such as name, parallel, serial, flow, velocity, efficiency, etc. The default order is descending according to “Pump + Motor Efficiency”, whereas user can customize the order by user’s requirement. The total result is available by the right side of “Possible Pumps”.

When the user chooses one of the possible pumps, the corresponding curve will display, and meanwhile detailed calculation about the input parameters will explained below the curve, such as duty point, design velocity, etc. Furthermore, data sheet for the selected type pump is available to download in five different languages, and another pdf file of pump could be generated.

In the result table, same pump name may appear several times because the same pump with different combination of parallel and serial could get different efficiency but all satisfies the input parameters which may help the user to find the most cost efficiency pump.

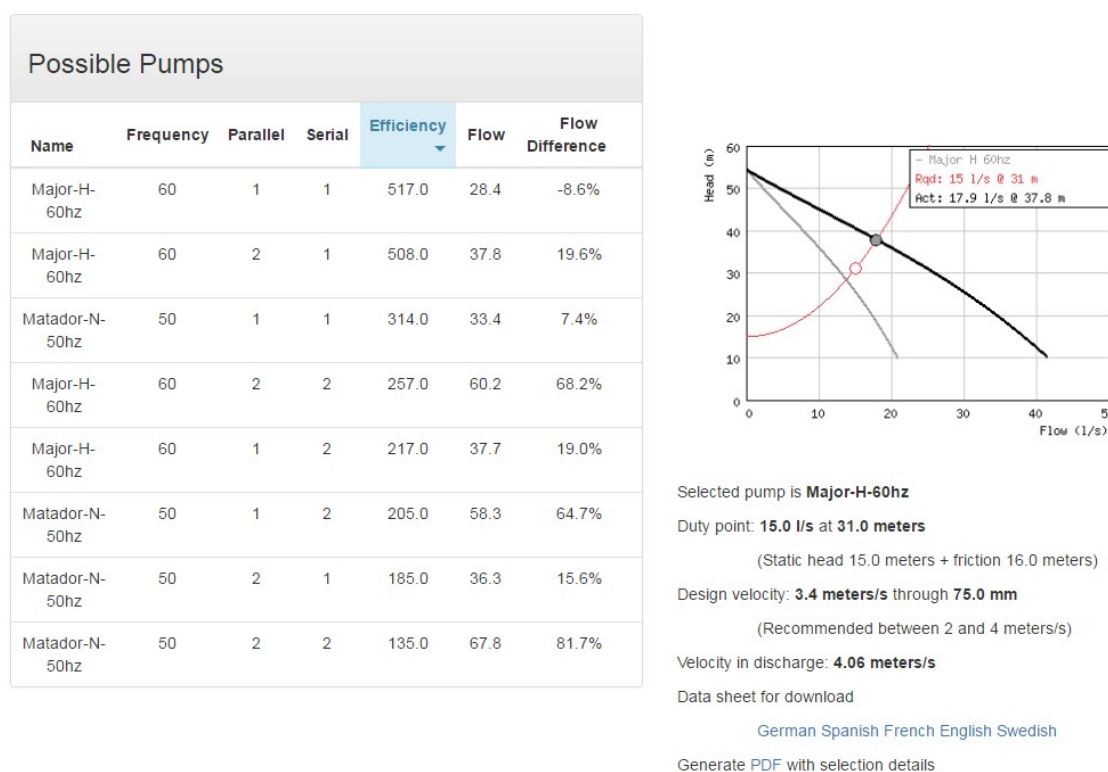


Figure 5-5: Search result display, Front-end

## 5.3 Back-end

Grindex Pump Management Platform is the back-end system. The system is used to operate pump parameters, which is connected to the database. The user-interface is well-functioned and easy to use. The general layout is that the top of the page is the title banner, the left side is navigation bar, the main content is in the middle right. The content is protected, and user needs to login so as to operate the data. The layout is illustrated in figure 5-6.

