ARCADA

Requirements of migrating work order tools and database to a Salesforce environment

Rasmus Nyqvist

Degree Thesis

Informations- Och Medieteknik

2016

Rasmus Nyqvist

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och Medieteknik
Identifikationsnummer:	5686
Författare:	Rasmus Nyqvist
Arbetets namn:	Krav på att migrera arbetsorderverktyg och databas till en Salesforce miljö.
Handledare (Arcada):	Jonny Karlsson
Uppdragsgivare:	Inspecta Oy

Sammandrag:
Syftet med detta examensarbete är att identifiera och presentera krav på arbetsorders processen hos företaget Inspecta Oy, för att migrera associerade verktyg till en ny miljö. Inspecta har i nuvarande läge en besynnerlig mängd främst MS Excel baserade verktyg arbetarna använder vid granskningar av olika slags mätutrustning för att underlätta granskningsprocessen samt för att generera uppföljbar data samt fakturerings uppgifter automatiskt. Det slutliga målet är för företaget att, till den grad det är möjligt, ha alla dessa olika verktyg implementerade i en och samma användarmiljö vid namn Salesforce med enkel och snabb tillgång till nödvändig kund, apparat och historik data. Detta examensarbete begränsar sig dock till att definiera kraven för det verktyg som används av granskarna för att hantera arbetsorder och även genera fakturerings uppgifter. Det handlar dels om olika aspekter kring själv arbetsprocessen, t.ex. hur en kund är delaktig, samt dels om mer konkreta tekniska krav, t.ex. vilka uppgifter måste anges i formuläret. Målet med detta är att konstruera ett kravdokument samt en databasdesign, för att underlätta det fortsatta förverkligandet av det nya systemet. För att uppnå målet används olika kända insamlings-, klassificerings- samt presentationstekniker av krav och liknande metoder för presentation av en databasstruktur presenterade i använda litterär källor.

Nyckelord:	Inspecta, kravspecifikationer, databaser, UML, molntjänster
Sidantal:	40
Språk:	Engelska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media technology
Identification number:	5686
Author:	Rasmus Nyqvist
Title:	Requirements of migrating work order tools and database to a Salesforce environment.
Supervisor (Arcada):	Jonny Karlsson
Commissioned by:	Inspecta Oy

Abstract:
The purpose of this thesis is to identify and present requirements of the work order process at Insepcta Oy, to migrate related tools to a new environment. In its current state Inspecta has a great number of tools based mainly on MS Excel, used by the employees for the inspection of different kinds of measuring devices to facilitate the inspection process and to automatically store traceable data and invoicing information. The ultimate goal of the company is, to the degree it's feasible, to implement all these tools in a single user environment, called Salesforce, with easy and quick access to all the necessary customer, device and history data. This thesis however restricts itself to define the requirements of the tools used by the inspectors to handle work orders and to generate invoicing information. It is partly about various aspects concerning the work process itself, like how the customer can get involved, and partly about more concrete technical requirements, like what information has to be supplied in the form. The goal of this is to create a requirements document as well as a database design to facilitate the continued implementation of the new system. Various requirements gathering, classification and representation techniques and models as well as presentation models for a database structure are presented the literature references.

Keywords:	Inspecta, requirements specification, databases, UML, cloud services
Number of pages:	40
Language:	English
Date of acceptance:	

CONTENTS

Figures

Tables

ABBREVIATIONS AND TECHNICAL TERMS

Attribute	A.k.a. Column or Field. A database term for characteristics of an entity.
CC	Cloud Computing. Refers to hardware, software and other technical solutions being made available over an internet connection.
CRM	Customer Relationship Management. Refers to methods and technologies used to manage and analyze customer interactions and data.
Database	A data structure held in a computer, for storage of any and all kinds of data.
Domain	A database term for the allowed values of an attribute for all records in a table.
Entity	A.k.a. Table or Object. A database term for a thing or object of interest, for which data is stored in the database.
ERD	Entity Relationship Diagram. A graphical representation of a particular ERM.
ERM	Entity Relationship Model. A model used to describe the attributes of tables and relationships between tables in a database.
IaaS	Infrastructure-As-A-Service. A CC term that refers to a cloud service that allows for utilisation of the hardware resources of the provider.
PaaS	Platform-As-A-Service. A CC term that refers to a cloud service that allows for users to develop their own applications on an online platform maintained by the service provider.
Record	A.k.a. Entry, Row or Tuple. A database term for a single occurrence of an object in the database table.
SaaS	Software-As-A-Service. A CC term that refers to a piece of software made available for users online.
SRS	System Requirements Specification. A document presenting the requirements of a system in natural language and/or with diagrams.

FÖRORD / FOREWORD

I would like to thank Mika Viitapohja, Mikko Törmänen and Kalle Bergman at Inspecta Oy for the opportunity to write this thesis as well as all the help they've provided along the way. Thanks also to all the inspectors who have provided some invaluable information on their work process.

I would also like to acknowledge Hanne Karlsson and Magnus Westerlund at Arcada for their motivational and organisational support. Finally I would like to thank Jonny Karlsson for taking on this thesis as supervisor on such short notice and for all the feedback he's offered.

1 INTRODUCTION

My employer, Inspecta Oy, is currently in the process of implementing a new platform, Salesforce, mainly for handling work orders but also if possible to replace any other tools. Salesforce is a cloud based customer relationship management (CRM) platform. It offers a streamlined, efficient web-based application to handle customer data, for managers to realise new customer opportunities, receive and assign new cases or work orders and for employees to track their tasks. This thesis will mainly analyse the process of receiving new work orders, the work order tool currently in use as well as the data provided for the associated database to determine what will be required to implement a similar process in the new environment. In other words, this thesis will identify the requirements of setting up a functioning work order process on the Salesforce platform.

Inspecta offers a number of services relating to inspecting, testing and certification, as well as providing a variety of technical consulting and training in their fields of expertise. The inspection services provided vary slightly depending partly on the properties of the devices themselves but also on the nature of their usage. If the measuring instrument is in any way used to determine the price of a product it is necessary to verify that they meet the requirements set by law. This is called a *verification of conformity* and such devices are required to be verified periodically to ascertain their continued functionality. Depending on the type of device this verification period is usually either 2 or 3 years, but can be shortened according to the special needs of the customer, never significantly lengthened though. In other cases the inspection is typically performed according the customer's wishes or needs, i.e. verification isn't required as mentioned above. Such an inspection is called a calibration or simply a test and is repeated if and as often as the customer requires it. For the purposes of this thesis, these shall collectively be referred to as *inspection*, and when needed to distinguish between them as simply *verification* and *calibration* respectively.

1.1 Background

Inspecta Oy has for a period of time been acting to improve the productivity of their employees by optimising or streamlining their work process. In the past, different departments of the company have used a variety of tools to do their work, which is understandable considering the great variance in what they actually do, some work with measuring instruments, others with lifting apparatus and others yet with chemicals.

As the company has grown, though, this has resulted in the departments developing these tools into separate divergent wholes, which in at least some aspects are meant to accomplish something very similar, i.e. get a record of invoicing information and of technical data on the devices. This of course means that whenever there's been any form of company-wide changes, all related tools and documents must be updated separately for each branch. Particularly the process of generating work orders and invoices, which are fairly similar for all branches could be implemented in a more unified environment.

It is with this in mind that Inspecta decided to move these processes over to a new platform, eliminating the need for each branch to maintain their own tools for this purpose. The platform is called Salesforce, which is essentially a cloud-based CRM, with a web-based interface, which will be covered more closely in chapter 2.

1.2 Purpose, goals and methods

The purpose of this thesis is to identify what kind of information and automation will be necessary to implement the work order tool on the new platform. This includes having a form for the employees to fill out, a printable document of the performed work for the customer and a database to store customer, device and other data in. This will result in having a requirement specification document defining what is needed for such a form to function properly, i.e. what information has to be filled in for both the purpose of invoicing and traceability as well as generating the printable document for the customer. Another goal is to set up a database design detailing the tables, attributes and relationships needed.

This will be accomplished mainly by analysing the existing work order MS Excel template and the database where the corresponding customer and device information gets stored for invoicing and traceability purposes. Traceability in this case refers to data identifying what devices have been inspected previously, when this was, if and when they should be inspected next, whether they passed the last inspection and other pertinent information. The analysis will also be extended to the actual work process itself to identify possible improvements for the new platform.

Analysis of the Excel template refers to having a closer look at what information has to be provided by the employee, what degree of automation exists in the template and what could be improved upon. Likewise, the database analysis entails having a look at its structure; identifying redundant data fields and entries as well as finding out if there are any improvements to be made.

1.3 Definition

Originally the plan was to analyse a number of other Excel templates as well, to identify what information and functionality would be needed to implement them in the Salesforce environment. There are, however, a great number of such templates and due to temporal limitations on my part I find myself unable to take on a task of this magnitude.

