Jari Laurila

Developing Computerized Maintenance Management System

Helsinki Metropolia University of Applied Sciences

Master's Degree

Information Technology

Master's Thesis

16 December 2017

Metropolia

Author Title Number of Pages Date	Jari Laurila Developing Computerized Maintenance Management System 52 pages + 2 appendices 16 December 2017
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor	Ville Jääskeläinen, Head of Master's Program in IT

The objective of this thesis was to evaluate the applicability of the Dynamics 365 platform to the development of a Computerized Maintenance Management System (CMMS). Having a CMMS to manage maintenance operations is important to many companies and Dynamics 365 is a potential platform for creating one.

The thesis starts with background research in maintenance. Essential requirements for a CMMS are then defined and compared to the existing functionality and extendibility and customization capabilities of Dynamics 365. Based on the analysis a set of requirements is then defined for a prototype implementation.

The thesis then describes the implementation of the prototype. Information is provided on the various customizations and extensions done to the system. The application implementation consists of JavaScript front-end code and C# back-end code.

After the implementation the prototype was evaluated by conducting functional testing and a technical review. Based on the findings the applicability was demonstrated and guidance to future development given.

Dynamics 365 proved to be a powerful platform for creating business applications. However, it is a complex product that offers multiple ways of developing functionality. For the best results, developers should get familiar with the platform and use the available tools and 3rd party components.

Keywords	maintenance, CMMS, Dynamics 365, prototyping

Contents

List of Abbreviations

API	Application Programming Interface
CDN	Content Delivery Network
CMMS	Computerized Maintenance Management System
CM	Corrective Maintenance
ERP	Enterprise Resource Planning
FSM	Field Service Management
HTTP	Hypertext Transfer Protocol
IT	Information Technology
LOB	Line of Business
MTTF	Mean Time to Failure
MRO	Maintenance, Repair and Operations
PM	Preventive Maintenance
RDL	Report Definition Language
RTF	Run to Failure
SDK	Software Development Kit
SPA	Single Page Application
xRM	Anything Relationship Management

1 Introduction

The world today is heavily dependent on technology and automation. As technology advances we can do more with machines but we also become more dependent on them. In many industries the equipment required to produce goods and services is expensive to own, operate and maintain and faults even in a single piece of equipment can halt an entire production. In addition to direct production losses, the downtime caused by an equipment failure may damage company reputation and in some cases, lead to fines and penalties. Equipment failure can also jeopardize health and safety. These things make maintaining the equipment properly a key activity for companies and in some industries companies spend up to 50% of their operational costs on maintenance. In today's competitive environment, understanding the total cost of maintenance and optimizing these costs has a big impact on company profitability. (Campbell, 2016)

Today, the companies typically manage their equipment using a software product. Computerized Maintenance Management System (CMMS) are software products that help companies run their maintenance operations efficiently. With these systems, companies can plan preventive maintenance, manage corrective maintenance, track machine hours, tires and other consumables, monitor warranty periods, track labor and fuel costs and manage inventory. More advanced CMMS systems integrate also with other company functions such as project planning, resourcing, and accounting. (Equipment World)

There are many good commercial CMMS systems on the market. However, some companies have specific requirements for managing their equipment or want a solution that can be integrated to their existing IT infrastructure and other operational systems. For them, the ability to customize and further develop the system and to integrate it tightly to other business applications is important. Increasingly, companies are also looking for cloud-based solutions. Solution providers that can offer CMMS solutions that meet these criteria can gain significant competitive advantage compared to point solutions or solutions requiring on-premise installation.

Microsoft Dynamics 365 (xRM) is part of Microsoft's Dynamics 365 suite. In addition to providing out of the box business applications it is an enterprise application platform that supports many key business applications out of the box: sales, customer service, field service and project service. Currently it does not include CMMS functionality, but many

of the functionalities such as work order management are closely related to CMMS key features. Dynamics 365 (xRM) offers to independent software vendors a platform for building business applications. These applications can leverage other functionality in the platform and use the same, extensible data model. Potentially it could be a good platform for implementing a cloud-based CMMS application that meets customer requirements and enables the solution provider to offer the solution globally.

1.1 Background

Kauko Oy is a Finnish system integrator specializing in mobile knowledge work and Field Service Management solutions. The company provides digital transformation services to customers using Dynamics 365. Based on the identified need, the company wishes to explore the possibility of adding a CMMS solution into their offering, either as a packaged software solution or as a solution accelerator that cuts development time and cost.

1.2 Objectives and Scope

The objective of this thesis is to evaluate the applicability of the Dynamics 365 (xRM) platform to the development of a Computerized Maintenance Management System. The thesis aims to answer the following research question:

- How can a CMMS be implemented using Dynamics 365 (xRM)?

The output of this thesis is not a complete software product but rather a functional prototype that can that can be used to demonstrate and evaluate how such a product can be implemented. Based on the findings recommendations for future development can then be given. The goal is also to be able to use the prototype as a foundation for the first customer implementation.

1.3 Method and Process

This study was conducted as follows:

1. The study started with a literature review of the existing knowledge related to maintenance and Computerized Maintenance Management Systems. It defined the key concepts and the motivation for implementing CMMSs.

2. Microsoft Dynamics 365 platform and the out of the box functionality and application platform capability were also investigated.

3. Relevant Dynamics 365 development technologies were examined.

4. Based on the information discovered, a fit-gap analysis was then made to see what functionality is missing from Dynamics 365 and features were then selected for prototype implementation.

5. Based on the analysis, the study designed a prototype implementation of CMMS.

6. Finally, the study evaluated the design. Evaluation was done by doing functional testing and by conducting a technical review. Based on the findings the design can be developed further.

The fit-gap analysis method that was used in step 4. It is a methodology for identifying where a system fits or does not fit the stated needs. It can be used in many business situations, frequently it is used in selecting new software solutions. The fit-gap analysis is performed by comparing the functionality of the planned system to the current business practice and identifying where the system fits the requirements and where there are gaps. The fit-gap analysis then allows to understand why the gaps occur, what is needed to solve them and allows prioritization of the solution (Infotivity, 2017)

2 Maintenance Fundamentals

This section first introduces the impact of maintenance and then gives definitions to different maintenance types. Finally, the section explains why there is a need for computerized solutions for maintenance management and introduces Computerized Maintenance Management Systems.

2.1 Impact of Maintenance

Maintenance costs are a major part of operating costs for many industries. Depending on the industry they can be up to 50% of total production costs. Additionally, downtime caused by poor maintenance can cause even bigger cost due to rework, rejected products, fines and damage to company reputation. Especially in capital-intensive businesses, the profitability of the business is tightly related to proper maintenance. For these reasons, it is important for companies to have an optimized maintenance management strategy that is both cost effective and ensures reliable operation of the equipment (Campbell, 2016)

2.2 Maintenance Types

According to the EN 13306:2001 standard, maintenance is defined as a combination of all technical, administrative and managerial actions during the life cycle of an item intended to retain it in, or restore it to, a state in which it can perform the required function. The standard divides maintenance types into two major categories: preventive maintenance and corrective maintenance. Preventive maintenance is further divided into condition-based maintenance and predetermined maintenance. The relationships between maintenance types are shown in Figure 1.

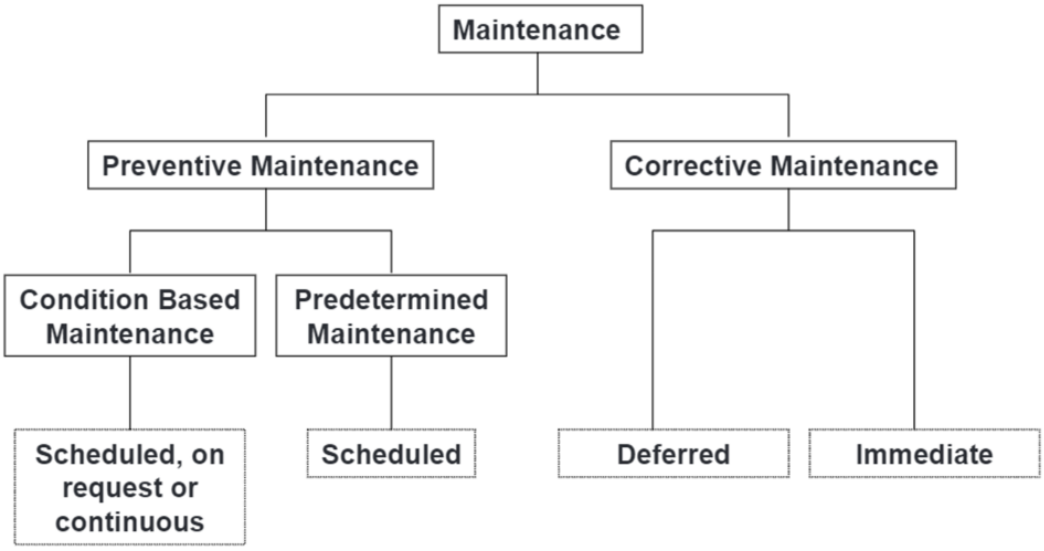


Figure 1. Maintenance – Overall view (EN 13306:2010)

Corrective Maintenance

Corrective maintenance is carried out reactively after a fault has been recognized. It aims to restore the equipment in question into a state in which it can again perform a required function. (CEN, 2010). Depending on the maintenance plan (or lack of one) corrective maintenance can be either planned or unplanned. Planned corrective maintenance is typically result of run to failure (RTF) maintenance plan where no maintenance is performed on the asset until the failure event. Unplanned corrective maintenance is typically result of a breakdown not stopped by preventive maintenance. Based on the fault and the business conditions corrective maintenance can either be done immediately after the fault occurs or it can be delayed and performed later, often while preventive maintenance is next performed. Corrective and unplanned maintenance is typically much more costly than planned and preventive maintenance. Despite this, it needs to be part of maintenance strategy, since equipment failure cannot often be reliably predicted, or the failing components are easy to replace and inexpensive. (Fiix software, 2017)

Preventive Maintenance

Preventive maintenance is performed proactively before the equipment fails. It intends to reduce the probability of failure or degradation of the functioning of an item. Preventive maintenance is scheduled: Predetermined maintenance is done based on established

maintenance programs for intervals of time or units of use and condition-based maintenance is done based on monitoring the equipment and scheduling maintenance based on the actual condition of the equipment. (CEN, 2010)

Predetermined Maintenance

Predetermined maintenance is based on a maintenance program based on established intervals of time or units of use but without considering the actual condition of the equipment. (CEN, 2010) The programs are typically provided by equipment manufacturers and are based on their knowledge of the failure mechanisms and mean-time-to-failure (MTTF) statistics for the equipment and its parts. The failure probability is typically higher when the equipment or part is new or worn out. This MTTF or bathtub curve is shown in Figure 2. Predetermined maintenance does not guarantee that the equipment does not fail and often leads to unnecessary repairs since the programs are based on failure statistics and not on the actual condition of the equipment. Managing predetermined maintenance can be complex since each equipment can have multiple maintenance programs and companies can have a large number of equipment. (Mobley, 2002)