Sitemesh is a tool to unify page layout of the back-end, which makes a clean separation of content from presentation. Specifically, the top banner and navigation bar are in a main page as decorator, and all pages use this decorator as background. The advantage is all webpages are consistent, if one object is changed in the decorator, all pages will be changed accordingly.

### 5.3.1 User Management

The back-end is designed for limited internal users, therefore all users have the same authorization to add, delete, update and view pump information from database.

- User Login

For security reason, login function is required. By correct username and password, the user could log into the system for further operations. Figure 5-6 is the login page.

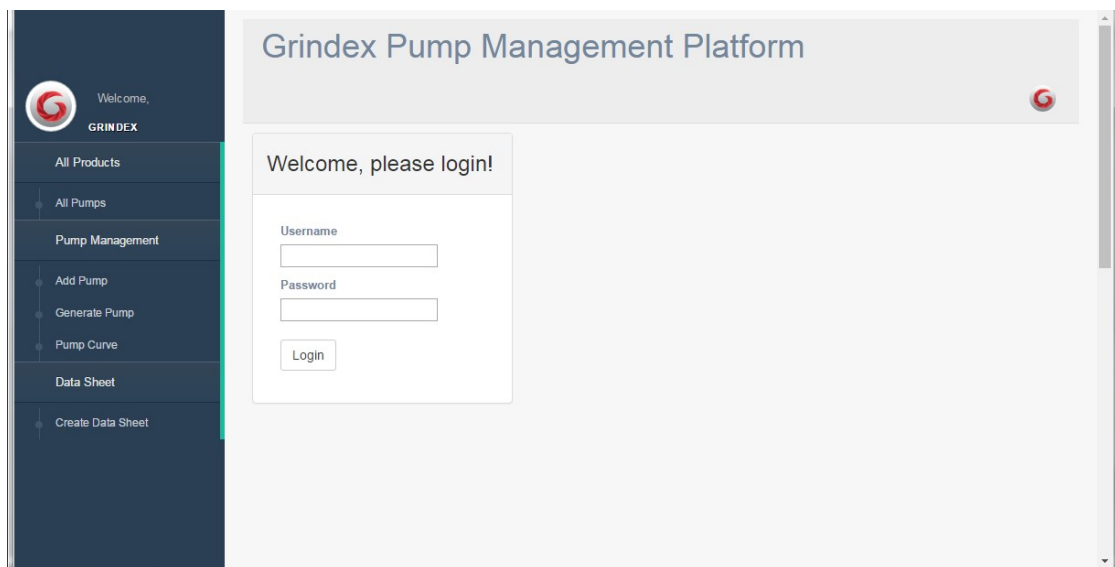
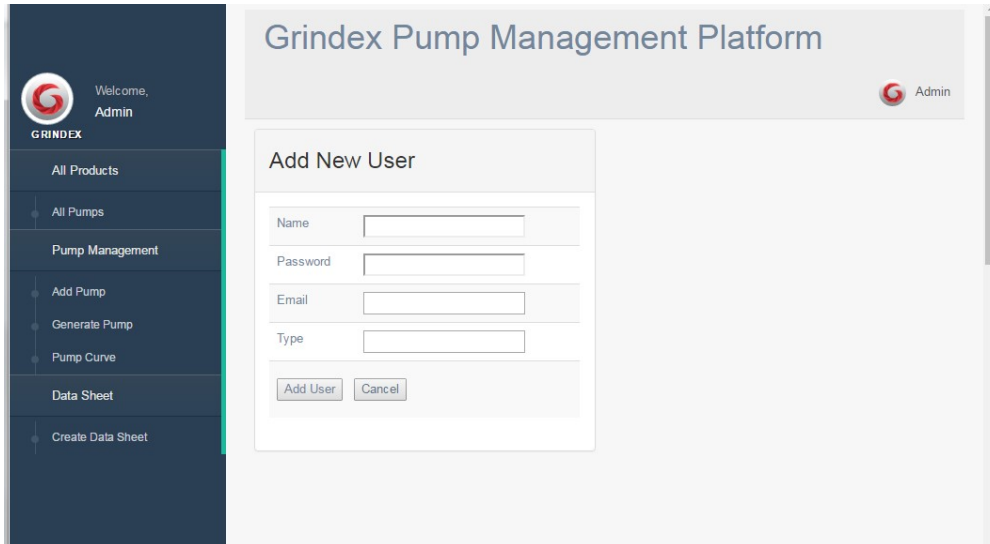


Figure 5-6: Login page, Back-end

- User Creation

Only administrator can create or delete a user. The administrator can log on to the system by a special username and a password. The function of adding user is shown in Figure 5-7.