Nor will this thesis do more than slightly touch on the actual implementation of the work order form and associated database in the Salesforce environment. It'll mostly serve as a documentation of information, functionality and structure needed to fully realise the intended purposes of the work order tool.

1.4 Structure

In chapter two there will be a short description of cloud-based services in general and also a closer look at Salesforce itself. Chapter three will be a more detailed description

of methods and materials used for gathering and analysis of requirements, writing a system requirement specification as well as for database structure and design. In the fourth chapter there'll be a detailed report on the findings of each analysis. The following chapter will describe some interesting aspects of the identified requirements of the work order form. The sixth and final chapter will be a description of the database design. The conclusion will naturally summarize the results of this thesis as well as give a couple of implementation suggestions.

2 CLOUD COMPUTING

This chapter includes a brief description of Cloud computing in general, what benefits it offers and the most common types of services provided. There is also an introduction of Salesforce, the services they provide and the degree of customisability available.

2.1 Cloud services

There does not exist any single accepted definition of Cloud Computing (CC), since it is such a broad and relatively new concept, rather there are a number of definitions that all zero in on some aspect of it (Salo 2010, p. 16). The definitions mentioned by Salo can neatly be summarised in two sentences:

1. CC makes IT resources dynamically accessible through the Internet.
2. The resources, which are freely customisable and scalable, can easily and quickly be activated or deactivated as needed.

Being dynamically accessible entails device independence, i.e. being able to access the services or your data in the cloud on any device connected to the internet. Scalable resources ensure that you always have access to the exact capacity that is needed, no more and no less. Another important aspect of CC is the ability to manage these resources yourself, as in being allowed to upgrade to higher capacity without contacting the provider. In this way, since one user isn't utilising all of the capacity the provider is able to provide all of the time, these same resources can be shared by other users according to their needs. This process of sharing resources is something that isn't, nor should it be,

visible to the users or affecting their needs of resources in any way. Information that should be made available to the user of cloud services is the capacity to which they are utilising the resources of the provider. This transparency is a base of trust between the user and the provider, and serves as the grounds on which the customer is billed.

As Cloud Computing is such a broad concept it can be divided into groups based on what kind of services are provided. Salo (p. 22) mentions the split into software as a service (SaaS), platform as a service (PaaS) and infrastructure as a service (IaaS) as the most common grouping. IaaS is the process of acquiring the resources of the provider for your own use. Basically, instead of building a physical server, the server is situated in the cloud and only utilises the necessary resources of the provider. PaaS as the name suggests, provides a platform upon which users can build their own applications according to their own needs. Often these platforms come with pre-built modules for common purposes and additional third-party modules. Instead of the traditional purchase of a license, installing the software and maintaining it; SaaS is paying for the software as it is needed, per user or machine, and having the provider maintain the software itself. SaaS is the largest of the three groups as, according to the IDC (International Data Corporation) as quoted by Salo (p. 22) SaaS represents almost half of all Cloud Computing solutions on the market. A well-established provider of services of these latter two types, PaaS and SaaS, is Salesforce, which will be described in further detail in section 2.2.

2.2 Salesforce

Salesforce started out as a provider of a simple online customer relationship management (CRM) system in 1999, but has since expanded to a multi-faceted provider of cloud based services. Some of their more prominent products include Service Cloud, Marketing Cloud and Analytics Cloud but the main product, consisting of the basic CRM system and tools is simply called Sales Cloud. See Table 1 for a listing of all products along with a short description. Most of these products are implemented as separate applications all running on the same Force.com platform, enabling them to interact closely with each other. The Force.com platform is the basic back-end server structure

upon which the different applications run, making sure everything works in unison. From the user's perspective everything is handled from a simple web application, be it administrative tasks, such as setting up the data structure or user roles, or end-user tasks, like communicating with a customer or closing a deal.

Table 1. Salesforce products and descriptions (Salesforce Developer)

Product	**Usage**
Sales Cloud	Manage your sales process end-to-end
Data.com	Get the right data at the right moment
Service Cloud	Support your customers after the sale
Desk.com	Get all-in-one customer support app; great for small businesses
Marketing Cloud	Manage your customer's journey
Pardot	Automate B2B marketing
Community	Collaborate online with employees, customers, and partners
Chatter	Make your business social and facilitate connections
Analytics Cloud	Drill into your data and get instant answers anywhere, anytime
Platform	Use Heroku and Force.com to build customer-facing and employee cloud apps

Sales Cloud is in its out-of-the-box state used to manage customers as well as potential, ongoing and closed deals. This data is not managed in Salesforce as a traditional relational database with tables consisting of columns and rows, but in a very similar manner, where entities are known as objects, consisting of fields and values or records. Contrary to a traditional relational database structure where fields are limited to text, number, date or Boolean values, these fields can be of some additional data types, as presented in Table 2. Objects may also contain so called relationship fields, taking on the function of a traditional database's primary and foreign key pairings. According to Salesforce Developer these objects allow for a lot more flexible structure, with built-in support for features such as access management, validation, formulas, and history tracking. The application comes with some standard objects for handling data on Accounts, Contacts, Leads and Opportunities. The Account and Contacts objects hold information on companies (business partners, established and potential customers) and contacts (external and internal) respectively, whereas the Leads and Opportunities objects are used to manage potential and established deals or sales respectively. It is however possible to

create your own additional objects to track data relevant to your organisation. Inspecta might for example need to store their customers' different locations of operation and track measuring instruments and other devices and thus create objects for these.

Table 2. Examples of special data types in Salesforce (Salesforce Developer)

Auto Number	A system-generated read-only sequence number, analogous to the SQL identity type.
Checkbox	For representing Boolean data.
Email, Phone and URL	Validated email, phone and URL string representations.
Picklist and Multi-Select Picklists	Represent values from a list.
Currency	Formatted number type, with multi-currency support.
Formula	Read-only field holding data generated from a formula expression.

Most of the other products provided are also applications, i.e. they are software as a service, but they all run on the same Force.com platform, for which users can also create their own applications to add functionality needed in their organisation to the applications already in use. This platform-as-a-service comes with some pre-built modular applications by the provider that can easily be added or simply enabled in the Salesforce environment. Force.com is also used as a term to refer to the actual web-based environment, where developers can customise the user interface of their applications using HTML, CSS and Javascript. For more complex custom logical functionality developers can work with Salesforce's dedicated programming language Apex, which is very Java-like in its syntax. Developers can chose to work in the Salesforce environment, in a browser based IDE, or download the Fore.com IDE plugin for the Eclipse IDE. Applications developed by other third-party companies can also be made available to all on the AppExchange, a sort of app store for Salesforce applications. Salesforce offers all their products in different variations, so called editions, based on the needs of the customer. In general most products come in 3 editions: Professional, Enterprise and Unlimited, where Professional is the most simple, covering only the most basic needs of a company, Enterprise offers a wider array of functions necessary to larger companies, and is as such usually the most popular edition, and finally Unlimited, as the name suggests, offers most everything there is to offer. (Salesforce Help)

3 METHODS

This chapter contains of a description of the methods used in gathering, classifying and presenting system requirements. It also contains information on why and how to design a database: how to recognise entities and their attributes, what a data model is and how it can be used to illustrate the structure of a database.

3.1 Requirements specification

According to Sommerville (2004, p. 64) the process of software development can be divided into four distinct fundamental steps: software specification, software design and implementation, software validation, and software evolution. Specification entails defining the desired functionality and constraints of the software. Design takes over from there, taking on the form of a blueprint for the implementation, which in turn is the actual development of the software. The distinction made between specification and design is not clear in all projects, but Wohlin (2005, p. 95) points out that the specification should be "what" is desired and design should then define "how" to realise it. Validation ensures that the result conforms to the wishes and needs of the customer or end-users and finally the software must evolve to meet potential changes in their needs. The step most relevant for the purpose of this thesis is the first step, specification, which itself can further be split into four phases as defined by Sommerville (p. 75-76): feasibility studies, requirements elicitation and analysis, requirements specification and requirements validation.

The point of the feasibility studies is to quickly and cheaply determine whether or not the proposed system or software is beneficial. Elicitation and analysis refers to observing the existing system, discussing with end-users and analysing procedures to identify system requirements as well as classify them. The next phase is to gather these requirements in a consistent and comprehensible document, often called a System/Software Requirements Specification (SRS). The last step is to validate the requirements, i.e. check them for realism, consistency and completeness. For the subject of this thesis the feasibility has already been determined by the company and need not be discussed,

likewise validation will not be touched upon further; rather the focus will lie with defining requirements and assembling them in an SRS.

