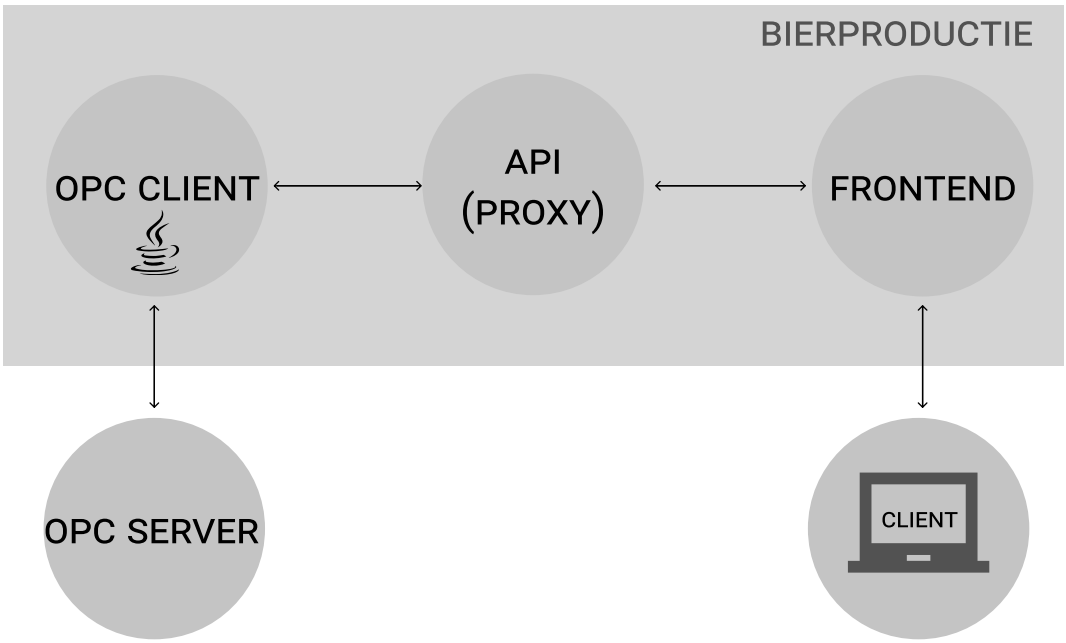
Bierproductie

A management system for brewing machines



Bachelor of Engineering, Software Technology

Semesterproject 3. semester, ST3-PRO

Project Period: 31.08.2020 - 19.12.2020

Hand in date: 19.12.2020

Group 06:

Jakob Rasmussen, jakra19@student.sdu.dk

Kenneth M. Christiansen kechr19@student.sdu.dk

Kevin K. M. Petersen, kepet19@student.sdu.dk

Kristian N. Jakobsen, kjako19@student.sdu.dk

Simon Jørgensen, sijo819@student.sdu.dk

Supervisor: Parisa Niloofar, parni@mmmi.sdu.dk

University of Southern Denmark
The Faculty of Engineering
The Mærsk Mc-Kinney Møller Institute
Campusvej 55, 5230 Odense M

Title:	Bierproductie
Institution:	University of Southern Denmark The Faculty of Engineering, The Mærsk Mc-Kinney Møller Institute Campusvej 55, 5230 Odense M
Education:	Bachelor of Engineering, Software Technology
Semester:	3. Semester
Course Title:	Industrial 4.0 cyber-physical software systems
Internal Course Code:	ST3-PRO
Project Period:	31.08.2020 - 19.12.2020
ECTS:	10 ECTS
Supervisor:	Parisa Niloofar
Project group:	06

Jakob Rasmussen, jakra19@student.sdu.dk

Kenneth M. Christiansen, kechr19@student.sdu.dk

Kevin K. M. Petersen, kepet19@student.sdu.dk

Kristian N. Jakobsen, kjako19@student.sdu.dk

Simon Jørgensen, sijo819@student.sdu.dk

Pages: 10
Appendix: 0

By signing this document, each group member confirms that everyone have participated equally to this project, and everyone is thus collectively responsible for the content of the report.