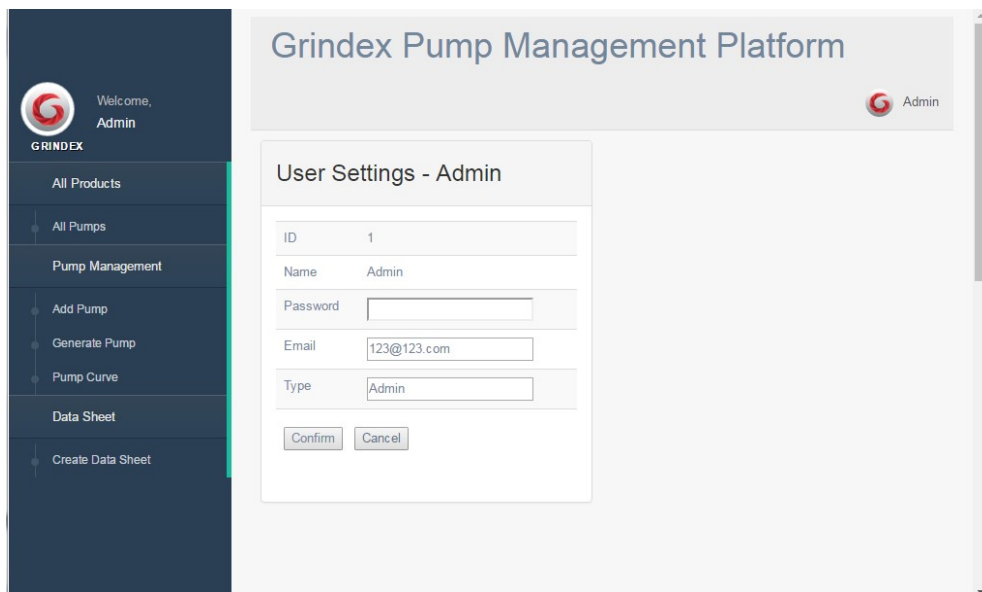


The screenshot shows the 'Grindex Pump Management Platform' interface. On the left is a dark blue sidebar with the Grindex logo and a 'Welcome, Admin' message. Below this is a list of menu items: 'All Products', 'All Pumps', 'Pump Management', 'Add Pump', 'Generate Pump', 'Pump Curve', 'Data Sheet', and 'Create Data Sheet'. The main content area has a light gray header with the platform name and a user profile icon labeled 'Admin'. Below the header is a white box titled 'Add New User' containing four input fields: 'Name', 'Password', 'Email', and 'Type'. At the bottom of this box are two buttons: 'Add User' and 'Cancel'.

Figure 5-7: Add new user page, Back-end

- User Settings

The user can change his/her own information such as password, which can't be changed by and displayed to the administrator. The function is implemented in the "setting", which locates in the top banner, besides Grindex logo with corresponding username. It is a drop-down menu, and it contains functions of "setting" and "logout". "Setting" is used for personal information, such as password, email address, etc. Figure 5-8 is page of user settings.



The screenshot shows the 'Grindex Pump Management Platform' interface with the 'User Settings - Admin' page. The sidebar is identical to Figure 5-7. The main content area has a light gray header with the platform name and a user profile icon labeled 'Admin'. Below the header is a white box titled 'User Settings - Admin' containing five input fields: 'ID' (with value '1'), 'Name' (with value 'Admin'), 'Password', 'Email' (with value '123@123.com'), and 'Type' (with value 'Admin'). At the bottom of this box are two buttons: 'Confirm' and 'Cancel'.