3.1.1 Identifying requirements

This can be described as the actual first step in creating an SRS, and there are many different ways to go about gathering requirements, the most relevant of which are document analysis, interviews, use-cases and other scenarios as well as ethnography. Interviews are the quickest and most basic way to get information from the users of the current and future systems, and they can be either closed, with a pre-determined set of questions to be answered (often a form) or open, with no pre-defined questions, rather like a discussion about the system, or a mixture of these, as is most often the case (Sommerville, p. 152). Requirements gained from interviews are prone to errors such as misinterpretation and leaving out information that is perceived to be obvious, and should as such be scrutinised and validated properly.

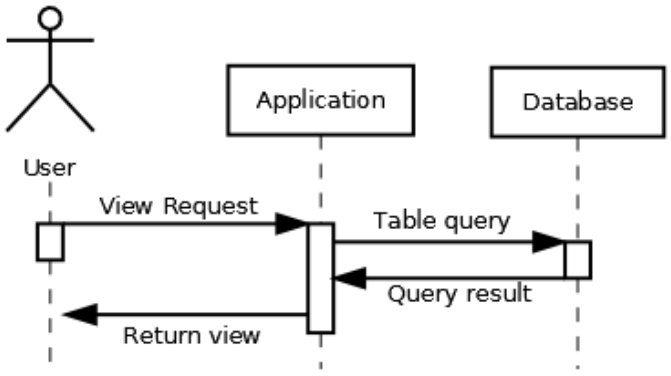

Figure 1.Sequence diagram of simple successful table view request

A scenario is a description of an interaction session either in the current system or as a simulation of an ideal system. A typical scenario consists of a description of the starting point, the supposed normal flow of events, things that may go wrong and a description of the finishing point. Scenarios can be documented freely as text, diagrams and pictures, but an example of a more structured approach is the so called use-case. A use-case identifies an individual action or process within the system, by most commonly representing actors and actions in a diagram. They can consist of only formatted text,

but are usually linked to a diagram made according to some Unified Modelling Language (UML) model. A popular diagram format is the sequence format which illustrates the interactions between an actor (e.g. user) and different parts of the system (e.g. database) as presented in Figure 1. (Sommerville, p. 153-156)

Ethnography is a form of gathering requirements where the analyst observes the everyday working environment of the future system, noting what, how and by whom tasks are performed. It is an effective approach in identifying requirements that specify how people actually prefer to work rather than how they're supposed to work according to other requirements. However Sommerville mentions (p. 158) that it's not an appropriate technique to discover non-user related requirements, and should as such be used in conjunction with other approaches, such as use-cases.

3.1.2 Requirement classification

Requirements can be classified or grouped in a number of different ways, depending on the degree of detail, what part of the process it concerns or even who it concerns. The distinction between different classes isn't always crystal clear, nor do their definitions appear to be that clear, since Sommerville (p. 118 - 131) makes two sets of distinction: abstract contra detailed, and functional contra non-functional requirements; whereas Wohlin (p. 98-99) makes an additional distinction between product (or system) and organisational requirements, as well as pointing out that some distinction can also be made depending on who or what is the source of a specific requirement. Sommerville counts the product-organisation distinction as types of non-functional requirements, adding the third type: external requirements.

Apart from the above they do agree on the definitions of these classes. An abstract requirement, or user requirement as Sommerville refers to them (p. 118), is a statement of what is expected of the product or what constraints are placed on it, expressed in a natural free-flowing language. Opposed to that is a detailed definition of a system function, service or specific constraint, called a system requirement by Sommerville. A broad us-

er requirement can often be broken down into several more detailed system requirements, providing a more technical definition of requirements and constraints. This distinction exists to provide information to different types of readers: managers and some users might only get confused by low-level detailed descriptions of the system, and prefer an approach in a natural language; on the other hand developers and designer benefit from the detailed descriptions when implementing them in the system.

Functional requirements refer to specific functions or services that the system should provide, or how it should or shouldn't handle certain situations. Non-functional requirements on the other hand often set constraints on such functions or services, and are actually more commonly aimed at the system as a whole rather than individual functions (Wohlin, p. 99). A functional requirement could be that a manager should be able to retrieve and employee's schedule, and a non-functional requirement on that operation could be that it should be retrievable within 10 seconds. Sommerville (p. 123) classes non-functional requirements based on where they derive from:

1. *Product* requirements are constraints or requirements set on the actual system or software itself, like response time or usability requirements.
2. *Organisational* requirements come from policies or procedures used by either customer or developer, like different standards or contractual agreements.
3. *External* requirements are set by the environment, e.g. as in other systems and software the product will need to work with, or requirements set by law.

3.1.3 Requirements documentation

Once the requirements are identified and defined they are presented in an SRS document. This document should contain both the less specific user requirements and the detailed system requirements, either integrated in a single description or separated. In cases with a great number of requirements, the system requirements may be assembled in an entirely separate document. For a system being developed by an external developer the SRS needs to be very specific and definite, as opposed to a system developed in-

house when the document can be less detailed since adapting it as needed isn't as time consuming or expensive (Sommerville, p. 136).

An example of a widely known and used standard, the IEEE/ANSI 830-1998, provides a suggestion for a structure of the requirements documentation (see appendix 1). Sommerville (p. 138) points out that it is in essence though only a framework, to be adapted and customized to fit the needs of a particular organisation or process. The information included in an SRS, as well as the form it's presented in, is also heavily dependent on the software in development and the methods used in development. Wohlin (p. 102 - 103) presents three different manners in which the requirements may be represented: natural language, graphical representation or mathematical notation. Natural language is of course normal written text that is easily understood by most. Graphical representation implies the use of some form of model or diagram to illustrate the requirements in a somewhat technical but still intuitive manner. Mathematical notation is commonly utilised in defining critical functions like security critical parts of the system, and they provide a very precise definition that is however harder to understand.

3.2 Database Design

Utilising a properly designed and managed database system will allow the end-users quick and easy access to ever-changing and integrated data, giving them a view of the big picture at their organisation. This, along with the fact that it also helps reduce inconsistency by eliminating redundant and repeated data, results in better quality information that is more readily available, increasing productivity and improving decision making. (Rob, Coronel & Crockett, p. 8-9)

At the heart of such a database system lies proper database design, which lays the bedrock on which the database stands tall if done properly, or crumbles if not. According to Rob et al. (p. 11) proper design entails identifying the expected use of the database: whether it will emphasize transactions or data storage, whether it will be a centralized, single-user or a distributed, multi-user one. To help with creating a reliable database

design there are a number of data or database models, which are usually graphical simple representations of data structures (entities), their characteristics (attributes), relationships, constraints (domains) and transformations. Rob et al. (p. 33) mention that such a data model may even facilitate the understanding of the organisation itself by quoting a client:

> "I created this business, I worked with this business for years, and this is the first time I've really understood how all the pieces really fit together."

3.2.1 Data model terms

Entities are any kind of object or thing about which data is to be stored, be it something physical like a person or abstract like an event. Attributes are characteristics that describe an entity, and represent the actual data that is stored. Between entities there can exist different types of relationships: one-to-one, one-to-many or many-to-many. Each employee being assigned a single workstation is a simple one-to-one (1:1), an employee being assigned many jobs is a one-to-many (1:*) and having several employees be responsible for several locations is a many-to-many (*:*) relationship. Both attributes and constraints may have certain constraints set on them, to ensure integrity: an employee's name must be supplied and an employee may only be assigned a single workstation at a time. These four form the base of any data model.

Entities, attributes and constraints are also sometimes referred to with the somewhat less abstract terms table, field or column, and domain respectively. These, along with the terms tuple, record and row, actually identify the essential logical terms associated with database tables. A tuple, record or row are the terms used for a single entity occurrence, i.e. an entry in a table, and to each record applies the attributes associated with the table. The domain of the attributes defines what kind of values are permitted as well as what constraints applies to them, e.g. an employee's date of birth could be limited to a date between 70 years in the past to 18 years in the past.

3.2.2 Business rules

To gain understanding of what type of data an organisation is interested in, how it is used and in what timeframe, as well as to represent that information in an unambiguous way, Rob et al. (p. 35) points out the necessity of so called business rules. These are short, precise and easy to understand descriptions of some procedure of any form within an organisation. These rules are then used to identify relevant entities, their attributes, relationships and constraints. Some sources to discover business rules are consulting managers, written documentation, end-users and checking the procedures themselves. Interviews are probably the fastest way to gather a basis for rules from end-users, but Rob et al. warns that information gained this way needs to be verified and double checked since it's very dependent on specific users' own perception of things. Here are a few examples of business rules:

- A customer may generate many work orders
- An employee may be assigned several work orders
- A customer may have several locations of operation

Business rules are most helpful in defining entities and relationships for the data model, and in general nouns in the rules translate into entities and verbs connecting nouns transform into relationships between entities. Rules that are specific enough also help identify the type of relationship in question. From the examples above we may arrive at the following entities: Customer, Employee, Work order and Location; and even identify some relationships between them, there is for example a one-to-many 'be assigned' relationship between employee and work orders

3.2.3 The entity relationship model

The entity relationship model (ERM) expands upon the relational data model, which is essentially characterized by relational tables, with columns and rows. One requirement set on an entity in the relational model is that it should contain an attribute, or combination of attributes, that uniquely identifies each tuple, called a primary key. The ERM, which is nowadays widely accepted as a standard for data modelling (Rob et al. p. 43),

provides a standard for representing entities and their attributes and relationships in a graphical diagram, referred to as an entity relationship diagram (ERD). Though there initially existed a dedicated notation for such a diagram, it had its limitations, and as a result of the growing popularity of the flexible UML its class diagram notation has now become commonplace in ERDs. UML allows for representation of the entity as a class diagram rectangle, headed by its name, with associated attributes listed within, and relationships represented by lines connecting the associated entities, headed by a label describing the relationship and the relationship type labelled at each end of the line. See Figure 2 for an example of a 1:1 relationship between two tables, showing only some simplified attributes.