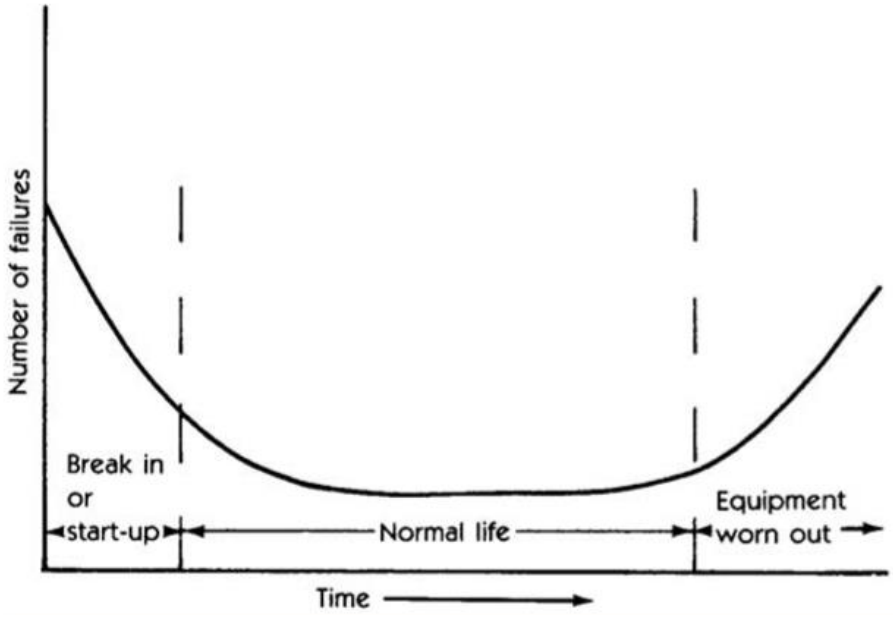


Figure 2. Typical bathtub curve (Mobley, 2002)

Condition Based Maintenance

Condition based maintenance tries to predict failure. It is based on regular monitoring of the condition, operating efficiency and other indications of the system. The monitoring can be done either on site or remotely via a network connection to the equipment. Monitoring can be done continuously, or it can be scheduled to happen at predetermined intervals. Monitoring can be either taking measurements from the equipment, inspecting it for wear and tear or running tests on the equipment. Condition-based maintenance is most complex maintenance type to implement but can be the most economical one since only the parts needing repair or replacement are maintained. (Mobley, 2002)

2.3 Maintenance Management

In today's business environment companies must constantly look for ways to gain competitive advantage. Many industries are very capital intensive and when maintenance costs represent so high percentage of total operating costs it is no wonder that companies are looking for ways to reduce maintenance costs. This must naturally be done without sacrificing equipment reliability or decreasing the quality of products and services. If equipment effectiveness, reliability or workforce productivity can be increased or usage of materials can be reduced this can have a significant impact on the company bottom line. For many companies, implementation of maintenance productivity programs often results in savings of 5% to 15% of total maintenance costs. (Campbell, 2016)

The activities of the management that determine the maintenance objectives, strategies and responsibilities and the implementation of them is called maintenance management. (CEN, 2010)

2.4 Computerized Maintenance Management Systems

Computerized Maintenance Management Systems are packaged software tools designed specifically to support companies in maintenance management. Most businesses that maintain equipment today have some sort of CMMS in use and there are hundreds of commercially available packages to choose from. The growth and evolution in IT has led to a dramatic increase in the capability and availability of software tools to support maintenance. This also means that existing systems can become outdated even in a very short time. Technology is however not a replacement for strategy and software should only be considered an enabler. Selection of wrong software, poor implementation

and poorly articulated goals often lead to project failures. Understanding CMMS functionality, comparing different software options and aligning them with business goals together with management support are key to successful implementations. Typically, smaller companies using specialized solutions are most satisfied with their CMMSs. The two most important features for successful implementation are ease of adapting to maintenance processes and user-friendliness. (Campbell, 2016)

Work Order Management

Work orders are at the heart of CMMS: They are used to initiate, track, and record all maintenance related activities. Work orders start as requests, which are then approved, the work is planned and scheduled, performed and finally recorded. Work orders contain detailed data about the maintenance in question and they produce valuable information on maintenance performance, costs and equipment history. Among the information tracked with work orders are:

- maintenance tasks and their start and completion dates
- detailed spare part usage
- detailed work instructions for each step
- labor and materials costs
- information about who performed the work
- life cycle information: where the work order originated from, when it was scheduled, approved, performed etc.

After the work order has been completed the information can be used to track maintenance costs for the equipment. The two main types of expenses that are tracked are time and material charges. Work order backlog is useful for determining staffing requirements and shutdown periods. (Wireman, 2013)

Equipment management

Equipment management contains information about each equipment in the system including subcomponents and even individual parts. The information can include basic information about the equipment such as identifying information, categorization information, and instructions, blueprints and pictures. It is also used to record the maintenance history of the equipment: the preventive and corrective maintenance done to it, what parts were replaced and when and what the downtime of the equipment has been. This enables analysis of equipment performance and maintenance costs.

Preventive maintenance module

The preventive maintenance module is used to create preventive maintenance plans for the equipment. The plans include service tasks to be performed for certain equipment at predetermined intervals of time or other usage metrics. These plans are then used by the system to automatically create work orders for equipment. The system can schedule work based on calendar time such as daily, weekly, monthly or yearly or based on a certain meter reading i.e. every 1000h machine hours. To be able to schedule maintenance based on meter readings the system needs to support logging of meter readings. They can be entered into the system manually or they can be uploaded automatically. (Wireman, 2013)

Planning and scheduling

One of the most important functions of a CMMS is planning and scheduling maintenance work. Planning of maintenance is strategic and refers to the design of maintenance work over time and how it will be done. Scheduling, on the other hand, is tactical and refers to what work will be done on what day and with what resources. For CMMS, planning refers to determining the maintenance policy for assets, including preventive maintenance programs, tasks, parts, and resources required for maintenance. Scheduling in CMMS refers to developing daily, weekly and monthly schedules, determining priorities for work, assigning maintenance work to technicians and maximizing asset availability. (Plant Services, 2010)

Inventory control and purchasing

Maintenance, Repair, and Operations (MRO) supplies are consumed in the production process but are not part of the end product. For many organizations, MRO inventory can account up to 40% of the annual procurement budget. For reliable operation, it is important to have the right parts available at the right time without tying up too much capital to inventory. It is also important to manage suppliers and purchases: some parts may have long lead times and the internal cost of processing purchase orders can be significant. Having an accurate and up to date data about inventory is key for solid MRO management (Modern Materials Handling, 2017).

3 Dynamics 365

This section introduces Microsoft Dynamics 365. The section starts with an overview of the suite and then describes how it can be extended both by customers using it and by 3rd party solution developers.

3.1 Overview and Standard Applications

Dynamics 365 is Microsoft's business application suite. The newest generation was launched in 2016 and combines Enterprise Resource Management (ERP) and Anything Relationship Management (xRM) functionality into a single platform. Dynamics 365 ERP is based on the Dynamics AX product and Dynamics 365 xRM is based on the Dynamics CRM product. Out of the box, the suite contains six business applications, which can be deployed separately or in any combination based on the customer need.

Technically, ERP and xRM are separate products and have their own databases, code and user interfaces. Microsoft is working on bringing the products closer to each other and is using the same product name for both, which can be confusing to readers not familiar with the products. For the scope of this thesis the Dynamics 365 will be used to refer to the xRM part of the product, which covers most of the applications and Dynamics 365 for Finance and Operations to refer to the ERP part, which covers Finance and Operations.

Sales

Dynamics 365 for Sales helps companies manage their sales pipeline. The application enables salespeople keep track of their accounts and contacts and to use a guided process to manage sales from lead to order. The application can be used to create sales related materials like quotes, orders, and invoices. The application also supports many marketing functions such as marketing campaigns and customer feedback. For managers, there is comprehensive analytics about the sales team performance.

Customer Service

Dynamics 365 for Customer Service allows companies to run customer service operations. Companies can use it to manage cases using queues, routing, service level agreements, and entitlements. There is also a knowledge base that can be used to find out and share information about products and services. The application includes a portal that can be used to provide end customers self-service and case tracking functionality.

Marketing

Dynamics 365 for Marketing helps companies create personalized digital marketing campaigns and to unify them with sales so that they can create better customer experiences.

Field Service Automation

Dynamics 365 for Field Service provides a complete Field Service Management solution that includes managing service locations and customer assets, work order management, resource management, product inventory, scheduling, and dispatch. The solution also includes functionality that enables automatic creation of recurring work orders that can be used for scheduling preventive maintenance tasks. There is also inventory management that can be used to track real-time inventory levels by warehouse.

Project Service Automation

Dynamics 365 for Project Service Automation enables companies do project planning, contract estimation, and resource management. Time and expenses can be tracked and billed.

Finance and Operations

Dynamics 365 for Finance and Operations is the ERP application in the product family. It includes support for the typical ERP functions like financial management, production, human resources, and planning.

Dynamics architecture

Dynamics 365 integrates into Office 365 and enables collaboration and working with Office documents without switching between applications. Microsoft's Business Intelligence products Power BI and Cortana Intelligence are natively embedded to the platform. Dynamics 365 is a cloud-based solution and runs on Microsoft's cloud platform Azure, but with some limitations, it can also be run on premises. The architecture of Dynamics 365 is shown in Figure 3.

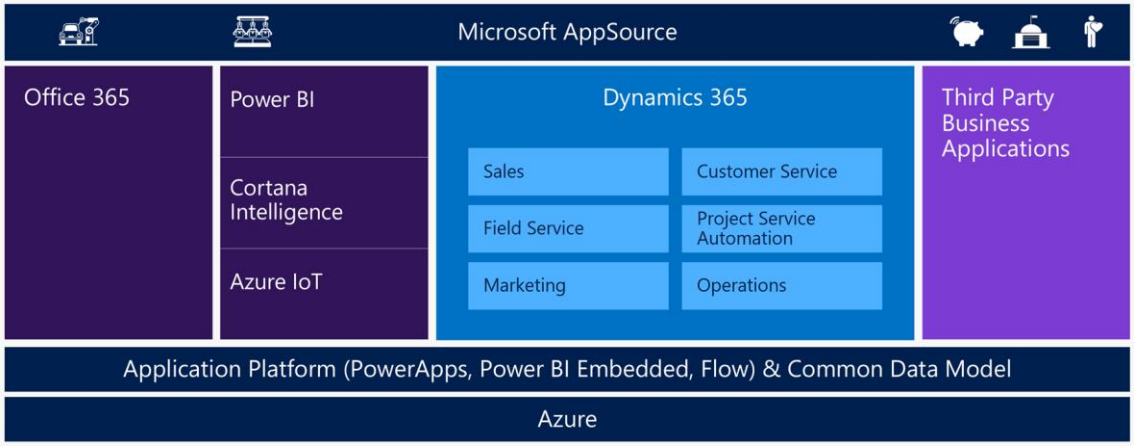


Figure 3. Dynamics 365 architecture

A key feature of Dynamics 365 is the ability for customers to adapt the out of the box applications to suit their needs by modifying the data model and processes. Additionally, the suite includes a business application platform that enables tech-savvy customers to implement low-code line of business (LOB) apps using the tools listed in Table 1.

Table 1. Dynamics business application platform components

Tool	**Purpose**
PowerApps	Building cross-platform mobile apps without coding.

Flow	Automating workflows between applications.
Common Data Model	Secure unified view to business data.

Finally, the platform contains support for creating 3rd party business applications on top of the platform and distributing them to Dynamics 365 users via a digital marketplace called AppSource. The rest of this chapter is devoted to describing the support for creating new applications. (Microsoft, 2017)

3.2 Dynamics 365 as Application Platform

Dynamics 365 offers developers a platform for creating line-of-business (LOB) applications. An LOB application is a business-critical software application that is vital to running an enterprise. The platform contains support for developing applications for both the ERP and xRM modules, but since only the xRM capabilities are in the scope of this thesis, only that application development capability will be covered in this chapter.

Building applications on top of a platform offers developers a lot of advantages compared to starting from scratch. For Dynamics 365 these include (Microsoft, 2017):

- rapid application development
- user interface ready for international use
- extensible data and security model
- extensible business logic
- Office 365 integration
- cross-platform mobile applications

- business intelligence

- support for standard web technologies

Developers also benefit from the periodic updates to the platform: New features are introduced twice a year and bug and security fixes as required. This way developers can leverage Microsoft's development effort in their solutions and can focus on developing their core functionality.

3.3 Extending Dynamics 365

Dynamics 365 can be extended in various ways. There are several customization points in the system that can be used either separately or in any combination. Each customization point serves a specific purpose and it is often necessary to use several of them to build a complete application. The customization points are explained below and are shown in Figure 4.