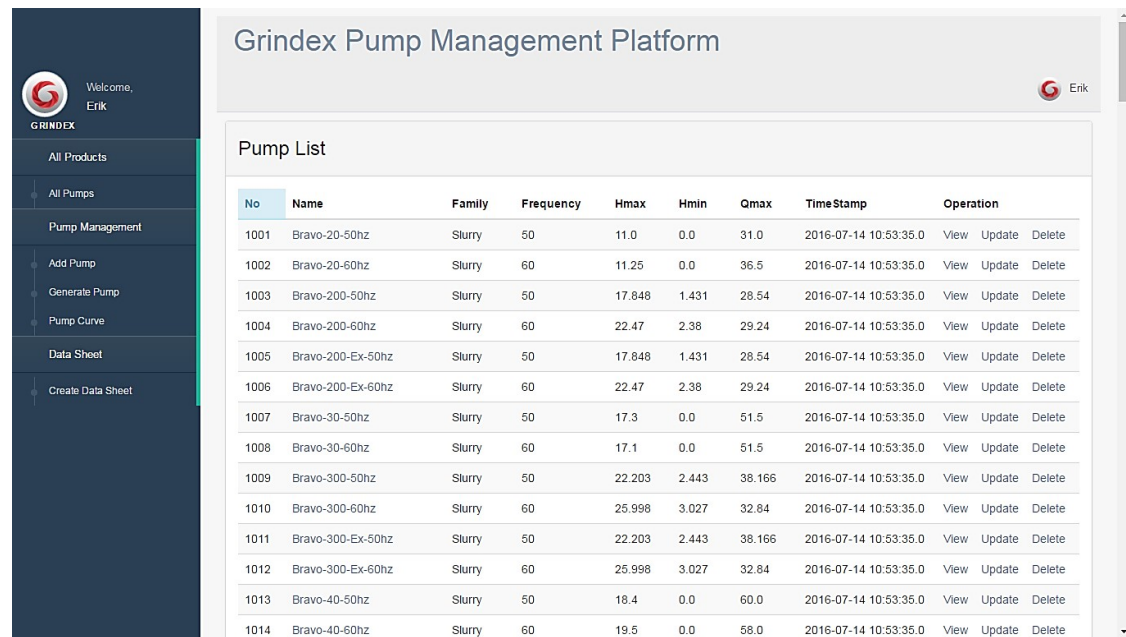
Figure 5-8: User settings page, Back-end

### 5.3.2 Pump Management

The main functions for the pump management are to create, read, update and delete (CRUD). Extra functions are for example sort, time stamp, history log and so forth.

- View Pumps

In the navigation bar, “All pump” is to view all pumps in the table, where the basic information for each pump is listed such as name, family, frequency, etc. If user wants to operate certain pump, “view”, “update” and “delete” can be chosen in the operation square. The view page of the back-end system is shown in figure 5-9. In the table of all pumps,



The screenshot displays the Grindex Pump Management Platform interface. On the left is a dark blue sidebar with a navigation menu. The main content area has a light gray header with the title 'Grindex Pump Management Platform' and a user profile 'Welcome, Erik'. Below the header is a 'Pump List' section containing a table with 14 rows of pump data. Each row includes a pump number, name, family, frequency, and various performance metrics, along with 'View', 'Update', and 'Delete' action buttons.