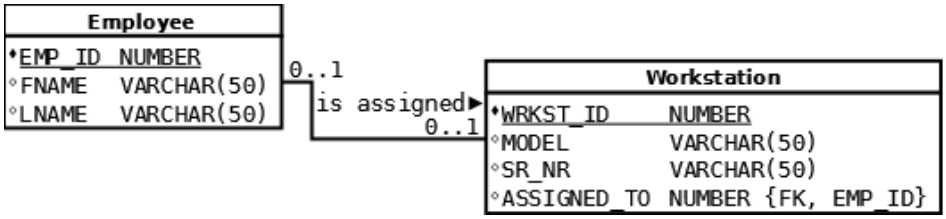

Figure 2.Example of ERD of two simplified tables

4 ANALYSIS

This chapter provides a closer look at the requirements gathering process, describing in some detail the existing work order tool, how it is performing adequately and where it is lacking. There is also a brief description of how pertinent information was identified through informal interviews and ethnographic studies. First there will however be a quick overview of the work process when handling work orders.

4.1 The work order procedure

There are in essence two different ways for the employees to get started on a job. They can either receive a new order from a customer or depending on the contract, if the validity of the verifications are about to expire, they can just start a new job on the devices and equipment at a customer's location. In the current process this responsibility falls

solely on the employee, because they have to access a database to see if any devices that are their responsibility have, or are about to, expire. They identify a device under their responsibility partly by it being assigned to the employee's personal ID number last time it was verified, and partly by what kind of device it is as well as where it is located geographically. This latter part is also how a new job gets assigned to a qualified inspector, as not everyone has the proper qualifications for every kind of device.

If the work order came from a customer, the employee simply starts with an empty work order document and fills it all in manually. If it is a periodical verification the employee can use a custom database tool to generate a partially filled in work order document, based on the data stored when the devices or equipment were last verified. It is possible to generate a batch of such pre-rendered files in case the employee won't have internet access at the time of performing the job. After performing the job itself, the employee then simply has to check that all the given information is correct and up-to-date.

Finally the employee submits a copy of the work order form to the accounting department, who handle the actual invoicing, and generates a paper or digital (in PDF format) copy for the customer. This copy the customer receives is either a simple copy of the work order, with invoicing details visible, or, in the case of performing a verification, a protocol providing some additional information required by law.

4.2 The work order form

The current work order tool is an MS Excel template file that mainly contains a form to be filled out (see Appendix 2 for such a form), and some automation to adapt the labels and contents of this form, facilitating the employee's work process. The form itself is then stripped of superfluous information, slightly reformatted and printed as a sort of indication of work performed for the customer. There are four basic different sets of information the user has to provide information on: customer, product or service, device and some miscellaneous info; and in verification cases there is an additional set of information. Below is a description of each set of information supplied for the form as

well as some of the automation, along with wishes or suggestions made for improvement in each case.

4.2.1 Customer information

There is an important distinction made between two sets of customer information: the paying customer and the customer in possession of the device. For brevity's sake to distinguish between these the first shall be referred to as the payer and the latter as the proprietor, even if that might be slightly misleading. In some cases the payer might operate at several locations, in which case the proprietor and the payer is the same but the address details are different, in other cases the proprietor might be a subsidiary or business partner of the payer, for example when the payer is a manufacturer and the proprietor is a user of their products. The address of both of these is needed, and additionally payment and business info for the payer. There are a few payers whose information is stored in the document, and is filled into the form automatically when the customer's alphanumeric ID is entered by the user. For efficiency purposes and to avoid data inconsistencies a suggested improvement is being able to simply enter any previously encountered payer's alphanumerical ID and then automatically being supplied with associated address, billing and business information, as well as a choice of known proprietors.

4.2.2 Service information

The service information provided is in the form of numerical IDs corresponding to certain services provided by the company, and their individual prices, as well as specifying how many of each service the payer is to be billed, this includes additional fees such as travelling and lodging expenses. Since these service IDs are closely tied to the type of device and form of inspection (i.e. verification or calibration), an employee put forth the wish of an automatic function that would be supplied with the inspection form and the device type to identify what services are being provided.

4.2.3 Device information

In general the information of each device is input on a single row in a simple table with 7 columns of data. The data differs somewhat depending on what branch of business the work concerns, determined by the business type of the payer, but some common information is the serial number and model of the device. The most major possible change here would be one of a data structural nature: currently each device has the same number of attributes stored, even if, for traceability purposes some devices could do with a few more attributes, and other devices don't even need as many as currently. Figure 3 illustrates how certain fieldnames of input fields might differ in three different cases according to the type of device inspected. Based on the figure it is evident that one input field might be used for totally different data types: the value provided for Interval will for example always be a decimal number while Quality and Fuel Quality are mostly text values, all stored in the same field in the database. Worth mentioning is also that the units of the numerical values in the first case are not stored in the database, meaning that the inspector has to instinctively deduce the unit when viewing old data.

Model	TEC number	Interval	Max. load	OIML class	Info
Identification	(VJ / EU)	g	kg		
Model	TEC number	Fuel			Info
Identification	(VJ / EU)	Quality	Amount	Field no:	
Model	TEC number			Vehicle	Info
Identification	(VJ / EU)	Quality	Flow(Q)	reg. no:	

Figure 3 Illustrates how the same fields in the work order form are used for completely different types of values.

4.2.4 Miscellaneous information

The miscellaneous information provided is related to the employee, the contact person of the customer or some detail of the job, e.g. the numerical codes of payment cards provided for testing purposes. Concerning this only a few improvement possibilities have been identified, such as the contact person being automatically associated with the customer and the detail likewise being tied to the customer and employee.

4.2.5 Verification information

When it comes to verifications, it is required by law that some additional specific information is readily available at the proprietor. This information includes but is not limited to the work instructions applied while performing the job, the equipment used and some environmental factors. The work instructions could again be automatically defined based on the types of devices in the work order, since each type has its own work instruction. An employee suggested that providing the used equipment could be a simple choice from a list of all available equipment.

5 REQUIREMENTS

This chapter contains a description of how the SRS was structured, how it conforms to the IEEE 830 standard and how it differs. There is also a specification of what structural models were used to represent different requirements.

5.1 The system requirements specification

As stated earlier, the SRS is structured according to the IEEE 830-1998 standard (appendix 1) with slight modifications. This along with the following few paragraphs all describe a single chapter or section of the SRS each. The introduction is in many ways similar to the one in this paper, with the addition of a reference to the work instruction from which some requirements are derived.

Chapter two of the SRS is structured more freely to give a short presentation of the Salesforce environment, and to describe the intended users and the training they might need to receive in order to efficiently transit over to utilising the Salesforce application. This signifies that neither the system perspective nor the general constraints are explored widely. There is however a specification in natural language of some essential user requirements.

The third chapter consists of system requirements, including both functional and non-functional requirements. These are mostly quite low level detailed descriptions of what kind of data input is needed for the work order form and subsequent invoicing and traceability purposes, as well as constraints placed on that data. For the sake of comprehensibility the requirements are first presented exclusively in a natural language, and the structure of this initial section itself being such that the requirements are sequenced in the same order a work order form would typically be filled in. The section following that is a more graphical representation with UML diagrams mixed in with the text, and is described in the section below titled *Requirements ranking and models*.

The appendix only contains a single page of the law detailing what is required to be included in the verification document that is presented to the customer. Rather than having an index at the end of the document, there is a table of contents at the beginning.

5.2 Requirements ranking and models

In the latter part of the third chapter of the SRS the requirements are listed from absolutely necessary to less important. There are three different orders of ranking which could be said to correspond to (1) things that must be included, (2) things that should be included and (3) things that could be included. The first rank contains most of the input fields required since they are needed for invoicing and traceability purposes, which is the whole point of the work order form to begin with. The second rank consists of mostly functions and operations that without which there is not much point with migrating to new system. These are for example operations that enable the customer to be a part of the process and functions that reduce the risk of inconsistency and duplicate data. The third rank does not contain a whole lot of requirements, only suggested improvements that would make filling the work order form simpler or make the whole process a bit faster in any way. A concrete example of this is automatically sending a customer a message when an invoice has been generated.