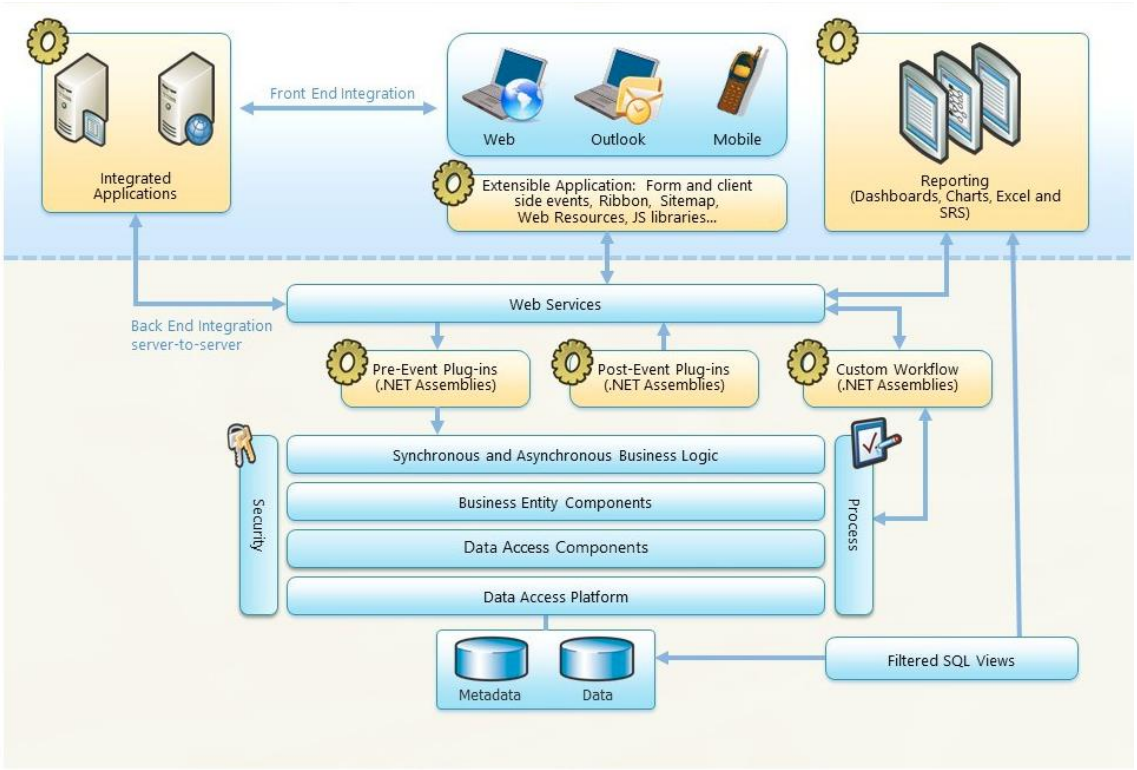


Figure 4. Dynamics 365 Customization points (Microsoft, 2017)

Customizing applications

The simplest way to extend Dynamics is by customizing the out of the box applications. Users can customize the views, forms, and entities without coding. They can add new fields, modify or delete existing ones and do basic customization of views and reports. It is also possible to create actions and processes that perform various actions when something happens, for example sending an email when a record is created. For some use cases, simple customization might be the only thing needed to take the application into production use. When extending the system, some customization to the entities etc. is almost always done.

Client-side extensions

Dynamics 365 includes clients for web, phones, and tablets. Each of these can be extended using JavaScript and HTML and the extensions can be applied to all clients rather than writing separate code for each client. Simple tasks like validation, automation and process enhancement and enforcement can be done by extending the Dynamics forms with JavaScript code using the provided SDK. The object model of the SDK is shown in Figure 5.

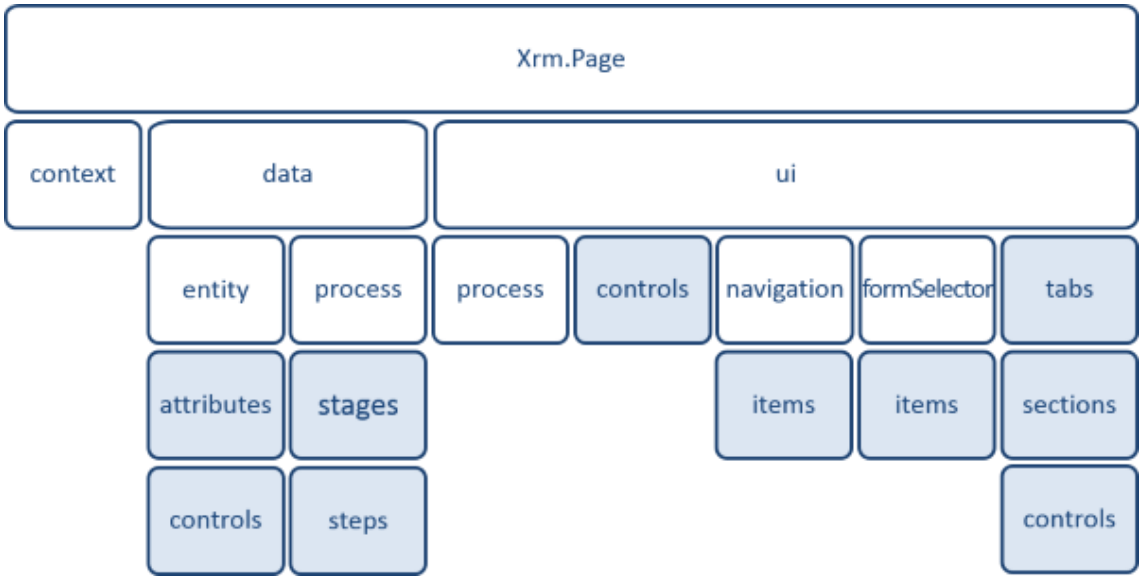


Figure 5. Xrm.Page object model (Microsoft, 2017)

Client-side extensions can also be complete HTML files that use 3rd party JavaScript libraries such as jQuery. The HTML files are rendered inside Dynamics web pages as

iframes. Client-side extensions are best used to implement UI customizations. They can access Dynamics 365 data and services using Web Services or Dynamics Web API, which is an OData 4.0 RESTful API.

Server-side extensions

Microsoft provides a Software Development Kit (SDK) for extending Dynamics on the server. Server-side development is done using .NET and is used to create code to handle custom business logic or custom workflow activities.

Plugins are the most advanced way to extend business processes. They allow developers to write code that is executed when an event is fired by the platform. The platform fires events in various situations, but to most developers, the interesting events are when records are created, updated or deleted. Developers can hook code to events by setting the event properties shown in Table 2. Multiple plugins can be hooked to the same event and many events can be hooked into the same plugin. For all plugins, there is a 2-minute time limit for execution. If it exceeded the system terminates the plugin execution and an exception is thrown.

Table 2. Event properties

Property	**Use**
Entity	The database object type
Message	The event: Create, Update, Delete, etc.
Pipeline stage	Pre-validation, Pre-operation, Post-operation
Synchronous / asynchronous	Whether the event processing needs to be done immediately or queued by the system for later execution.

Since the plugins can run before the system modifies the database, they can be used for doing advanced validation. Also, since they are used in a transaction context they are

well suited for complex database modification, where a single failure could lead into a corrupted database. Figure 6 shows how the event execution pipeline works.

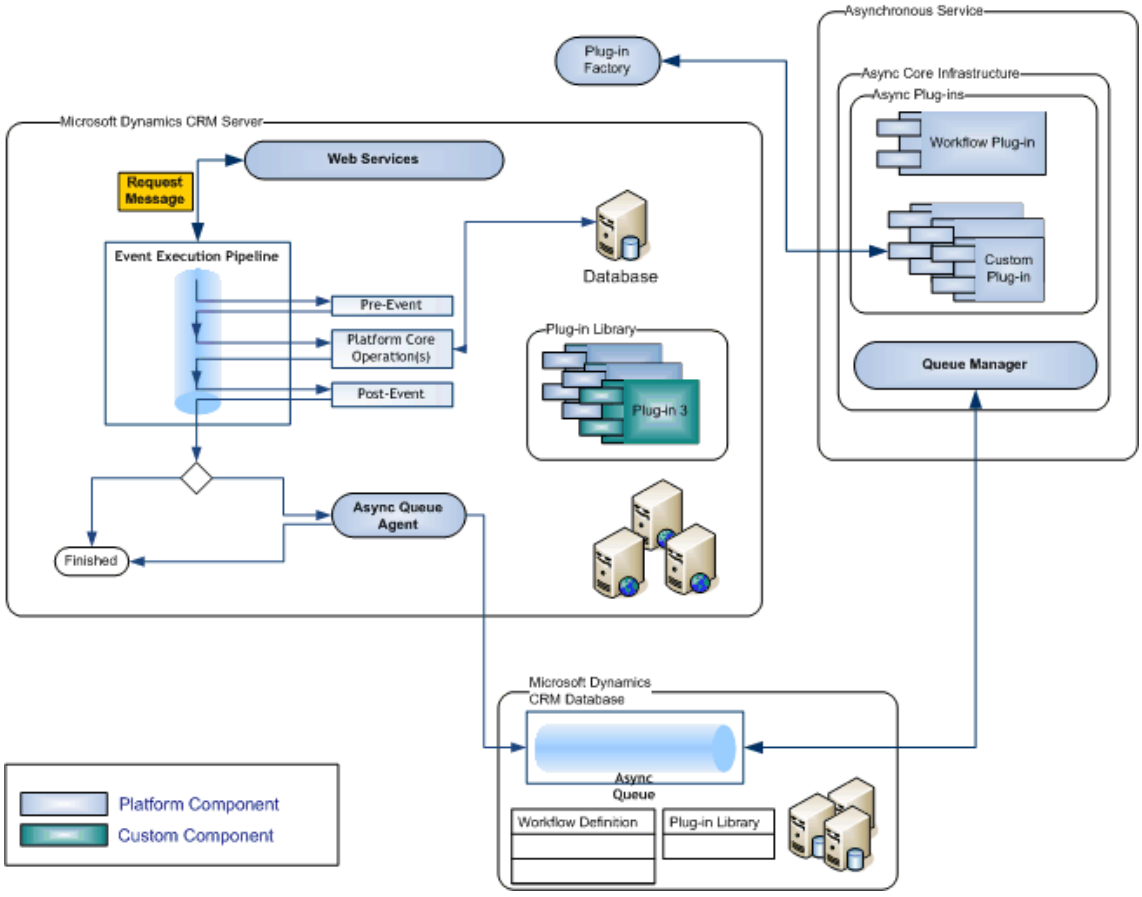


Figure 6. Event execution pipeline (Microsoft, 2017)

Custom Workflow Activities can be used to create business processes that can be called from within Dynamics forms, processes and the Web API. They are registered to the system using the same parameters as plugins and additionally they have input and output parameters. Custom Workflow Activities are executed based on explicit calls and not on events.

Metadata and data models

Dynamics 365 uses a metadata-driven architecture that gives developers the flexibility to modify the data structure without any change in the code on the server or client applications. The metadata architecture also hides the underlying data storage layer from the developers, which means that it cannot be accessed directly, only via system provided

APIs. Developers can add new metadata objects to Dynamics or they can modify existing objects by adding, modifying and deleting attributes. This includes changes to many of the system defined objects. To preserve system integrity and proper functioning of applications developers can protect metadata objects from further customization. This protection is always in place for critical system-defined objects. Metadata model also includes localization support: In addition to the system names, the metadata objects include localizable display names that are used by the client applications based on the language in use. The metadata objects and their use are described in Table 3.