I Summary

II Table of Contents

III Editorial

IV List of Figures

1 Introduction

2 Background

3 Problem analysis

4 Theory & Methods

5 Requirements

5.1 Overall Requirements Specification

5.2 Selected Detailed Requirements

5.2.1 Functional & Non-Functional Requirements

5.2.2 The Physical Setup (The Brewery Machine)

5.2.3 The Simulator

5.3 Use Cases

5.3.1 Actor List

5.3.2 Detailed Use Cases

From project description

5.3.3 Use Case Diagram

6 Analysis

6.1 Use Case analysis

6.1.1 Class Candidates

In order to find potential class candidates, every noun of the detailed Use Cases are found. These are potential candidates, and can be sorted to avoid duplicates and candidates that won't be turned into classes. Naturally, every potential class for the entire system will not be found, as this only reflects use cases. A potential class candidate such as MES (where Start and Stop functionality would otherwise be implemented) will not be reduced to a single class and is therefore not added to the list of class candidates.
The final list of classes, as well as a description of them, can be seen in table 1.

Class Candidate	**Attributes**	**Definition**
Batch	Id, type, product_amount (total, defect, acceptable), amount (time), state (current, history), OEE, production_speed,	A batch refers to a specific batch of products the brewery has made
Product	Id, type, Ingredients,	Product refers to the different options of beer to be produced
Ingredient	Name, id	An ingredient refers to a specific ingredient. Products contain a list of ingredients.

Table 1: Potential class candidates

6.1.2 UML Analysis Diagram

From the verb/noun analysis from the previous chapter, the UML analysis diagram seen in figure 1, can be generated. This diagram shows the classes and attributes found in the requirements from the project description.

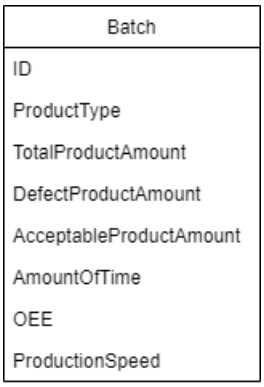


Figure 1: UML Analysis diagram

6.2 Use Case Realisation

6.2.1 Sequence Diagrams

A sequence diagram shows the system events for a given scenario of a use case, and how the actor interacts with the system to solve the use case. There are two kinds of sequence diagrams, system and operation. The system sequence diagram displays the system as a 'black box', where the internal system events are not shown, but only the external. This means that the diagram displays how actors generate system events and what the system output is. Furthermore, the diagram functions as a timeline for the system events.

maybe add a system sequence diagram and explain why we used it
The operation sequence diagram displays the system as a 'white box', where both the internal and external system events are described, as seen in figure 2.