No	Name	Family	Frequency	Hmax	Hmin	Qmax	TimeStamp	Operation
1001	Bravo-20-50hz	Slurry	50	11.0	0.0	31.0	2016-07-14 10:53:35.0	View Update Delete
1002	Bravo-20-60hz	Slurry	60	11.25	0.0	36.5	2016-07-14 10:53:35.0	View Update Delete
1003	Bravo-200-50hz	Slurry	50	17.848	1.431	28.54	2016-07-14 10:53:35.0	View Update Delete
1004	Bravo-200-60hz	Slurry	60	22.47	2.38	29.24	2016-07-14 10:53:35.0	View Update Delete
1005	Bravo-200-Ex-50hz	Slurry	50	17.848	1.431	28.54	2016-07-14 10:53:35.0	View Update Delete
1006	Bravo-200-Ex-60hz	Slurry	60	22.47	2.38	29.24	2016-07-14 10:53:35.0	View Update Delete
1007	Bravo-30-50hz	Slurry	50	17.3	0.0	51.5	2016-07-14 10:53:35.0	View Update Delete
1008	Bravo-30-60hz	Slurry	60	17.1	0.0	51.5	2016-07-14 10:53:35.0	View Update Delete
1009	Bravo-300-50hz	Slurry	50	22.203	2.443	38.166	2016-07-14 10:53:35.0	View Update Delete
1010	Bravo-300-60hz	Slurry	60	25.998	3.027	32.84	2016-07-14 10:53:35.0	View Update Delete
1011	Bravo-300-Ex-50hz	Slurry	50	22.203	2.443	38.166	2016-07-14 10:53:35.0	View Update Delete
1012	Bravo-300-Ex-60hz	Slurry	60	25.998	3.027	32.84	2016-07-14 10:53:35.0	View Update Delete
1013	Bravo-40-50hz	Slurry	50	18.4	0.0	60.0	2016-07-14 10:53:35.0	View Update Delete
1014	Bravo-40-60hz	Slurry	60	19.5	0.0	58.0	2016-07-14 10:53:35.0	View Update Delete

Figure 5-9: View all pumps page, Back-end

- Update Pumps

If user wants to update or delete a pump, there is a “update” button in the all pumps table. For detailed information user can click “view”, the interface of “view” is similar as “update” page. The update page is shown in figure 5-10.

Grindex Pump Management Platform

Welcome, Admin

Pump Update - Bravo-20-50hz

ID	1001
Name	Bravo-20-50hz
Family	Slurry
Frequency	50
Hmax	11.0
Hmin	0.0
Qmax	31.0
A3	-1.8849796889422E-4
A2	0.0042235288144842
A1	-0.30753075039857

Figure 5-10: Update pump page, Back-end

- Add Pumps

“Add pump” is to create a new pump into the database. For the rest options in the navigation bar are designed for the further work, right now they are linked to list all pumps. The add page of the back-end system is shown in figure 5-11.

Grindex Pump Management Platform

Welcome, Admin

Add New Pump

Name	
Family	
Frequency	
Hmax	
Hmin	
Qmax	
A3	
A2	
A1	
A0	

Figure 5-11: Add new pump page, Back-end

- Error Page

In the add function, there is server-side validation regarding pump name and family which are not allowed to be empty. The timestamp is also created automatically. Moreover, error page is also considered for example 404 error, 500 error, NullPointerException and so on. The error page is shown in figure 5-12.