Table 3. Metadata objects (Microsoft, 2017)

Metadata object	**Description**
Entity	A container for data representing an object. Entities are the things, persons, places, and objects in the system. Entities contain a set of attributes.
Attribute	Defines the information about the entity that needs to be stored. The attributes have a name and a datatype: text, number, date, image, currency, option set etc.
Relationship	A relationship defines an association between entities. Dynamics supports 1: N, N:1, N: N and self-referential entity relationships.
Option set	Option sets define a collection of named values for an attribute.
Option	Option is a single value available in an option set.

Reports

Dynamics 365 reporting is based on Microsoft SQL Server Reporting Services. Reports can be created by querying the metadata model using FetchXML-queries and authoring the layout using Report Definition Language (RDL). There is also a user-friendly editor called Visual Studio Report Designer. Dynamics reports can be integrated into Dynamics forms and dashboards and they can also be accessed via an HTTP interface.

Integrated applications

Developers also have an option of creating separate applications that integrate to some Dynamics 365 functionality. These applications can integrate to Dynamics back-end via Web Services and they can also be embedded into front-end applications using HTML.

3.4 Solutions

Solutions are the way to package the various extensions into redistributable packages. For 3rd party application developers, they are the way to provide their application to the customers. With solutions, the various solution components can be managed and deployed together. The solution components that can be included in a solution are listed in Figure 7. It should be noted that data is not a part of the solution and to include data in a deployment some other mechanism than solution must be used.

Figure 7. Solution components (Microsoft, 2017)

Solutions are created using the Solution Editor. Using the solution editor developers can select the components to be included in the solution and manage their dependencies.

Developers can use the solution editor to set the publisher and version number of solution and it can also be used to export the solution as a redistributable file. The user interface of Solution editor is shown in Figure 8.

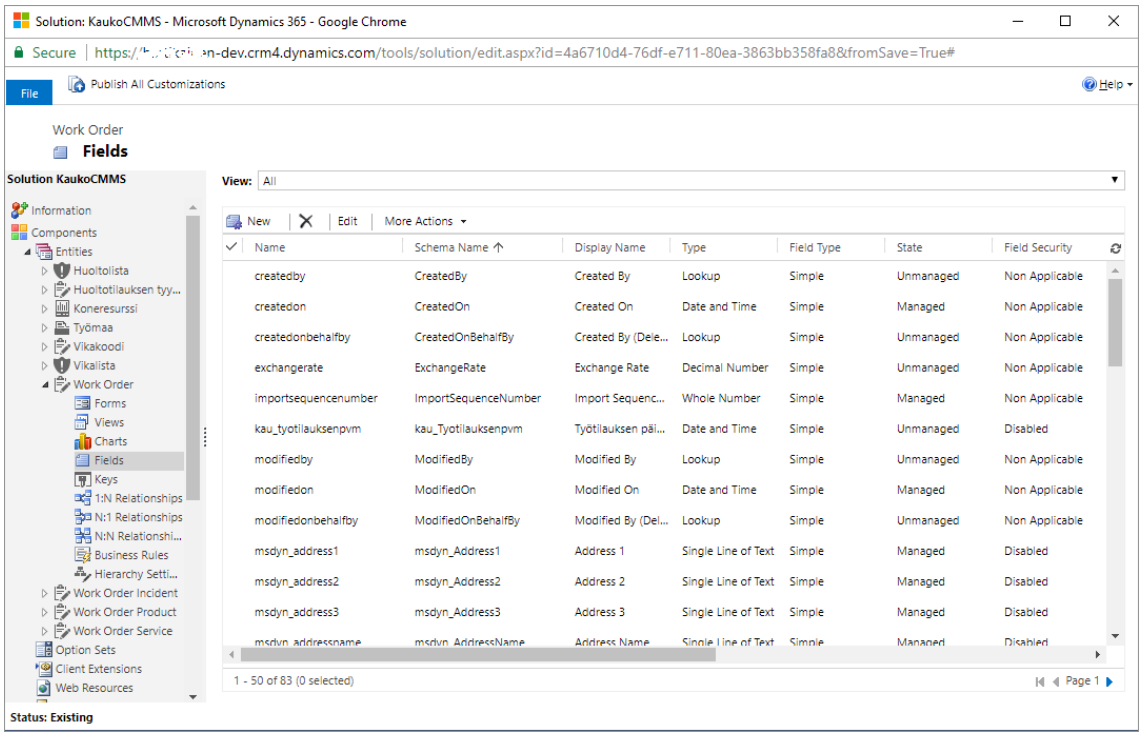


Figure 8. Solution editor

Dynamics solutions can be either managed or unmanaged. Managed solutions are completed solutions that are meant to be distributed and installed. Unmanaged solutions are solutions under development or ones that are not meant for distribution. Managed solutions can lock down system functionality and they provide a way to resolve potential conflicts between solutions customizing the same entity or another area of the solution. Managed solutions can also be uninstalled.

4 CMMS Requirements

This chapter begins the technical part of the thesis. Based on the literary review a list of essential requirements for a CMMS was created. In the real world, the exact requirements for a CMMS vary a lot between organizations but there are many basic requirements that any feasible CMMS should implement. The minimum identified requirements are listed in Table 4.

Table 4. Identified requirements for CMMS

Category	**Requirements**
Work Order Management	R1.1 Tracking planned and actual labor R1.2 Tracking planned and actual materials R1.3 Sortable work order backlog R1.4 Resourcing work orders to technicians R1.5 Managing work order status
Preventive Maintenance	R2.1 Creating Preventive Maintenance Programs R2.2 Scheduling maintenance based on calendar time R2.3 Scheduling maintenance based on meter reading R2.4 Manual scheduling R2.5 Forecasting labor, material and tool requirements R2.6 Combining due Preventive Maintenance R2.7 Written Predictive Maintenance procedures
Corrective Maintenance	R3.1 Punch list management for equipment R3.2 Scheduling corrective maintenance

Asset Management	R4.1 Tracking equipment information R4.2 Saving work orders to equipment history R4.3 Tracking cost and repair information on component level R4.4 Corrective maintenance management R4.5 Bill of materials for each piece of equipment R4.6 User defined screens for storing additional information about equipment
Inventory Management	R5.1 Tracking inventory on-hand R5.2 Tracking unit price information R5.3 Support for multiple warehouses R5.4 Support for warehouse to warehouse transfer R5.5 Generation of purchase orders
Reporting	R6.1 Predefined reports R6.2 Custom report editor R6.3 Maintenance budget reporting module R6.4 Tracking equipment downtime

The focus in the prototype was on functional requirements that define how the system should behave and what the inputs and outputs to the system are. Additionally, systems have non-functional requirements like accessibility, performance, interoperability etc. but no non-functional requirements were included in the list since they typically are very case specific.

4.1 Fit-gap Analysis

In this thesis, a fit-gap analysis was performed by comparing the requirements found in Table 4 to the standard functionality of Dynamics 365. Each requirement was analyzed for the fitting Dynamics functionality and given a rating from 1 (poor) to 5 (great) based on how well Dynamics 365 functionality fits the requirement. The identified gaps are also

listed along with the classification on if the requirement can be met by customizing or extending Dynamics 365. The results of the analysis are shown in Appendix 1. The results show for example that requirement R2.3 has a poor fit (1), since there is no functionality in Dynamics 365 for it and to implement support for it would require extending the system. Requirement R1.1 on the other hand has great fit (5), since there is identified functionality in the system and the implementation can be done by customizing the system.

4.2 Functionality Selected for Prototype

Based on the fit-gap analysis requirements were then selected for the prototype implementation. The selected requirements are shown in Table 5. The table shows which requirements were selected, the rationale of selecting them and what technologies are required for implementing the requirement.

Table 5. Requirements selected for the prototype implementation

Requirement	**Rationale**	**Technology**
R1.1 & R1.2	Fit validation	Customization
R2.2	Fit validation	Customization
R2.3	Functional gap Usability	Customization Front-end development Back-end development
R2.4	Usability	Front-end development

R2.6	Functional gap	Back-end development
R2.7	Functional gap	SQL Server Reporting Services
R3.1 & R3.2	Functional gap	Customization Front-end development Back-end development
R4.4	Functional gap	Customization Front-end development
R5.1	Fit validation	Customization

The focus of the prototype implementation is not to cover all the requirements but rather to implement functionality to fit the key functional gaps, verify the fit of key requirements and to test the various extension and customization technologies. This approach results in good coverage of key questions while keeping the development effort small.

5 Solution Implementation

This chapter describes the actual implementation of the solution. It starts with an overview of creating a Dynamics 365 solution and then describes the various objects added to the solution.

5.1 Overview

The solution implementation started by creating a new Dynamics 365 solution to contain all the customizations and extensions needed. The solution was named **Kauko CMMS** and the publisher set to Kauko Oy. Solution publisher information is used when distributing solutions and it helps to manage schema namespace by having a prefix for metadata entities. For Kauko Oy the prefix is **kau** and it is used in the naming of all custom entities, attributes, and resources. Solutions support versioning and upgrades so a version number of 0.1 was set. To test the localization ability of Dynamics 365 it was decided to localize the solution to the Finnish language.

Dynamics 365 supports dependency tracking of solution components. This enables developers to create solutions that depend on other solutions and only include the necessary customizations to their solutions. The system then enforces that the necessary components are present in the system when importing solutions. Dependencies are also enforced when uninstalling solutions so that the integrity of the other solutions is not compromised by removing components that they require. Dynamics 365 out of the box functionality can also be used as the foundation of solutions and entities and other objects already present in the system can be depended on and customized further. By using standard object model developers can leverage the functionality already present and cut development time while maintaining interoperability with other solution providers. For the implementation, it was chosen to depend on Dynamics 365 for Field Service schema and the minimum version of Dynamics 8.2, so these were installed in the development instance.

5.2 Schema Customizations

Schema or the data model is at the heart of the solution since it describes what data will be processed by the system and how the data is stored. For the implementation, the existing schema in the system was evaluated against the requirements and the following questions were asked:

- What existing entities can be used in which role?
- What new attributes and relationships need to be created?
- What new entities need to be created?

The goal in the implementation was to minimize the customizations done so that the requirements are met, but the end users still have the flexibly to add their own customizations based on their specific needs.

Equipment management

Customer Asset (**msdyn_customerasset**) entity was chosen to represent a single piece of equipment that can be maintained. It is the foundation of asset management and enables storing information about equipment and managing the life cycle. The attributes listed in Table 6 were added to the entity.

Table 6. Customizations to the msdyn_customerasset entity

Attribute	**Type**	**Use**
kau_asssettype	Option Set	Filtering asset categories, separating solution assets from generic Customer Asset entities

kau_serialnumber	Text	Serial number / identifier for the asset
kau_description	Text / multiple lines	Description data for the asset

Preventive maintenance

For defining preventive maintenance programs, it was chosen to use standard schema entities that are part of the Field Service application. These entities allow the definition of collections of service tasks and spare parts that are needed during maintenance. The entities and their selected role in the prototype are listed Table 7.

Table 7. Entities for managing preventive maintenance programs

Entity	**Purpose**
msdyn_incidenttype	Represents a single maintenance program.
msdyn_incidenttypeproduct	Represents spare parts required for the maintenance program.
msdyn_incidenttypeservicetask	Represents a work item to be done as part of maintenance

Product	Represent the spare part catalog.
msdyn_servicetask	Represents different work items that can be done as part of maintenance.

For actual maintenances of equipment, it was also decided to use the standard schema, which is focused around the Work Order entity. The relevant entities and their selected roles in prototype implementation are listed in Table 8.

Table 8. Entities for managing maintenance work orders

Entity	**Purpose**
msdyn_workorder	Represents a single maintenance.
msdyn_workorderincident	Represents a maintenance program to be done as part of the maintenance.
msdyn_workorderservicetask	Represents work done as part of maintenance. Allows tracking of hours.
msdyn_workorderserviceproduct	Represents spare parts allocated and used as part of the maintenance. Allows tracking of MRO costs.

To enable scheduling the schema additions in Table 9 were made.