In addition to being listed according to their rank some requirements are also represented in UML diagrams, e.g. a part of the work order process. A work order can be generated in a few different ways, with several different actors (managers, customers and employees) who interact with each other and the system to reach a point where the employee can get started on his job and fill in a work order form. There are as such a few sequence diagrams in the SRS detailing these processes, for example the diagram in Figure 4 illustrates the process of a customer generating a new work order in the system. In the diagram the actors and objects interacting are the customer, the system, a details form the customer needs to fill in and the database where the work order will be generated.

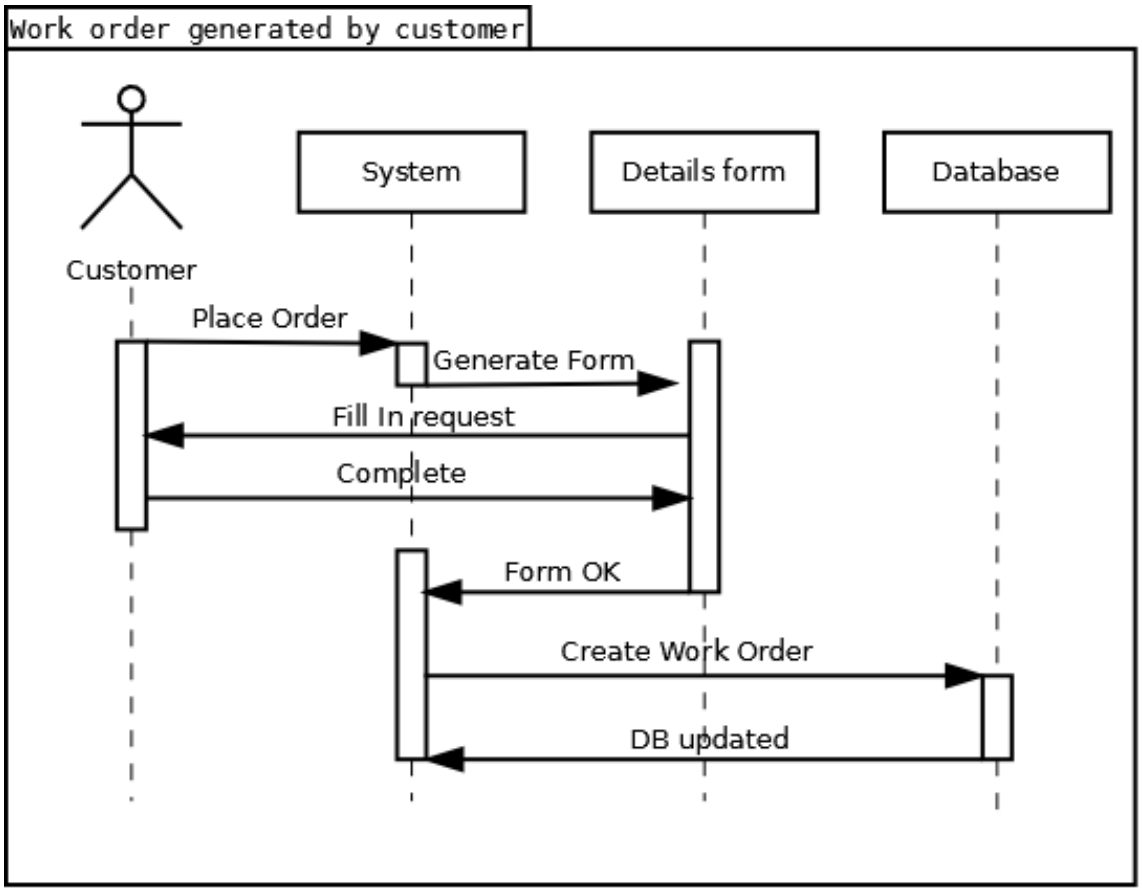

Figure 4 Sequence diagram of a customer generating a work order

The difference in the work process between a calibration and a verification must also be taken into account. The work order form will have to function slightly differently depending on which type of job is being performed. For a calibration the only required information on the device is serial number and model. In the case of a verification, on

the other hand, many of the attributes of the device as well as the results of the inspection are required for traceability. Furthermore, it is by law required that these appear in a verification document or protocol along with some other information such as the work instructions and equipment used for the job. If a single work order only contains calibrations or verifications exclusively, it is quite clear how to proceed, however if even a single verification is included in a job consisting otherwise of solely calibrations, a verification protocol has to be generated. The protocol may exclude information on the calibrated devices, but to limit the complexity of implementation it is suggested to simply include all devices in the verification protocol as long as calibrated devices are marked as such.

Some of the so called extra information needed for verifications is not necessarily needed for traceability and does not have to be stored separately as data in any form or way, but can rather just be visible in the document that is linked to the work order. The work order document itself however does not need to be stored as a document linked to the invoice, as all the information in it is readily available as raw data, thereby not wasting storage space. Considering that this means the document would always have to be generated anew each time anyone wanted a copy of it, brings to question whether it is a viable solution. Storing it as an actual document has the added benefit of the old data always being available, as a name change in a company or a price change would affect old data unless it was stored as hard values in the database itself. As this thesis does not aim to specify how to implement these requirements, both possibilities are presented in the SRS, along with the benefits and limitations that follow. The difference in database structure that each solution would entail is also part of the database design produced, which will be presented in the next chapter.

6 DATABASE DESIGN

This chapter will contain descriptions of how a basic database design was constructed: how the business rules of the database were acquired, how they were assembled into a database dictionary and a schema.

6.1 Business rules

Since business rules are in essence fairly similar to system requirements, especially functional user requirements, it was a rather simple process of identifying most rules from the finished requirements documentation. Some system requirements also translate quite well into business rules, but since non-functional requirements are mostly detailed technicalities of or constraints placed on the system they are not that easily applied as rules, except as being somewhat useful in recognizing limits on entity relationships.

Some examples of the defined business rules are:

- A customer may place several work orders
- The system may generate periodical work orders
- A work order is assigned to an inspector (by inspector or manager)
- A customer may operate at several locations or have several proprietors
- A device is located at a proprietor

From these rules five entities are easily recognisable: customer, employee, work order, proprietor and device. Their relationships are also at least partially distinguishable, in so far as that they are defined in one direction. A work order may for example be assigned to only one employee, but on the other hand, an employee may be assigned several work orders simultaneously. For clarity's sake, such rules are defined separately so as to not mark them erroneously in the database design.

6.2 Data dictionary

Most of the attributes for the entities derived from the business rules are quite intuitive and obvious in their necessity, like name and address for a customer, but some have to be determined by a closer analysis of the business rules and how the entities are to relate to each other and to the functions of the system itself. An example of this is, since the system is supposed to generate work orders when the validity of a verification is about to expire, it needs to know when that is, i.e. the device needs to have an expiration date. The entities, their attributes and constraints, as well as relationships represented by primary and foreign keys are all represented in an exhaustive data dictionary.

The dictionary structure used is based upon a model presented by Rob et. al. (p. 84) and is most efficiently used in junction with the creation of the database tables, as it provides an excellent overview of table and attribute names as well as properties. The datatype of each attribute is defined, and when relevant so are the format and the actual domain. See Table 3 for an example of a table in the data dictionary. The example is of the Proprietor table, whose structure and definitions are examined in the next paragraph.

Table 3 Data dictionary entry for Proprietor table

Attribute	Contents	Data Type	Format	Domain	Required (default)	Key	FK Reference
PropID	Automatically generated ID	Integer	#####	10000-99999	Y	PK	
StreetAddress	Address of property	Text(30)			Y		
PostCode	Postal code for location	Char(5)	#####	00001-99999	Y		
PostOffice	City/Region of location	Text(25)			Y		
VATIN	Value added tax identification number	Text(15)			Y	K	
CompanyID	ID of company	Integer	#####	10000-99999	Y	FK	Company

The attribute names are defined as descriptively as possible without exceeding about 15 characters in length, written in CamelCase to improve readability, after which follows a brief description of the attribute. The data types at this stage are defined in quite a general manner, i.e. a way that is easily understood by most, even without any previous database knowledge. The exception to this might be the Char() datatype, which simply stands for character, with the number of characters required within the parentheses. In

Table 3 an example of this is the PostCode attribute, which refers to the postal code of where the company is situated, and which is defined is char(5) meaning it must always be 5 characters long. The attribute is further formatted as only numbers (represented by the # symbol) in the domain 00001-99999. The dictionary also shows whether an attribute is required (Y) or not (N), essentially disallowing or allowing null values in that field. Some attributes that are required do not have to be specified by the user though as they have so called default values, included in the dictionary within brackets in the same column. Finally is specified whether or not the attribute acts a primary key (PK), foreign key (FK) or just a candidate key (K). The table a foreign key relates to is specified in the final column. A candidate key is an attribute that contains unique values for all entries in the table, and is as such a simple alternative attribute to identify a specific entry.