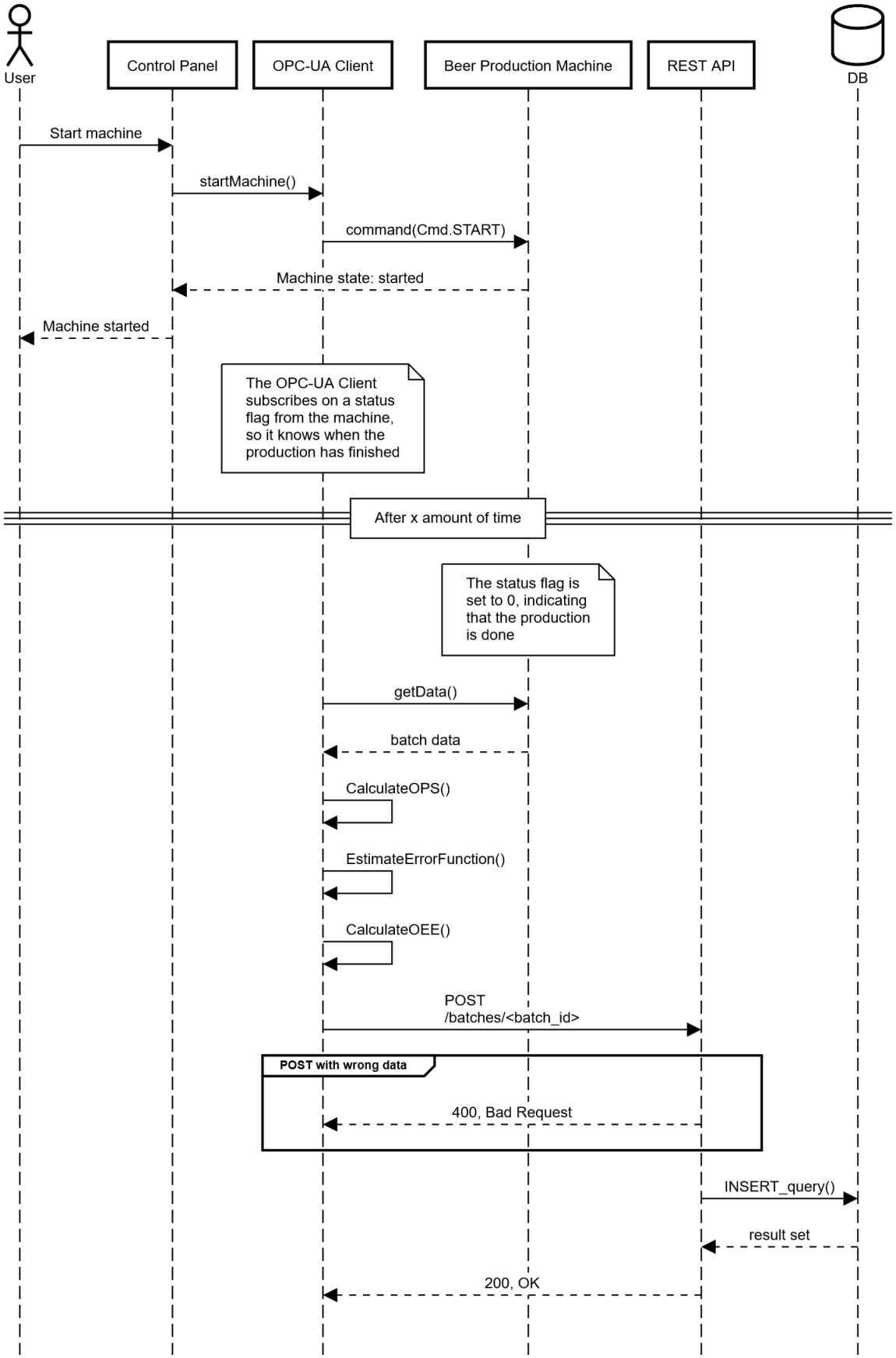


Figure 2: Sequence diagram: start

This sequence diagram is used to identify system functions, as the events shown in the diagram

are the functions needed to complete the use case. In this specific use case, the actor, the user, wants to start the beer production machine. The user interacts with the control panel by pressing the start button, which then sends a command to the OPC-UA client. The OPC-UA client interprets the command as a start machine command, which triggers an event in the OPC-UA client to send a command to the beer production machine. The beer production machine interprets the command as a start command, which then turns on the machine. As a response to the user, to beer production machine sets a flag which the control panel reacts to, and sends a message to the user.

When the beer production has finished, the OPC-UA client collects all relevant data from the beer production machine. This data is used to calculate the optimal production speed, estimate the error function, and calculate the OEE. These calculation are used to optimise the beer production. The calculated data and the data collected from the machine is then stored in a database. This happens through a REST API which acts as a translator between the different subsystems within the MES.

6.2.2 Operation Contracts

An operation contract describes the responsibility of the operation. The contract focuses on what the operation can change, and not how it is changed. It is also used to describes the state of the system before and after the operation is called.

start	
System operation	start
Cross References	Use case: Start machine see table **??**
Responsibility	Starting the beer machine if the pre-conditions is met. If the pre-conditions is not met, the beer machine will not start
Output	The beer machine started the production
Pre-conditions	The beer production machine needs to be in ready mode, that is, not producing beer.
Post-conditions	The beer machine started brewing

Table 2: Operation Contracts start

stopProduction	
System operation	stopProduction
Cross References	Use case: Stop the beer Machine see table **??**
Responsibility	Stop's the beer machine if the pre-conditions is met. If the pre-conditions is not met, the beer machine will not do anything
Output	The beer machine is stopped
Pre-conditions	The beer machine needs to be running
Post-conditions	The beer machine is stopped

Table 3: Operation Contracts stopProduction

reset	
System operation	reset
Cross References	Use case: reset see table **??**
Responsibility	It is responsible for resetting the beer machine.
Output	reset the beer machine.
Pre-conditions	The beer production machine needs to be in ready mode, that is, not producing beer.
Post-conditions	The beer production machine has been reset.

Table 4: Operation Contracts reset

clear	
System operation	clear
Cross References	Use case: clear see table **??**
Responsibility	It is responsible for clearing the beer machine.
Output	The beer machine has been cleared.
Pre-conditions	The beer production machine needs to be in ready mode, that is, not producing beer.
Post-conditions	The beer production machine has been cleared.