Table 9. Schema additions for scheduling

Entity	**Object**	**Purpose**
msdyn_incidenttype	kau_product	Defines the product that this maintenance program is used for.
msdyn_incidenttype	kau_maintenanceInterval	Setting the interval between maintenances
msyn_incidenttype	kau_maintenanceInterval type	Option set defining the measure for scheduling maintenance (machine hours, oil change, number of uses)
msdyn_workorderincident	kau_maintenanceProgramNumber	Identifying which maintenance program this is (ie. 10th 1000h maintenance)
msdyn_workorderincident	kau_operatingHours	Actual / predicted machine hours at maintenance time.

msdyn_workorder	kau_workOrderDate	Tracking when a maintenance work order was performed

Two custom entities were also created for scheduling: **kau_operatinghours** entity stores the operating hours measurements and **kau_usageplans** stores information about the predicted daily use of the equipment. Using the information in these entities it is possible to calculate when preventive maintenance programs are due for each equipment and to create a tentative maintenance schedule automatically. The custom entity definitions are given in Appendix 2.

Corrective maintenance

Support for corrective maintenance was implemented by using two custom entities. **kau_faultList** entity allows storing fault information for assets and recording fix and downtime information about faults. **kau_faultCode** entity allows classification of faults for planning and reporting purposes. The entity definitions are shown in Appendix 2.

Entity model diagram

The resulting entity model diagram for equipment management, preventive and corrective maintenance is shown below in Figure 9. It shows the relevant system entities, custom entities and key relationships between entities. Many of the system defined entities and attributes have been left out for simplicity. The custom entities that were created in the implementation are shown in orange color and the system entities in blue color. For each entity the primary key attribute is marked with [PK] -text. Also shown are the added custom and most important system attributes.

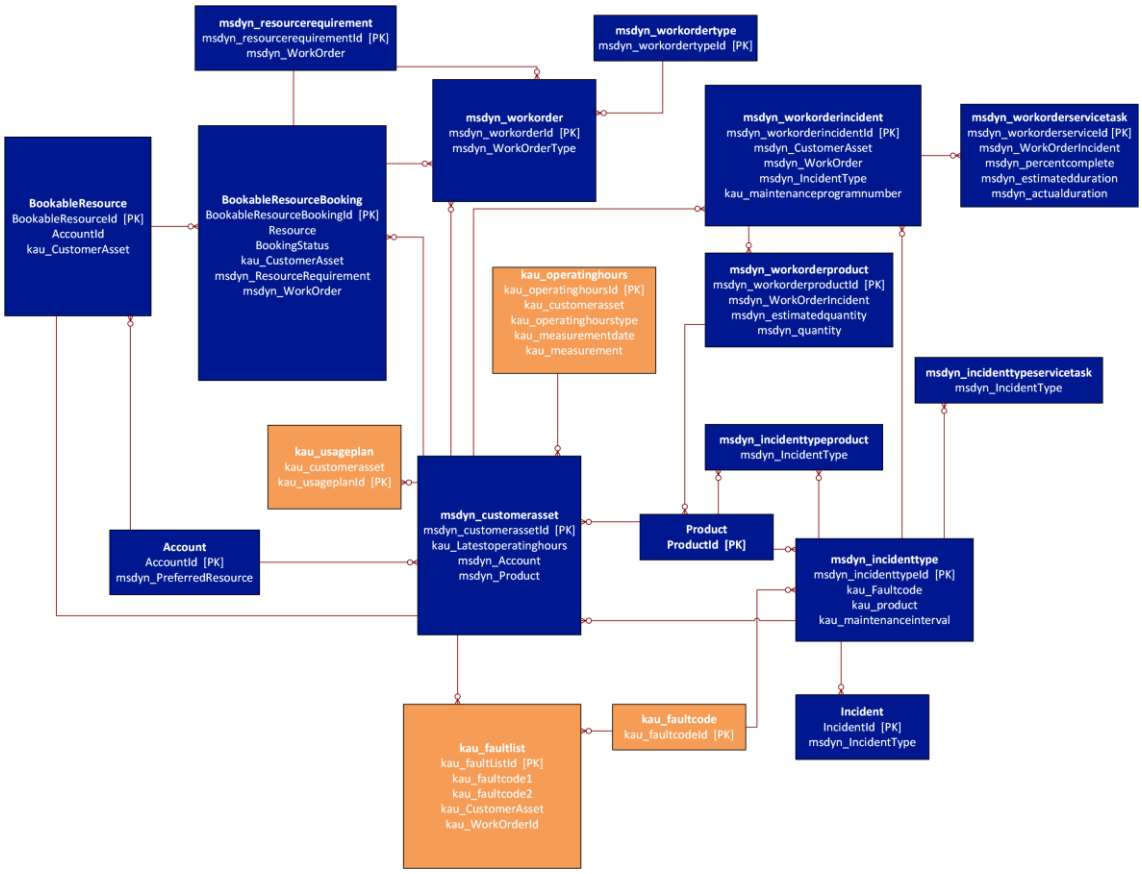


Figure 9. Solution metadata model with custom entities (equipment management, preventive and corrective maintenance)

Inventory management

Inventory management functionality of Field Service application was analyzed, and it was found to fit the requirements without modifications. The relevant entities and their purpose are defined in Table 10.

Table 10. Inventory management entities

Entity	**Purpose**
Product	Represents a product.
msdyn_warehouse	Represents a warehouse.
msdyn_productinventory	Presents the inventory.
msdyn_purchaseorder	Represents a purchase order.
msdyn_purchaseorderproduct	A purchase of single product in an order.
msdyn_inventorytransfer	Journal for logging transfers between warehouses.
msdyn_workorder	Allocation and use of MRO parts.

The simplified inventory management model with key entities and their relationships are shown in Figure 10. The entities in green are Organization-owned entities, meaning that they are viewable by all users in Dynamics 365.

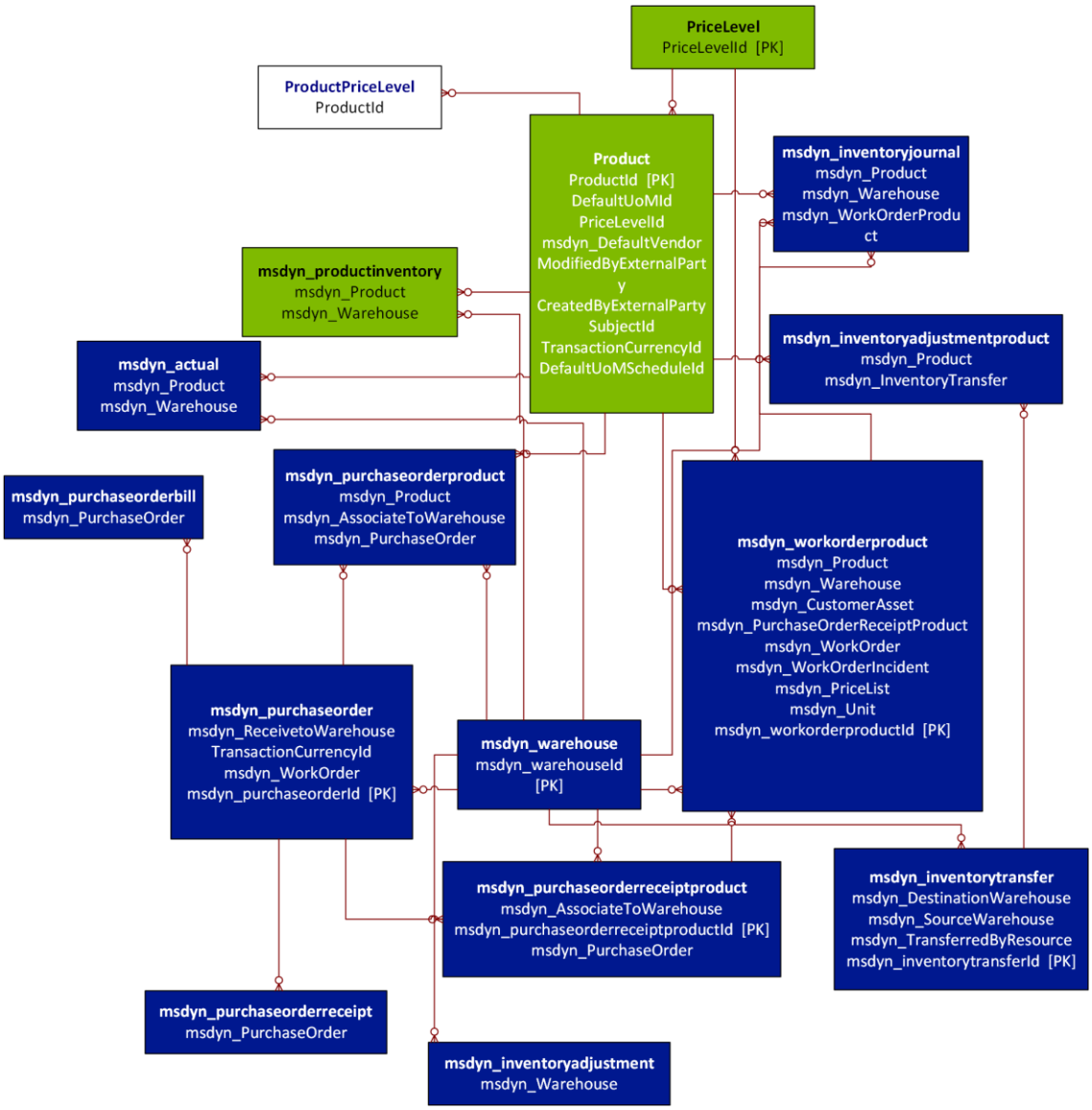


Figure 10. Solution metadata model (inventory management)

Product entity is one of the most central entities in Dynamics 365 and is used by many of the standard applications. Being able to use it directly in the prototype is a demonstration of benefits of developing applications on top of the platform.

5.3 Views and Forms

Customizing views and forms using the solution editor is the no-code solution for customization. It was used for functionality that needed only simple changes like adding new metadata fields. The biggest customizations made in the project are described below.

Asset form

New fields added to the metadata were also added to the customer asset form. Additionally, the work orders, fault lists and operating hours related to the customer asset were added as associated views to the form to allow viewing the most important asset related information on one page.

Fault code management

Fault code is a new custom entity as described in 5.2. For managing fault codes, the system automatically creates a default read-only view that lists all the fault codes and can be used to open the default edit form for each code. To make the management of fault codes easier this view was changed from the read-only grid to editable grid, which supports inline editing of existing records.

Sitemap

The new entities and views were added to the sitemap, which is the navigation UI of the application. To make the application more usable, unused parts of Dynamics 365 were removed from the sitemap. The sitemap of the solution is shown in Figure 11.

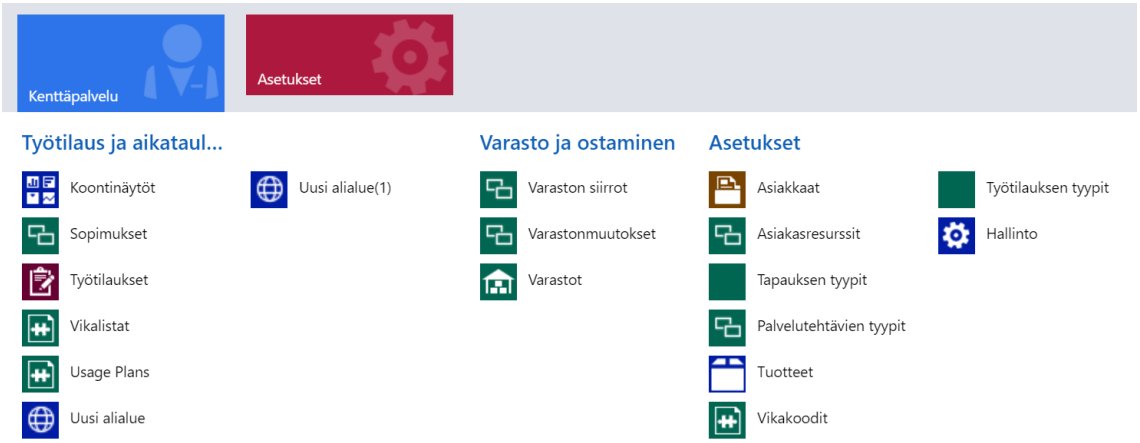


Figure 11. Application sitemap

Sitemaps can include multiple levels of menus: areas in the top, groups below them and finally sub areas that are menu items that navigate to various parts of the application. Sitemaps are security aware and the users see only the parts of the application that they have access to.