Some special constraints of a few attributes are also represented in the data dictionary in a particular way. Text fields whose length may vary are defined as Text() where the brackets contain the maximum number of characters allowed in the values of said field, such text fields are for example the VATIN field which contains a code whose length may vary from 8 to 15 characters. Another special field is the Email field whose values need to be validated to only contain emails consisting of alphanumerical characters and certain special symbols, followed by an @ symbol and a domain. A very specific manner in which an attribute's allowed values are constrained is by specifying a list of allowed values. An example of this is in the representation of the numerous device types defined as separate tables, where a parent table contains an attribute (DeviceType) describing what particular child table a device belongs to, as illustrated in Table 4. This attribute only allows values corresponding to one of the child tables such as: Scale, Weight or Fuel Dispenser.

Table 4. Data dictionary entry for VerDevice table

Attribute	Contents	Data Type	Format	Domain	Required (default)	Key	FK Reference
DeviceID	ID number of device	Integer	#####	10000-99999	Y	PK, FK	Device
DeviceType	The type/sort/category of the device	Text(20)	List	{Weight, Scale, FuelDispenser, AlcoholDispenser, Dimension, Truck, Other}	Y		
TUKES	2 char code for inspection results	Char(2)	List	{VE,VM,VV,HM,HE,}	Y		

The reason for most of these attributes being defined and described in such general terms instead of more technical or system-specific terms is to allow for future implementation of the database structure in other systems, without having to revert back to such a basic stage. The aforementioned validation of an E-mail address might for example be something included in the actual database management system itself or it might be something that has to be implemented in the application. To only allow values from a list might likewise be implemented in the database, by adding a table containing only the permitted values, or by having the application only allow values from an array of values.

The data dictionary allows for a necessary and useful overview of the tables, attributes and their properties, but does not do a good job of illustrating how all the tables are related. For the purpose of providing a graphical representation of the tables and their relationships a UML diagram, as presented in the next section, does a much better job.

6.3 Schema

The database schema provides a logical overview of the structure of the database, the tables it contains, the relationships between them and the multiplicities and constraints of the relationships. A couple different variations of the schema were created to provide different levels of details and clarity: one version with only table names and relationships visible and another with attribute names and datatypes as well. Figure 5 is a sort of incomplete hybrid version ERD of these two, where most attributes have been omitted, except for primary and foreign keys.

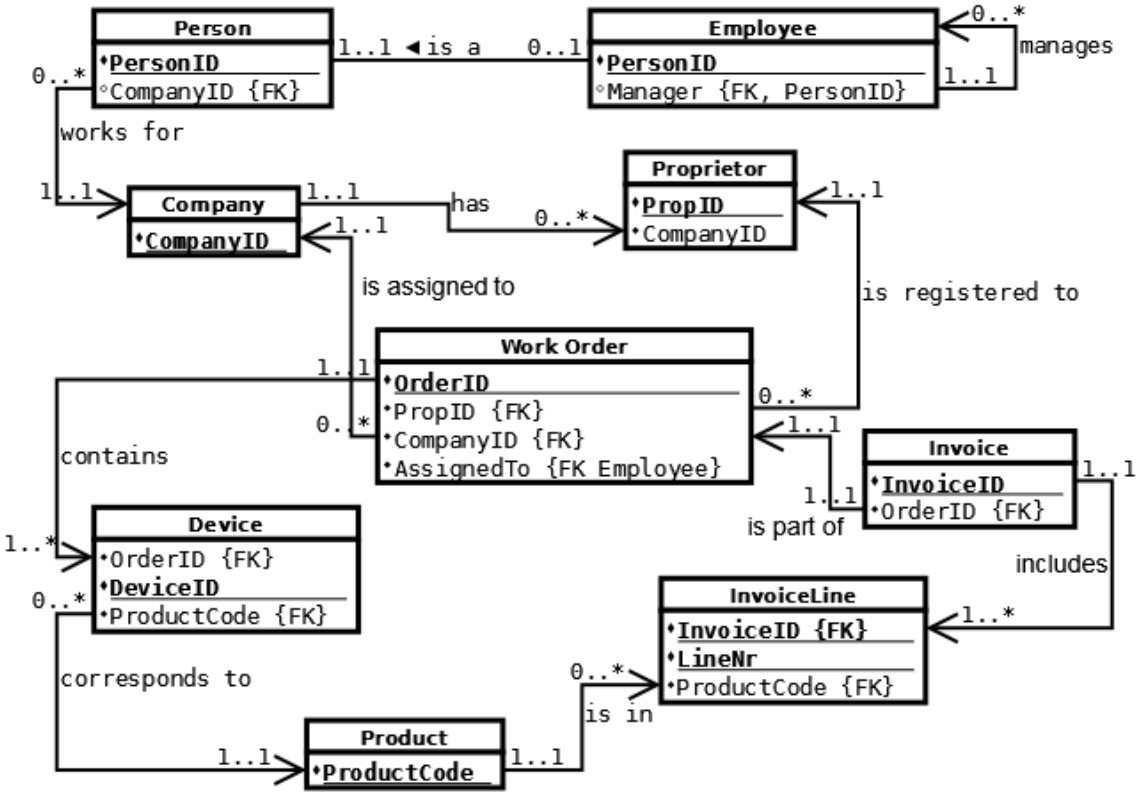

Figure 5 Simplified ERD of most tables included in the database design

The figure illustrates entity relationships as simple lines with arrows and the multiplicities at each end. The multiplicities are relatively simple where the numbers at one end of the line correspond to how entries in that table are related to entries in the table at the other end. In other words, with the 1:* relationship between the Person and Company tables as an example, a Person *works for* exactly one Company and a Company has none or any amount of Persons working for it. The Company table in this design contains both Inspecta itself, its business partners and customers, which is why a company may ostensibly appear to have no employees; for the simple reason that none have been registered in the system.

As is evident from Figure 5, most relationships are of the straightforward 1:* type, though there are a couple of necessary 1:1 relations, like between Person and Employee. This is a case of inheritance, where an Employee record shares a lot of attributes with a

Person record, both have names, a phone number and an employer for example, but an employee however has some additional information like an employee ID and a job title. In other words, all employees are persons, but not all persons are employees. Another interesting aspect of the Employee table is that it contains a recursive relationship, since some employees are managers, who supervise a number of employees. Managers have not been identified to have any additional attributes compared to other employees, which makes the recursive design feasible. Would a manager have such attributes another Manager subtype of the Employees table could be defined.

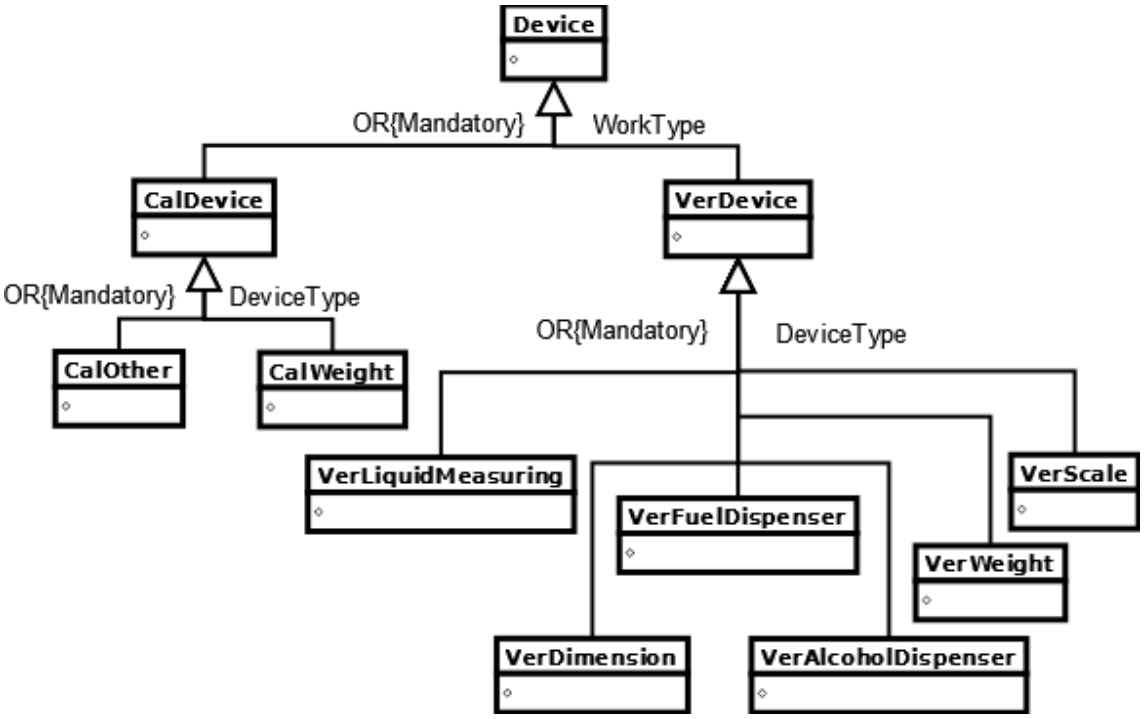

Figure 6.Device subtypes specialisation hierarchy

Another set of subtypes not depicted in Figure 5 is the multitude of different types of devices that are either verified or calibrated. These are presented in a specialisation hierarchy diagram with the Device entity as a supertype in Figure 6. This diagram is again of a simplified nature, where no attributes are listed. The WorkType label in the diagram refers to the attribute in the Device table and is set to either CAL or VER according to the job performed, which then determines in which table, CalDevice or VerDevice, additional information is stored. The neighbouring label signifies that a device can

only be either calibrated OR verified, not both simultaneously, and the mandatory notation refers to an entry in the Device table has to be included in one of the subtypes. The same of course goes for any further subtypes. Due to inheritance the subtypes furthest down the line still inherit all attributes and even relationships of the supertype entity Device.