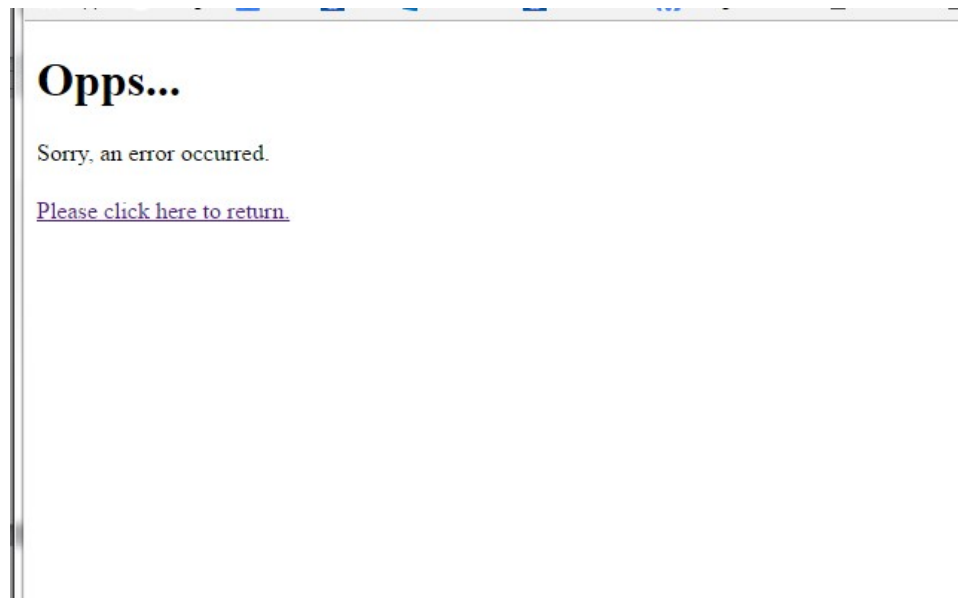


Figure 5-12: Error page, Back-end



## **6 Analysis and Evaluation**

The purpose of this thesis is to research on how to develop web-based software by reengineering. To better respond the purpose, Grindex AB as a case study is selected. The research is based on Chikofsky and Cross' [8] reengineering model, that is reverse engineering, forward engineering and restructuring. What is more, modern methods of software development are adopted, which are MVC design pattern, Java integrated framework, relationship database and so forth. In this chapter, author will analyse the new system by comparing with the legacy system, and meanwhile evaluate the overall performance by a qualitative method to interview the target groups.

### **6.1 Comparison between legacy system and new system**

Based on the results of “the main aspects to consider” in Chapter 5, a comparison between legacy system and new system will be discussed in this section, the key differences are list in table 6-1.

#### **6.1.1 Advantages of reengineering and new system**

By reengineering, the development time is shorter because some existing materials could be reused, such as core algorithm, layout of user interface and so on. Moreover, lower risk is another character for applying reengineering, because reengineering is based on incremental improvement of systems, rather than radical replacement. The critical business knowledge will not lose drastically.

From all the six aspects we discussed in the thesis project, the new system performs better than the legacy system. Specifically, by using Java and other advanced tools the new system has more robust architecture for easy maintain and extend, more functions are achieved, the interfaces of both front-end and back-end are more interactive and responsive, and data storage is more flexible and safe by adapting relationship database.

When it comes to the programming aspect, the new system is programmed in Eclipse developing tool, and the structures of the projects are clear and well organized following the industry standard comparing to the legacy system. The code is more compact and efficient, as well as sufficiently documented.

#### **6.1.2 Disadvantages of reengineering and new system**

The current version of the new system is a core framework for the next generation product selection system, it support most core functions, but datasheet and pump curve are not ready yet, so it can't put online right now.

Table 6-1: Key differences between legacy system and new system

	Legacy System	New System
<b>Architecture</b>	Basic 3 tier architecture Client/Server model	Model-View-Controller (MVC) Front-end: Struts2 + Hibernate Back-end: Spring MVC + Spring + Hibernate
<b>Function</b>	Core functions	Enhanced functions based on direct demands, interviews and peer study
<b>Interface</b>	Front-end: Simply interface Back-end: Complex logic and not user friendly	User friendly interface Input validation
<b>Language</b>	PHP HTML/CSS/Javascript Non-UML	Java Bootstrap/Javascript UML
<b>Data Storage</b>	XML document	MySQL database
<b>Core Algorithm</b>	Existed but no process flowchart	Keep the same algorithm but draw the flowchart clearly

## 6.2 Evaluation of reengineering and new system

Qualitative analysis is a method to evaluate the deliverables, and open-ended questions are useful for providing in-depth information. Therefore several interviews have been conducted regarding user experience and technical performance after reengineering the legacy system.

### 6.2.1 Evaluation of user experience and functionality

Regarding user experience and functionality, an interview was conducted with employees in technical support team at Grindex AB. Generally the user experience is good for both front-end and back-end, especially the back-end is a completely new design, although some small bugs still need to be improved. The functions are implemented as expected, however, because the result is a core framework instead of a mature product to put online, so they are expecting more functions in the next version. The interview questions and answers are shown in table 6-2.