5.4 Web Resources

For usability purposes, it was decided to implement some of the functionality using HTML and JavaScript and to include them in the solution as Web resources. Web resources are added to Dynamics solutions as files. There are restrictions to file names and extensions, so it is best to keep the number of files small. To avoid having to rename some 3rd party files with Dynamics incompatible names, some files were included using Content Delivery Network (CDN) service and not included in the solution as files.

Development tools

In the implementation, 3rd party JavaScript libraries were included using npm package manager, which takes care of dependency management and versioning and provides a registry for discovering and installing packages. The packages used in the implementation are described Table 11.

Table 11. 3rd party JavaScript libraries used in the project

Package	**Usage**	**License**
@progess/kendo-ui	User interface components	Commercial
jQuery	HTML document traversal	Open source
vis.js	Timeline control	Open source
xrm-webapi-client	A promise based library for Dynamics CRM Web API	Open source
underscore	utility functions	Open source

To make the JavaScript easily maintainable a build system was setup around Webpack, which is a static module bundler that analyzes code for dependencies, includes relevant modules and packages the code into bundles. It supports plugins and loaders to perform various tasks as part of the build process. The build system was configured to use BabelJS to transpile JavaScript into cross-browser compatible version and UglifyJS to make the code smaller and faster.

The editor used in development as WebStorm IDE. It provides several key functionalities that enhance developer productivity: syntax highlighting, code completion, error detection and code refactoring.

Single Page Application

Since usability is a big factor in an enterprise application, user navigation (web page loads) to perform key use cases should be minimized. For this purpose, the web pages were implemented using a Single Page Application (SPA) design that allows changing the web page content without reloading the page from the server. All the required user interface components are retrieved from the server when the page loads and JavaScript code is then used to render parts of the user interface as required. The SPA design was done using Kendo UI router, which allows tracking the application state and using views and layouts to render the UI. The router uses fragment part of URL to store the state and to determine the views to render. Since the state is part of URL, it can be used to bookmark the specific location in the application and to enable proper functionality of browser back-button.

Dynamics Web API

Communication with Dynamics 365 to read, write, update and delete data and to execute custom workflow activities was done using the Dynamics Web API. It implements OData (Open Data Protocol) version 4.0, which is an OASIS standard for building and consuming RESTful APIs. An open source library xrm-webapi-client was used to make working with Web API from JavaScript easier. The library supports all the needed API functionality including batch queries, asynchronous calls and invoking custom workflow activities. The library returns results from API calls as JSON objects and automatically detects Dynamics CRM server URL and authentication tokens from the pages embedded in Dynamics 365. Sample code to retrieve customers and customer assets in a single API call is shown in Figure 12.

```
export function getCustomerAssets () {
    var requestCustomerAssets = {
        entityName: "msdyn_customerasset",
        queryParams: "?$select=msdyn_name,_msdyn_account_value,kau_assetType&$orderby=msdyn_name asc",
        asBatch: true
    };
    var requestCustomers = {
        entityName: "account",
        queryParams: "?$select=name",
        asBatch: true
    };


    var batchItems = [];
    batchItems.push(WebApiClient.Retrieve(requestCustomerAssets));
    batchItems.push(WebApiClient.Retrieve(requestCustomers));

    var batch = new WebApiClient.Batch({
        requests:  batchItems

    });

    return WebApiClient.SendBatch(batch);
}
```

Figure 12. Example asynchronous Web API query to retrieve customer assets and customers

Scheduling view

Scheduling view was done using vis.js timeline control. It shows the assets in a list that can be filtered by asset type and site. Users can navigate in time by selecting the time

scale and moving forward and back in time. The scheduling view allows users to see upcoming maintenance tasks, change their status, reschedule them and open the related work orders and asset pages. The user interface of scheduling view is shown in Figure 13.

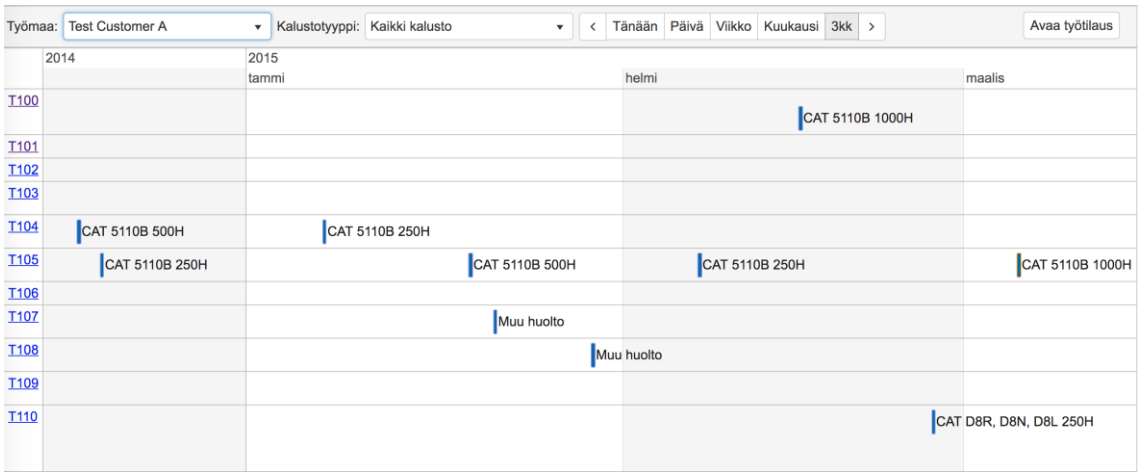


Figure 13. Scheduling view user interface

Work order management

Work order page was implemented as a Kendo UI TabStrip, which shows the various maintenance programs linked to the work order as tabs as well as the faults and spare parts linked to the work order. The various lists of tasks in the tabs were implemented using Kendo UI Grid widget. Using work order page, service tasks can be marked completed along with notes, products can be used from inventory and faults can be associated with the work order and marked fixed. Users can also print out the work order instructions. The user interface of work order page is shown in Figure 14.

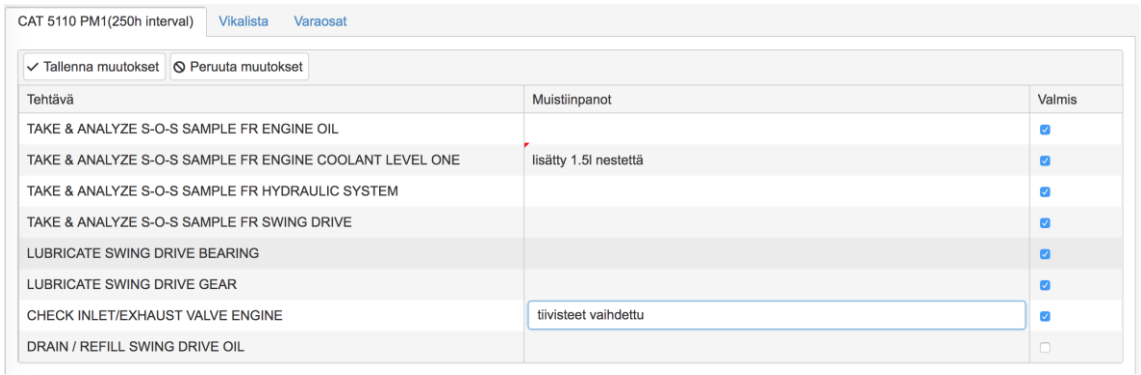


Figure 14. Work order page showing work order service tasks

Fault list

Fault list was implemented using Kendo UI Grid. It shows the faults associated to the equipment along with fault codes and statuses. Users can also create new corrective maintenance work orders from selected faults. The user interface of fault list page is shown in Figure 15.

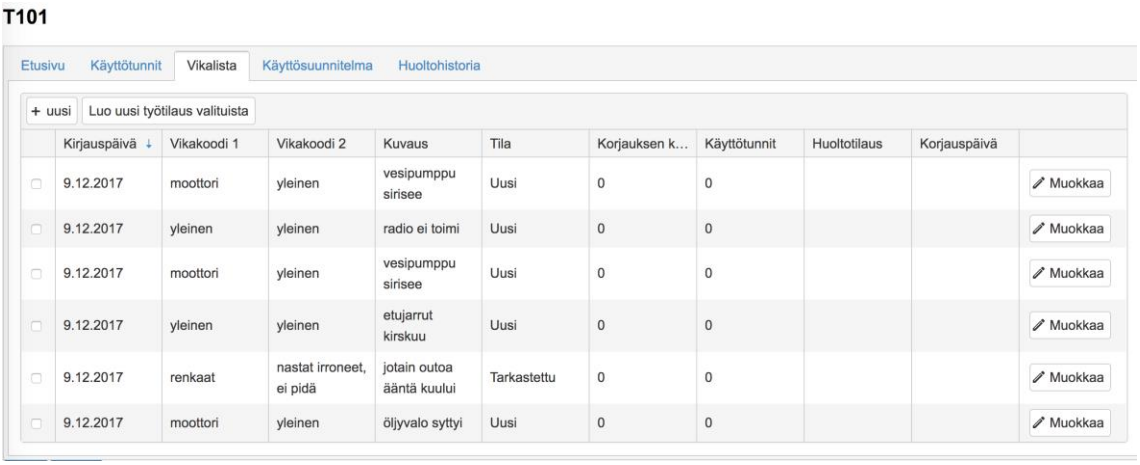


Figure 15. Fault list

Logging machine hours

Machine hours logging was implemented as a separate HTML page. It allows users to filter the assets using asset type and site and to record new measurements for each measurement type. Previous measurement date and the recorded value are shown for reference. The page was implementing using Kendo UI Grid widget. The user interface of the page is shown in Figure 16.

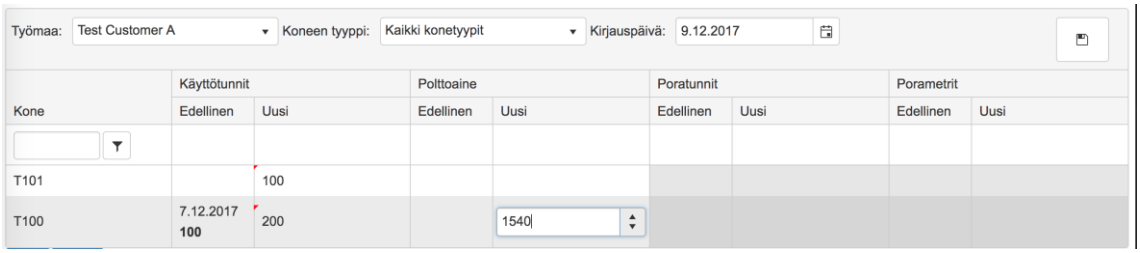


Figure 16. Machine hour logging user interface

After the development was completed, the finalized HTML and JavaScript files were packaged using the build system and the resulting bundles were added to the solution

using the solution designer. As a best practice, they were added to a separate folder with the solution publisher prefix. This resulted in the following URL naming for them: https://tenant-name.crm4.dynamics.com//WebResources/kau_/filaname.ext.

5.5 Back-end Code

Back-end business logic was developed by creating a single .NET assembly containing all the necessary code. Combining the code into a single assembly makes managing the solution simpler and when running in cloud-based environment external references to other assemblies are not supported. Development was done using C# programming language in Visual Studio 2017 Enterprise.