7 CONCLUSION

For the purpose of defining the requirements of migrating the existing work order tool to a Salesforce environment the solutions presented in this thesis are both satisfactory in some aspects and lacking in others. Considering the goal of creating a requirements document, the results are likewise mixed: a detailed SRS does indeed exist, containing all identified requirements presented both in text and graphically. It is however likely, since no implementation effort have been made at this point that problems arising at the next development stage will necessitate at least a partial revision of the document, filling in details as needed. So the requirements of creating a similar, but improved, work order tool are defined, but the solutions fail to touch on any kind of special needs or properties of the new environment, raising the question of why to mention Salesforce at all.

The SRS did however prove a considerable contribution in creating a comprehensive database design, which was the second of the initial goals defined. As the design includes both an exhaustive data dictionary and an ERD providing a clear overview of entity relationships it should be of considerable help in implementing the database itself. Though since this thesis was defined as not to include any further steps of implementation, there is indeed no description of how to bring the design into the Salesforce database structure, which would be the next phase in development. Without going into too much detail on the subject, some simple suggestions on how to move forwards with the implementation are presented in the next few paragraphs.

The Salesforce platform comes with some standard entities readily available. These entities, or objects as they are referred to, could be utilised efficiently in the implementation of the design. As mentioned, there is for example an Account object whose attributes include almost all of the attributes required for the Company entity. These kinds of corresponding entities exist for at least the following entities defined in the design presented here: Person, Work Order and Product. Even if all these objects do not contain all the necessary attributes, they can easily be implemented as custom attributes. Likewise, custom objects can be created for the entities without similar standard counterparts, e.g. the Invoice and Device tables.

Some of the attributes whose data types are normally slightly more complex to implement in a database system, are not such in Salesforce. There are for example readily available data types for special formats such as email addresses or phone numbers, whose values are always validated. Another such data type is the so called Picklist, where the developer can define a list of allowed values for that attribute. A practical use of this feature is to define a so called Global Picklist, where the same list can be used in several different tables, only needing to update the allowed values in one place, when needed.

The final part to consider for implementation is of course the work order form, which could basically be the simple creation or editing of an entry in the Work Order table, i.e. the actions of adding a new or editing a work order would be equivalent to filling out the form. This is at least partially readily available in Salesforce, as the action of adding or editing a record brings up a form-like page with all the fields to be filled in. The challenge here would be to figure out how to be able to provide data for several entities from a single form page.

These are just a few of the problematic situations to solve, as how to implement the requirement of an employee being able to fill in the work order form while being offline and still having the customer information available might be the greatest challenge to overcome. I believe however that this system, though complex and time-consuming to implement will have huge beneficial consequences. Partly it will eliminate data inconsistencies that existed before, significantly decreasing the occurrence of data duplicity

as well as bringing the work process of the whole organisation into a single unified environment. It will also allow for closer collaboration with customers, as they can also be allowed to be an active part of the process, being automatically informed of when jobs are performed. Furthermore, utilising the other applications provided by Salesforce could improve on other aspects of the organisation. The Analysis service might for example be used to analyse sales and other data to present an overview of any analysable aspect of the organisation.

REFERENCES

Immo, Salo. 2010, *Cloud computing – palvelut verkossa*, Jyväskylä: WSOYpro OY, 168 p.

Rob, Peter; Coronel, Carlos & Crocket, Keeley. 2008, *Database Systems: Design, Implementation & Management*, international edition, London: Cengage Learning EMEA, 808 p.

Salesforce Developer. 2016, Salesforce.com, Inc. [online] Available: https://developer.salesforce.com. 25.4.2016

Salesforce Help. 2015, Salesforce.com, Inc. [online] Available: https://help.salesforce.com/home. 25.4.2016

Sommerville, Ian. 2004, *Software Engineering*, 7th ed., Harlow: Pearson Education Limited, 754 p.

Wohlin, Claes. 2005, *Programvaruutveckling*, Lund: Studentlitteratur, 226 p.

APPENDIX 1 IEEE 830-1998 STANDARD

REQUIREMENTS DOCUMENTATION STRUCTURE SUGGESTION AS PRESENTED BY SOMMERVILLE, IAN

1 Introduction

1.1 Purpose of the requirements document

1.2 Scope of the product

1.3 Definitions, acronyms and abbreviations

1.4 References

1.5 Overview of the remainder of the document

2 General description

2.1 Product perspective

2.2 Product functions

2.3 User characteristics

2.4 General constraints

2.5 Assumptions and dependencies

3. Specific requirements cover functional, non-functional and interface requirements. This is obviously the most substantial part of the document but because of the wide variability in organisational practice, it is not appropriate to define a standard structure for this section. The requirements may document external interfaces, describe system functionality and performance, specify logical database requirements, design constraints, emergent system properties and quality characteristics.

4. Appendix

5. Index

APPENDIX 2 BLANK WORK ORDER FORM

Work order

2052-16-xxx

Date

Inspecta Tarkastus Oy

Customer

Customer		
Precision		
Address		
Post office	Chain store	
Payer	Customer number	
Precision	Cost pool	
Address/operator	Customer group **1**	
Post office	Business area **1**	

will be sent to the e-mail copy

Performer of measuring

Mauri Nyqvist
+35850412 9700
2052

Invoicing VAT- number

Product	Amount	Unit price		Sum
210501				
210502				
210503				
210504				
210505				
210506				
210803				
210804				
8011				
8012				
801A				

Sum free of tax 0,00 €

Invoice information

Ref. Contactperson for invoice

Phone

or job / order number

V	verified, meets the requirements
H	Rejected, requirements not met
Vv	adjustment during verification

mpe= max.permissible error

The invoice will be sent separately

E	seal intact
M	seal broken
HS	service seal
EV	the first verification

Traceability of the measurements

Inspecta Tarkastus Oy takes care of the traceability of their measuring instruments corresponding to the national normals.
For each undertaking Inspecta's general terms of sales and contract TC-GSCTC-FI-WW_091218 are observed
We provide you with a report of the used measurement normals if required.

Measuring instrument

Customer contact
Phone
Other information

Serial number	Model Identification	TEC number (VJ / EU)	Interval g	Max. load kg	OIML class Info	

Measuring results

Measured error (g)

Error Adjusted

Muutoksenhaku: Jos olette tyytymätön tarkastuspäätökseen, voitte hakea siihen kirjallisesti oikaisua Inspecta Tarkastus Oy:ltä.
Hakuohjeet: www.inspecta.fi.

APPENDIX 3 SUMMARY IN SWEDISH

INLEDNING

Min arbetsgivare, Inspecta Oy, är som bäst i färd att ta i bruk en ny användarplattform, Salesforce, främst för att hantera arbetsorder. Salesforce är ett moln baserat kund hanterings system, som via en webbapplikation erbjuder effektiva funktioner för att hantera kunddata, för arbetstagare att följa upp sina uppgifter m.m. Inspecta erbjuder tjänster som besiktning/granskning, provning och certifiering. Avdelning vars verktyg behandlas i detta arbete har främst hand om granskning av olika slags mätutrustning. Granskningarna kan vara olika slag: om de utförs endast för kundens behov till den noggrannhet kunden begärt kallas det en kalibrering, om det å andra sidan är frågan om utrustning som används i direkt samband till försäljning av något baseras kraven direkt på lagen och det kallas då en verifikation av överensstämmelse.