Table 6-2: Interviews regarding interface and functions

---

Q1: What do you feel the front-end interface as a customer?

The general feeling is modern and fresh, and it keeps the layout which I'm used to. I know how to navigate and how to find the result. The result table is better than list as before. And "Advanced Search" in a collapse way makes it cool. However, the table size should be constrained and scroll bar, now it's too big if there are many results.

Q2: What do you feel the interface of Pump Selection Platform (back-end)?

The platform is much better than before, this is an interactive interface as other nowadays systems, it's very clear to find the function. So from the interface aspect I think it's good and as you pre-defined it will be easy to add more functions in this interface.

Q3: What do you feel the functions of pump selection (front-end)?

As discussed before you designed the system, the main functions should be kept and add some new functions, so in this new system I feel it follows the suggestions and it's well functioning. The advanced search function is now designed but not implemented yet, so I hope it could be done.

Q4: What do you feel the function of the back-end platform?

I'm satisfied with the current functions to manage existing pumps and add

---

---

new pumps. In addition, it's very important to have login feature and user feature, so from security aspect it's safer. However I hope for more functions in the next version, for example filter function, due to huge amount of pump data it's not easy to find certain one quickly.

---

### 6.2.2 Evaluation of technical performance

Regarding technical issue which are architecture, language, database, an interview was conducted with Product Owner. Generally he is satisfied with the technologies in this project, and it's up-to-date and easy to extend for further development, however some improvements are proposed as well. The interview questions and answers are shown in table 6-3.

Table 6-3: Interviews regarding technical issues

---

Q1: What do you think about the architecture of the new system?

MVC is well structured, as expected it will be easy to maintain and extend for new functions in the future. UML analysis, such as use case and sequence diagram, is a good method to understand the system and where to improve.

Q2: What do you think about Java as programming language?

As we discussed before, I think Java is good for industry and I don't see the difference between Java and other language. Moreover, I think the SSH framework of Java works quite well for the new system.

Q3: What do you think about the data storage?

The database is a must to use in the new system, and you manage it well with the existing XML file. In addition, MySQL is an open source, so it's suitable for the current level of this system at Grindex. However, the datasheet and pump curve are not connected to database yet, so it needs to be added as a complete system.

Q4: What do you think about the core algorithm?

The algorithm is a bit complicated, but you manage to understand by analyzing the codes, and surprisingly drew the flowchart to visualize the process. The algorithm works correct in the new system.

---

## **7 Conclusion and Future Work**

This thesis work researches on the main aspects to be consider in practicing reengineering for web-based software. By developing product selection software at Grindex Ab, the author applies the reengineering methodology in a real case which is reverse engineering, forward engineering and restructuring. In the stage of reverse engineering, the legacy system is analysed in different aspects thoroughly, and then transformation analysis is conducted by competitive analysis and semi-conducted interviews. Finally forward engineering is carried on to develop the new system. During the reengineering process, the author identifies six main aspects to consider during reengineering a software application, namely architecture, function, interface, language, data storage and algorithm. Based on the result, a further analysis and evaluation are performed.

The current results are build an overall architecture for the system, and key functions and algorithms are implemented, however, more functions need to be designed and implemented in the next step, for example, the system will generate datasheet and plot curve based on the data points from database.

Along with the new technology, some frameworks or technologies would be replaced, which means the reengineering will happen again, however the methodology would be similar.

Since many companies are facing upgrade system from legacy system, it may be possible to create a general model of the reengineering to apply for certain demands of companies so as to make it more efficient.