Dynamics 365 interacts with custom assemblies using the IPlugin interface. The interface has a very simple signature consisting of only one method shown in Figure 17.

```
namespace Microsoft.Xrm.Sdk
{
    //
    // Summary:
    //     Base interface for a plug-in.
    public interface IPlugin
    {
        void Execute(IServiceProvider serviceProvider);
    }
}
```

Figure 17. IPlugin interface description

Using the **IServiceProvider** reference in the Execute method, developers can access the various services available to the plugin and interact with the system. The services are listed in Table 12.

Table 12. Services available to plugins

Interface	**Use**
IPluginExecutionContext	Access to input and output parameters and to message name.
IOrganizationService	Access to Dynamics 365 data.
ITracingService	Logging diagnostic information.

The prototype implementation consisted of two plugins and three custom workflow activities. The activities were registered to Dynamics 365 using the execution pipeline parameters shown in Table 13.

Table 13. Execution pipeline parameters for prototype messages

Entity	**Messages**	**Stage**	**Mode**
kau_operating-hours	Create	Pre-validation	Synchronous
kau_usage-plan	Create Update	Pre-validation	Synchronous
none	kau_ScheduleMaintenace	Post-operation	Asynchronous

none	kau_MoveBooking	Post-operation	Synchronous
None	kau_CreateFaultWorkOrder	Post-operation	Synchronous

Validating of user input

Two user input validations were done on the server side because of their complexity: validation of machine hours logging and usage plans. For machine hours, the validation consisted of checking if a newer measurement already exists and for usage plans the validation checked that there are no overlaps in planned usage segments that would make determining the next maintenance dates unambiguous. In case of a validation failure an exception is thrown and because of the configuration the transaction is rolled back, and the invalid values are not entered into the database.

Scheduling Preventive Maintenance

Scheduling preventive maintenance and moving existing maintenance to a new date were implemented as custom workflow activities so that they can be invoked explicitly from other processes and the front-end code. The implementation consisted of calculating when maintenance is due and creating new records in the database. To stay within the 2-minute execution time limit for the code the scheduling is done per asset. Figure 18 shows how the back-end plugin can be called from front end JavaScript.

Associating faults to work orders

Creating a new corrective maintenance work order and associating faults to it is a complex operation requiring creation and update of several objects. For this reason, it was implemented as custom workflow activity, which runs in a transaction context so in case of error there are no partial updates to the database.

```
function scheduleMaintenance(){
    WebApiClient.Requests.kau_ScheduleMaintenance = WebApiClient.Requests.Request.prototype.with({
        method: "POST",
        name: "kau_ScheduleMaintenance",
        bound: false,
        entityName: "none"
    });

    var request = WebApiClient.Requests.kau_ScheduleMaintenance
        .with({
            payload: {
                CustomerAsset: {
                    "@odata.type": "Microsoft.Dynamics.CRM.msdyn_customerasset",
                    msdyn_customerassetid:maintenanceListModel.get("id")
                }
            }
        });
    WebApiClient.Execute(request).then(function (response) {
        alert ("Huolto aikataulutettu");
        return response;
    }).catch(function (error) {
            alert(error);
    });
}
```

Figure 18. JavaScript code to call custom workflow activity from the user interface

Adding code to the solution

Custom workflow activities were first defined using the solution editor. The implemented messages were then registered using Plugin Registration tool that uploads the assembly .dll file to the server and registers the messages and their properties. The plugin and the message registrations were then added to the solution using solution editor. The plugin Registration tool user interface is shown in Figure 19.

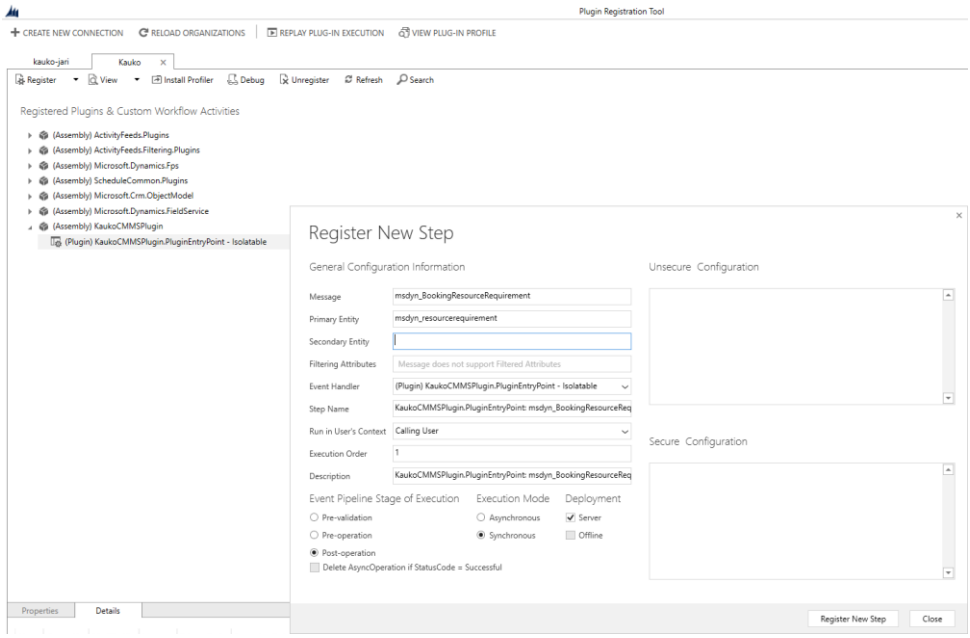


Figure 19. Plugin registration tool

Plugin Registration tool can also be used to debug plugins. Developers can use it to install a profiler component to the server and capture trace data from plugin execution and then replay it using the tool and then use Visual Studio for debugging.

5.6 Security Roles

Dynamics 365 allows very comprehensive access control to data, UI components, and system functionality. Access control is role-based and each user can be assigned multiple roles. For solution developers, it is essential to define the security roles but setting the exact permissions is typically done when implementing the solution to a specific customer. For the prototype, the security roles shown in Table 14 were defined.

Table 14. Security Roles

Role	**Allowed functionality**
Maintenance Technician	Accessing asset information, logging operating ours, performing work orders. Managing faults.
Equipment operator	Accessing asset information, logging operating hours, logging faults.
Foreman	Accessing asset information, scheduling work orders, performing work orders, Managing faults.
Business manager	Accessing asset information, accessing cost and time usage, accessing reports.

Dynamics 365 includes by default several roles typically used in a business setting. Often these should be extended with solution specific permissions rather than creating new roles, but in the prototype, case the existing roles were not a match, so new roles were created.

6 Testing and Evaluation

This section evaluates the solution against the objectives of this thesis defined in Section 1.2. The evaluation consists of functional testing to see if the system meets the requirements defined in Section 4.2 and a technical review to analyze how well the selected technologies performed and how well the implementation fit into the Dynamics 365 solution model. The user interface of the completed prototype running inside Dynamics 365 is shown in Figure 20.

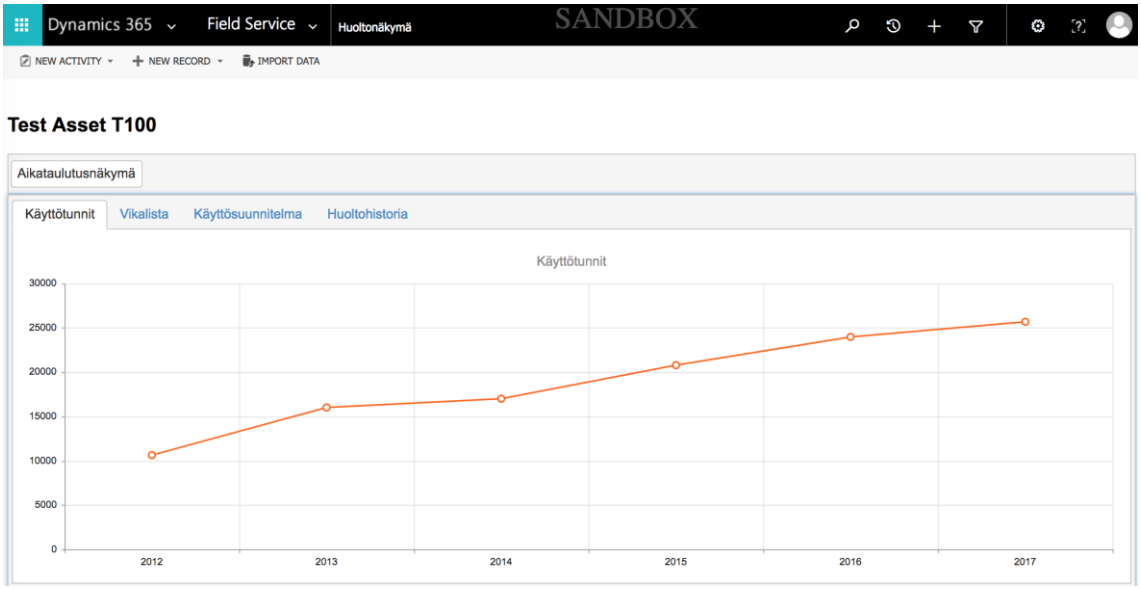
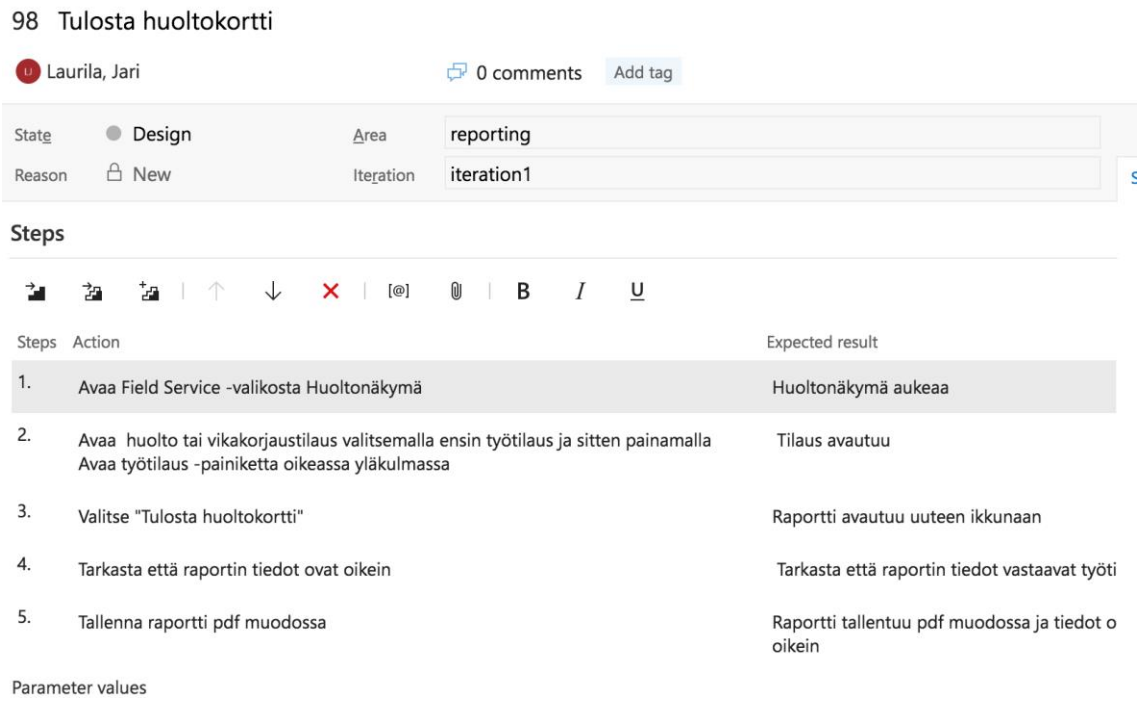


Figure 20. Finalized prototype

Functional testing

Functional testing was done by creating a test plan that consisted a set of test cases that cover the functional requirements of the system. Each test case consisted of a user story that defines what a user wants to accomplish, steps needed to perform the test, the required input data and expected results. Since the implementation project is a prototype, the focus of the test cases was on positive testing that tests that the system

does what it is supposed to do with valid inputs. For the same reason, the number of test cases was limited to cover only the essential functionality. An example test case that shows the steps and expected results of printing out work order instructions in PDF format is shown in Figure 21. The test cases are written in Finnish due to the testing team being Finnish.