Table 5: Operation Contracts clear

display live data	
System operation	displayLiveData
Cross References	Use case: displayLiveData see table **??**
Responsibility	It is responsible for posting data to the client.
Output	Post data to the client.
Pre-conditions	The beer production machine needs to be on and producing beer.
Post-conditions	Live data has been displayed for the user.

Table 6: Operation Contracts monitorAndDisplayData

batchReport	
System operation	batchReport
Cross References	Use case: batchReport see table **??**
Responsibility	Make a report after the pre-conditions is met and adds the report to the database.
Output	Produces a batch report and display it for the user.
Pre-conditions	The beer Machine needs to have produced a batch.
Post-conditions	A batch report has been displayed for the user.

Table 7: Operation Contracts produceBatchReport

6.2.3 Updated UML Class Diagram

The updated UML class diagram illustrates the current system idea based on the analysis of the system. Although this diagram only shows the OPC-UA client, it still gives a good idea of

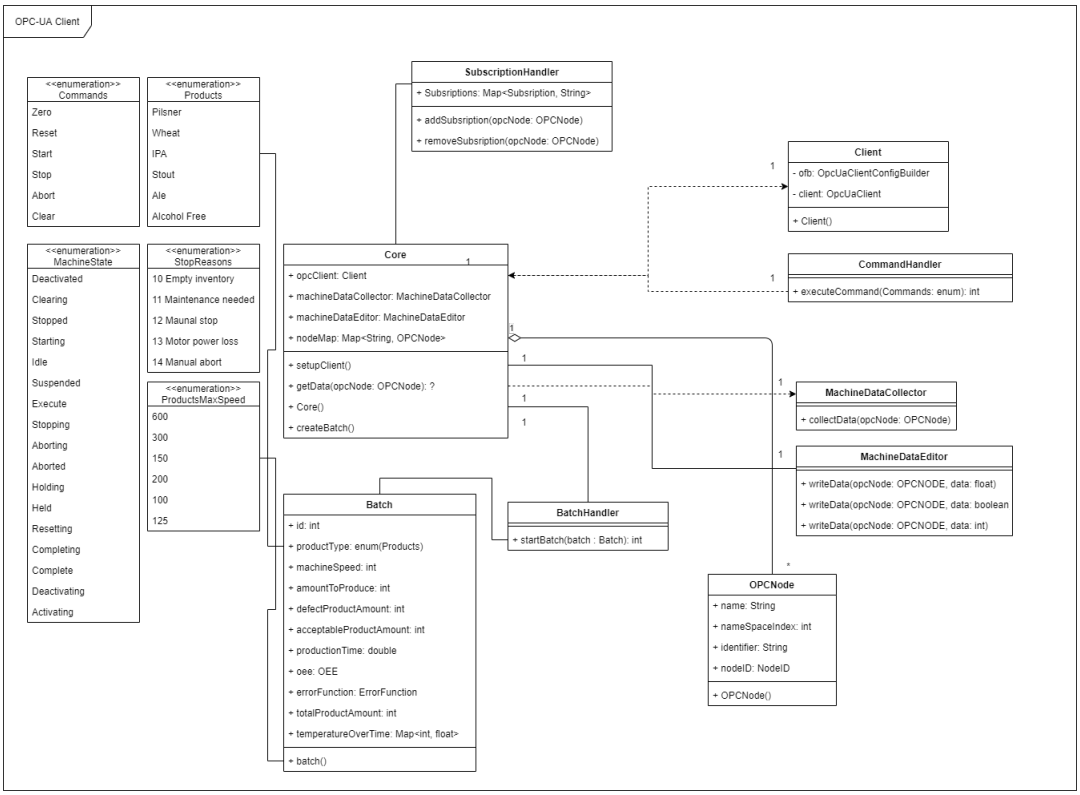


Figure 3: Updated UML Class Diagram

how this part of the system is going to be, once implemented.

By using the diagram in the implementation phase, the group has a good starting point to expand on. The classes in the UML class diagram have a chance of not being implemented if the group finds them unuseful or changed to adhere to the program.

7 Architecture

8 Design

9 Implementation

10 Verification & Validation

11 Evaluation

12 conclusion