Syftet med detta examensarbete är att identifiera hurdan information samt automation är nödvändiga för att förverkliga arbetsorderverktygen på den nya plattformen. Detta innebär krav på ett formulär för arbetstagarna att fylla i, ett intyg över utfört arbete som skall finnas hos kunden samt strukturen på en databas för att lagra all information. Målet är att presentera dessa krav i ett kravdokument samt att åskådliggöra databasens struktur i en databasdesign. De existerande verktygen och databasen analyseras för att samla in krav utgående från vad som fungerar bra och vad som kunde förbättras, samt även från förslag på förbättringar från arbetstagare.

MOLNTJÄNSTER OCH SALESFORCE

Salesforce erbjuder en mängd olika tjänster för olika organisatoriska processer som molntjänster. Molntjänster är enligt Salo (2010, s. 16) svåra att definiera på något enhetligt sätt, men i princip är det frågan att erbjuda traditionella lösningar, som hårdvara, server miljöer eller licenserad programvara som tjänster över en internetförbindelse.

Salesforce började som en erbjudare av en sådan molntjänst i form av ett CRM, men har sedan övergått till att erbjuda ett flertal olika typer tjänster på sin online plattform, som marknadsförings- och analystjänster. Alla köpta tjänster finns tillgängliga från en enkel

webapplikation där såväl administratörer som vanliga arbetstagare kan sköta sina dagliga uppgifter.

Salesforce erbjuder även en inbyggd databas som kan presenteras till användarna som en traditionell databas med tabeller, kolumner och rader, men behandlas av det bakomliggande systemet snarare som objekt av instanser med olika egenskaper och förhållanden. Denna objektorienterade struktur möjliggör enligt Salesforce ett flertal datatyper som i en traditionell databas inte är lika lätta att förverkliga.

Dess utöver är det möjligt att tillämpa Salesforce miljön till just det egna behovet, då företag kan lägga till sina egna objekt typer med tillämpade egenskaper som möter just de krav och behov i den egna organisationen. På den bakomliggande plattformen, Force.com, är det även möjligt för företag att bygga upp sina egna tilläggsmoduler för att skapa ytterligare tillämpad funktionalitet.

METODER

Kravhantering

Enligt Sommerville (2004, s. 64) är programvaruspecifikationen det första steget i utveckling av programvara, varefter följer design processen. Ibland kan det vara svårt att särskilja på dessa men Wohlin (2005, s.95) menar att specifikationen är "vad" som skall göras och designen svarar på "hur" det kan förverkligas. Specifikationen inleder med att klarlägga huruvida den föreslagna lösningen är möjlig och föredelaktigt genomförbar. Denna process är vid examensarbetets inledande redan genomförd så arbetet behandlar främst därpå följande två steg: att klarlägga och analysera systemkrav samt att klassificera och presentera dessa i ett kravdokument.

Wohlin (s. 95-110) och Sommerwille (s. 115-238) behandlar ett flertal metoder för att klarlägga, klassificera och presentera krav, för att slutligen samla de i kravdokumentet. De nämner klarläggningsmetoder så som dokumentanalyser, intervjuer, användningsfall, olika scenarier samt etnografiska studier. Klassificeringen går i princip ut på att dela upp kraven enligt vad kraven ställs på och hur pass detaljerade de är: beskriver de

en transaktion mellan systemet och användarna eller en specifik funktion i systemet, samt hurdana begränsningar det läggs på kraven. Kravdokumentet bör innehålla alla klarlagda krav samt även en inledande beskrivning över situationen och bakgrunden till behovet på det nya systemet.

Databasdesign

Databasdesignen är viktig för att konstruera en databas utan motsägelser och överflödig data, vilket resulterar i bättre information som är enkelt och snabbt åtkomlig (Rob, Coronel & Crockett, s. 8-9). För att skapa en sådan ordentlig design finns det olika databasmodeller: oftast grafiska representationer av datastrukturen för tabeller, dess egenskaper och förhållanden till andra tabeller.

För att effektivt och pålitligt skapa dessa modeller, bör man enligt Rob et. al. utgå från så kallade affärsregler (business rules), som är korta, exakta och lättförståeliga beskrivningar av någon process i organisationen. Dessa affärsregler används sedan för att identifiera tabeller, dess egenskaper och förhållanden till andra tabeller. I regel handlar det om att översätta substantiv i affärsreglerna till tabeller i databasdesignen och verb mellan substantiv till förhållanden mellan tabellerna i fråga.

Dessa tabeller och övriga klarlagda egenskaper presenteras sedan i en så kallad begreppsmodell. En populär sådan modell är enligt Rob et. al. (s. 43) den så kallade ER-modellen (entity relationship model) som kan användas för att på ett överskådligt vis presentera tabeller samt dess egenskaper och förhållanden i olika diagram, oftast gjorda i UML (unified modelling language).

Analys

Analysen går ut på att genom att kolla på arbetsorderprocessen och det egentliga verktyget främst identifiera brister med det gamla systemet, för att ställa upp dem som krav för att det nya systemet. Somliga krav har även identifierats direkt från förslag på förbättringar av arbetstagarna.

Den information som samlas in i arbetsorderformuläret kan grupperas i fem olika grupper enligt följande: kund, produkt, mätutrustning, diverse och verifiering. Möjliga förbättringar vore att exempelvis kunddata kunde fyllas i automatiskt på basis av ett simpelt val av kund. Produkt och mätutrustnings uppgifter kunde likaså bindas ihop på något vis: endera kunde typen av den inmatade mätutrustningen automatiskt avgöra vilken slags produkt det är frågan om, eller tvärtom kunde val av produkt begränsa och definiera vilket sorts uppgifter som måste matas in för mätutrustningen ifråga. Det senare förslaget innebär även att på ett effektivare vis kunna särskilja på olika slag av apparater, som alla har olika mängd av och typers egenskaper som bör anges.

RESULTAT

Kravdokument

Själva kravdokumentet skapades enligt en standard som både Wohlin och Sommerville lyft fram som en bra riktlinje: IEEE 830. Dokumentet är strukturerat ganska långt enligt standarden, med vissa modifieringar enligt behov. Den introducerande delen beskriver bakgrunden samt behovet för det nya systemet. Själva kraven är presenterade både i normal brödtexts form samt i vissa fall med stöd av olika diagram eller tabeller.

Kraven är indelade i tre olika grader av vikt. Den första klassen beskriver främst de inmatningsfält som är absolut nödvändiga för att systemet alls skall fungera ens till samma grad som tidigare, medan den andra beskriver främst de tilläggsfunktioner som skulle förbättra systemet anmärkningsvärt och eliminera motstridigheter. Den tredje klassen är främst förbättrings förslag som skulle underlätta användarnas arbetsprocess.

De krav som presenteras även i UML diagram är främst sådana som beskriver någon längre eller mer komplicerad process i systemet. Med andra ord kan det ofta ingå flera detaljerade krav i ett sådant omfattande diagram. Ett exempel på detta är processen då en kund lägger in en ny arbetsorder, då måste systemet bl.a. stöda inloggning för kunden, samt generera ett formulär som kunden kan fylla i uppgifterna i, och dessutom kunna lagra dessa uppgifter i databasen.

Databasdesign

Att övergå från kraven till databasdesignen var ett rätt så simpelt steg i och med att den inledande fasen i designen var att klarlägga affärsreglerna som relativt enkelt kunde plockas fram ur kravdokumentet. Systemkrav översattes i flera fall rakt till affärsregler medan krav som beskriver begränsningar, översattes rätt så enkelt som begränsningar även till designen. För klarhets skull definieras de flesta regler gällande förhållanden från bägge delaktiges synpunkt.

Tabellerna, dess egenskaper och egenskapernas definitioner samlades i ett så kallat databibliotek (data dictionary) som för varje egenskap eller attribut beskriver dess syfte, data typ, formatering, begränsningar samt nyckelegenskaper. I fall ett attribut används som en så kallad främmande nyckel (foreign key) definieras även tabellen som nyckeln binder sig till. På så sätt framgår även tabell förhållanden ur biblioteket.

Alla tabeller finns även representerade i ett UML diagram. Diagrammet presenterar alla tabeller som rutor, med dess attribut och datatyper listade inuti, samt förhållanden till andra tabeller representerade som linjer. Linjerna är märkta med beskrivande verb, samt förhållandetypen i vardera ända. Det skapades även simplare diagram för olika ändamål, t.ex. ett utan attribut skapades för att göra förhållandena mera åskådliga.

KÄLLFÖRTECKNING

Immo, Salo. 2010, *Cloud computing – palvelut verkossa*, Jyväskylä: WSOYpro OY, 168 p.

Rob, Peter; Coronel, Carlos & Crocket, Keeley. 2008, *Database Systems: Design, Implementation & Management*, international edition, London: Cengage Learning EMEA, 808 p.

Sommerville, Ian. 2004, *Software Engineering*, 7th ed., Harlow: Pearson Education Limited, 754 p.

Wohlin, Claes. 2005, *Programvaruutveckling*, Lund: Studentlitteratur, 226 p.