98 Tulosta huoltokortti

Laurila, Jari 0 comments Add tag

State Design Area reporting

Reason New Iteration iteration1

Steps

Steps	Action	Expected result
1.	Avaa Field Service -valikosta Huoltonäkymä	Huoltonäkymä aukeaa
2.	Avaa huolto tai vikakorjaustilaus valitsemalla ensin työtilaus ja sitten painamalla Avaa työtilaus -painiketta oikeassa yläkulmassa	Tilaus avautuu
3.	Valitse "Tulosta huoltokortti"	Raportti avautuu uuteen ikkunaan
4.	Tarkasta että raportin tiedot ovat oikein	Tarkasta että raportin tiedot vastaavat työti
5.	Tallenna raportti pdf muodossa	Raportti tallentuu pdf muodossa ja tiedot o oikein

Parameter values

Figure 21. An example test case

After the test plan was ready, the actual testing was done using the following process:

1. A test instance for Dynamics 365 was provisioned from the cloud. The instance was configured to use Finnish as the base language and Euro as the currency. The instance was provisioned with Field Service application already installed.

2. The test solution was imported as a managed solution

3. A fictional company operating heavy equipment was set up. For testing asset data was entered from a market leading equipment manufacturer for excavators, drills, dozers, and trucks. Based on the publicly available information, preventive maintenance programs including the maintenance tasks and spare parts were entered.

4. Test users were created, and security roles were assigned to them
5. Test cases were run
6. Results were recorded

The tests were run several times during the project and based on the findings the prototype functionality was changed and errors fixed. When all test cases were successfully passed the system was considered to meet the defined functional requirements.

Technical review

Overall, the technologies used in the implementation fit the Dynamics 365 extendibility model well and all the key extendibility points of Dynamics 365 were used to craft the solution, so the study gave a good insight to extending Dynamics 365. Key findings of the development are described below.

Schema modifications proved to be very straightforward. The standard schema fits the requirements very well and required only a handful of changes. The only inconvenience in developing the schema is versioning: It is not straightforward to roll back schema changes when working with unmanaged solutions since removing the objects from the solution do not remove them from the database.

The development of web resources with the selected JavaScript libraries worked well. Building complex pages with a lot of data require many queries to the server. To increase page load performance, it was found to be beneficial to bundle queries and to use batch mode, which resulted in fewer roundtrips to the server. With batch mode, it was also possible to perform several data modifications inside one transaction. Another optimization point was to modify queries to return only the minimum attributes needed and to perform the queries in asynchronous mode when possible to minimize the time the single threaded JavaScript code is blocking user interface updates.

Building back-end code was made a bit more challenging due to cloud-based server not allowing direct access to debugging interface. This was helped by using a mockup library to allow creating unit tests that mimic server operations.

7 Discussion and Conclusions

The objective of this thesis was to design a prototype of a Computerized Maintenance Management System using Dynamics 365. The work started with a background research to understand the domain and to gather requirements for the prototype. Dynamics 365 as an application platform was studied and a technical solution to implement the requirements was found and the prototype was created. The prototype was then proven successful by the functional testing and the technical review.

Dynamics 365 is a powerful platform for create applications. Developers got a lot of functionality out of the box and can create complex business applications with relatively small development effort. But Dynamics 365 is also very large and complex and offers many possible ways to solve a use case. When choosing development strategy for customizations, it is best to understand how the platform works and implement features in a standard way. Some of the lessons learned in the project are:

- 3rd party libraries and tools used in the project helped cut development time. Especially the build system for web resources helped by enabling the use of modern JavaScript that is packaged into browser compatible bundles. Many great tools are available open source for Dynamics 365 development activities: solution development, testing and building. Their use is highly recommended.

- Kendo UI is a powerful way to build user interfaces for Dynamics 365. Support for Single Page Applications, MVVM-architecture and the various user interface components produce modern web pages that meet the user's expectations.

- For performance reasons the amount of data retrieved with queries should be kept as small as possible. Complex queries take a lot of time to execute and require a lot of bandwidth to return the results to browser. When possible, multiple queries should be bundled together to reduce the number of roundtrips to the server.

- Some of the data i.e. asset type list is slowly changing and should be cached to browser local storage to reduce the number of queries to the server.

- Transaction context is available in many operations and it should be used when possible to enable rollback in case an error happens in the middle of a complex operation

- Plugin development can be complex, since there are many event parameters and possibly system and 3rd party components are hooked to the same event. In many cases it might be a better idea to use custom workflow activities than hooking up complex code to events that are called often (i.e. Update message is called every time a single record changes, which happens very often in a large system with many users).

As the result of the study, the project gave the case company valuable information about developing Dynamics 365 applications and a lot of the code created in the prototype is applicable to future projects. The results of the prototype project also resulted in development process and tool improvements in the case organization:

- a continuous integration (CI) environment was taken into use to automate build process. The cloud based solution listens to changes in solution components and automatically builds redistributable solution packages when a change in any component happens.

- Visual Studio Team services was taken into use for agile development and testing. It enables tracking of work items, bugs and test cases.

- Scribe online was taken into use for migrating data between Dynamics 365 instances.

Based on the findings in this study, the case company also understands maintenance and maintenance management better and can apply the lessons learned to customer projects.

References

Campbell, J. D. (2016). *Uptime, 3rd Edition.* CRC Press.

CEN. (2010). *CEN - EN 13306:2010 Maintenance - Maintenance terminology.* European Committee for Standardization.

Fiix software. (2017, 11 16). *Preventive maintenance (PM)*. Retrieved from https://www.fiixsoftware.com/maintenance-strategies/preventative-maintenance/

Infotivity. (2017, 11 27). *What is Fit-GAP Analysis?* Retrieved from http://www.infotivity.com/fit-gap.html

Microsoft. (2017, 11 22). *Dynamics 365 application platform*. Retrieved from https://msdn.microsoft.com/en-us/library/mt706473.aspx

Microsoft. (2017, 11 32). *Dynamics 365 Process architecture*. Retrieved from https://msdn.microsoft.com/fi-fi/library/gg309387.aspx

Microsoft. (2017, 11 23). *Event execution pipeline*. Retrieved from https://msdn.microsoft.com/fi-fi/library/gg327941.aspx

Microsoft. (2017, 11 22). *Extend Microsoft Dynamics 365*. Retrieved from https://msdn.microsoft.com/fi-fi/library/gg327974.aspx

Microsoft. (2017, 11 22). *Introduction to Solutions*. Retrieved from https://msdn.microsoft.com/fi-fi/library/gg334576.aspx

Microsoft. (2017, 11 21). *Microsoft Dynamics 365: Intelligent Business Applications*. Retrieved from http://dynamics.microsoft.com/en-us

Microsoft. (2017, 12 9). *The metadata and data models in Microsoft Dynamics 365*. Retrieved from https://msdn.microsoft.com/en-us/library/gg309434.aspx

Microsoft. (2017, 12 4). *Use the Microsoft Dynamics 365 Web API*. Retrieved from https://msdn.microsoft.com/en-us/library/mt593051.aspx

Microsoft. (2017, 11 26). *Use the Xrm.Page object model*. Retrieved from https://msdn.microsoft.com/en-us/library/gg328474.aspx

Mobley, R. K. (2002). *An Introduction to preventive maintenace.*

Modern Materials Handling. (2017, 11 26). *The case for managing MRO inventory*. Retrieved from http://www.mmh.com/article/the_case_for_managing_mro_inventory

Mounla, R. (2017). *Microsoft Dynamics 365 Extensions Cookbook.* Packt Publishing.

Plant Services. (2010, 8 10). *Effective planning and scheduling*. Retrieved from https://www.plantservices.com/articles/2010/08assetmanager/

Wireman, T. (2013). *Succesfully Utilizing CMMS/EAM Systems.* Reliabilityweb.com.

Fit Gap Analysis

Category	Requirement	Fit level (1- poor to 5-great)	Fitting Dynamics functionality	Gaps	Implementation technology
Work Order Management	R1.1 Tracking planned and actual labor	5	Field Service: Work Orders		Customization
	R1.2 Tracking planned and actual materials	5	Field Service: Work Orders		Customization
	R1.3 Sortable work order backlog	5	Field Service: Work Orders		Customization
			Field Service: Schedule board		
	R1.4 Resourcing work orders to technicians	4	Field Service: Scheduling assistant		Customization
	R1.5 Managing work order status	4	Field Service: Work Orders		Customization
Preventive Maintenance	R2.1 Creating Preventive Maintenance Programs	4	Field Service: Incidents		Customization
	R2.2 Scheduling maintenance based on calendar time	5	Field Service: Agreements		Customization
				Lack of meter readings	
	R2.3 Scheduling maintenance based on meter reading	1		No scheduling support	Extension
	R2.4 Manual scheduling	3	Field Service: Schedule board	Lack of weekly and month	Extension
	R2.5 Forecasting labor, material and tool requirements	2	Reporting		Customization
	R2.6 Combining due Preventive Maintenance	1		no support	Extension
Asset Management	R3.1 Tracking equipment information	4	Field Service: Customer Assets		Customization
	R3.2 Saving work orders to equipment history	4	Field Service: Work Orders		Customization
	R3.3 Tracking cost and repair information on component level	3	Field Service: Customer Assets		Customization
	R3.4 Punch list management	1	Customer Support: cases		Extension
			Field Service: Products		
			Document management integration		
	R3.5 Bill of materials for each piece of equipment	3	(Sharepoint, Onedrive)		Customization
	R3.6 User defined screens for storing additional information about equipment	5	Field Service: Customer Assets		Customization
Inventory Management	R4.1 Tracking inventory on-hand	4	Field Service: Warehouses		Customization
	R4.2 Tracking unit price information	4	Dynamics: Products, Price Levels		Customization
	R4.3 Support for multiple warehouses	4	Dynamics: Warehouses		Customization
	R4.4 Support for warehouse to warehouse transfer	5	Dynamics: Warehouses		Customization
	R4.5 Generation of purchase orders	5	Dynamics: Purchase orders		Customization
Reporting	R5.1 Predefined reports	5	Dynamics views and charts		Customization
	R5.2 Custom report editor	5	Report Designer		Customization
	R5.3 Maintenance budget reporting module	1		no support	Customization
	R5.4 Tracking equipment downtime	1		no support	Customization

Custom Entity Definitions

Kau_faultList entity definition

Attribute	**Purpose**
kau_customerasset	identifies the asset
kau_faultcode	Category of the fault, enables classification and tracking
kau_startDate	Date when the fault was logged
kau_faultState	Option Set describing the state of the fault (new, fixed, checked)
kau_endDate	Date when the fault was fixed
kau_duration	Estimated / actual duration of fixing the fault
kau_msdyn_workorder	Work order where fault is fixed

Kau_faultCode entity definition

Attribute	**Purpose**
kau_faultCode	Alphanumberic fault code
kau_name	Name of the fault code
kau_description	More detailed description of the fault
kau_assetType	Identified the asset category that this fault can occur in.
kau_parentFaultCode	Parent fault code. Allows hierarchical fault code structure.

kau_usagePlan entity definition

Attribute	**Purpose**
kau_customerasset	identifies the asset
kau_startDate	Start date of the usage plan
kau_endDate	End date of the usage plan
kau_hoursPerDay	Estimated usage of the asset/day

kau_operatingHours entity definition

Attribute	**Purpose**
kau_customerasset	Identifies the asset
kau_measurementDate	Date of the measurement
kau_operatingHoursType	Option Set that defines what measurement this is
kau_hours	The measurement value