## References

- [1].Sneed, H.M.: Software Renewal: A case Study. In: IEEE Software 1 (1984), S. 56–63. – ISSN 0740–7459. – DOI 10.1109/MS.1984.234710
- [2].Canfora, Gerardo ; Cimitile, Aniello: Software Maintenance. Benevento Palazzo Bosco Lucarelli, Piazza Roma 82100 : Uni-versity of Sannio, November 2000
- [3].Seacord, Robert C. ; Plakosh, Daniel ; Lewis, Grace A.: Modernizing Legacy Systems: Software Technologies, Engineer-ing Process and Business Practices. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2003. – ISBN 0321118847
- [4].De Lucia, A. ; Giordano, M. ; Polese, G. ; Scanniello, G. ; Tortora, G. ; Role based reengineering of Web applications. Seventh IEEE International Symposium on Web Site Evolution 2005, pp.103-110
- [5].De Lucia, A. ; Francese, R. ; Scanniello, G. ; Tortora, G. Reengineering web applications based on cloned pattern analysis, Proceedings 2004, pp.132-141
- [6].C. Wagner, “Chapter 2 Fundamentals,” Model-Driven Software Migration: A Methodology, Potsdam, Germany: Springer Vieweg, 2014, ch. 2, sec. 3, pp. 26
- [7].Håkansson, A., 2013, January. Portal of research methods and methodologies for research projects and degree projects. In Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS) (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).
- [8].Chikofsky, Elliot J.; Cross, James H.: Reverse Engineering and Design Recovery: A Taxonomy. In: IEEE Software 7 (1990), Nr. 1, S. 13–17
- [9].D. Casey Tucker, Devon M. Simmonds, "A Case Study in Software Reengineering," itng, pp.1107-1112, 2010 Seventh International Conference on Information Technology, 2010
- [10]. Yener, Murat, and Alex Theedom. Professional Java EE design patterns. Chap 14. John Wiley & Sons, 2014.
- [11]. Yates, C., Ladd, S., Deinum, M., Vervaet, E., Serneels, K. and Vanfleteren, C., 2012. Pro Spring MVC: With Web Flow. Apress.
- [12]. Gupta, P., Bari, P. and Govil, M.C., 2010. Spring Web MVC Framework for rapid open source J2EE application development: a case study. Interface, 2(6), pp.1684-1689.
- [13]. Johnson, R., Hoeller, J., Donald, K., Sampaleanu, C., Harrop, R., Risberg, T., Arendsen, A., Davison, D., Kopylenko, D., Pollack, M. and Templier, T., 2004. The Spring Framework–Reference Documentation. Interface, 21.
- [14]. Safonov, V.O., 2008. Using aspect-oriented programming for trustworthy software development (Vol. 5). John Wiley & Sons.

- [15]. Ren, Y., Jiang, D., Xing, T. and Zhu, P., 2011. Research on software development platform based on SSH framework structure. *Procedia Engineering*, 15, pp.3078-3082.
- [16]. Li, K., Yang, K. and Zhu, S., 2012, July. A web management platform of Internet-based electrical engineering lab: Using SSH framework. In 2012 7th International Conference on Computer Science & Education (ICCSE).
- [17]. [www.java.com/en/about/](http://www.java.com/en/about/)
- [18]. Dwarampudi, V., Dhillon, S.S., Shah, J., Sebastian, N.J. and Kanigicharla, N., 2010. Comparative study of the Pros and Cons of Programming languages Java, Scala, C++, Haskell, VB. NET, AspectJ, Perl, Ruby, PHP & Scheme-a Team 11 COMP6411-S10 Term Report. arXiv preprint arXiv:1008.3431.
- [19]. Kayal, D., 2008. *Pro Java EE Spring Patterns*. Dreamtech Press.
- [20]. [www.php.net](http://www.php.net)
- [21]. [www.xml.com](http://www.xml.com)
- [22]. Codd, E.F., 1970. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), pp.377-387.
- [23]. Blackwell, A. H., Manar, E., eds. "Prototype". *UXL Encyclopedia of Science* (3rd ed.). Retrieved 13 July 2015.
- [24]. T.J. Biggerstaff, Design recovery for maintenance and reuse, *Computer*, July 1989, pp. 36-49.
- [25]. Laplante, Phillip (2007). *What Every Engineer Should Know about Software Engineering*. Boca Raton: CRC. ISBN 978-0-8493-7228-5.

TRITA TRITA-ICT-EX-2017:18