Collaboration as a Service

Citation for published version (APA):

Document status and date:
Published: 06/10/2021

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:
• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 27. Sep. 2023
Collaboration as a Service
Extending the capabilities of the Model as a Service platform for the Internet of Food

Niels Rood
10/2021
Department of Mathematics & Computer Science
Collaboration as a Service
Extending the capabilities of the Model as a Service platform for the Internet of Food

Niels Rood

October 2021

Eindhoven University of Technology
Stan Ackermans Institute – Software Technology

PDEng Report: 2021/071

Confidentiality status: Public

Partners

Sustainable Food Initiative and Internet of Food Consortium including:
- Wageningen University & Research
- Unilever Research & Development
- NIZO Food Research BV
- ISPT Foundation
- Symrise AG

Steering Group
Prof. dr. Jakob de Vlieg
Dr. Yanja Dajsuren
Dr. Michiel Gribnau
Dr. Rogier Brussee

Date
October 2021
Composition of the Thesis Evaluation Committee:

Chair: Prof. dr. Jakob de Vlieg
Members: Dr. Yanja Dajsuren
Dr. Michiel Gribnau
Dr. Rogier Brussee
Ing. Sicco Pier van Gosliga MSc
Dr. Egor Bondarau

The design that is described in this report has been carried out in accordance with the rules of the TU/e Code of Scientific Conduct.
The Sustainable Food Initiative (SFI) wants to improve the sustainability of the food industry by exploring innovative technologies. They developed the Model as a Service (MaaS) prototype: a model-sharing platform that allows organizations to share models according to the FAIR principles, without giving them away. This platform increases the interoperability between models and data from different organizations, lowering barriers to facilitate inter-organizational collaboration. Such collaborations can reduce the impact of high development costs of food models, as expensive proprietary models can be shared across chain partners. The MaaS prototype set a baseline for an effective model-sharing platform. However, its sharing capabilities are limited to one use case, and its design makes adding new use cases difficult. The Collaboration as a Service (CaaS) project has extended the MaaS prototype such that it is more flexible, supporting more use cases. The resulting CaaS platform supports three new use cases and is set up such that new use cases can be added in the future. The platform uses ontologies to add meaning to models and data, increasing their interoperability.

Keywords
Food, Sustainability, Modeling, FAIR, Ontology, Sharing, Collaboration
no warranty, representation or undertaking whether expressed or implied, nor does it assume any legal liability, whether direct or indirect, or responsibility for the accuracy, completeness, or usefulness of any information.

Trademarks
Product and company names mentioned herein may be trademarks and/or service marks of their respective owners. We use these names without any particular endorsement or with the intent to infringe the copyright of the respective owners.

Copyright
Copyright © 2021. Eindhoven University of Technology. All rights reserved. No part of the material protected by this copyright notice may be reproduced, modified, or redistributed in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the Eindhoven University of Technology and TKI-SFI Internet of Food Consortium.
Foreword

A global challenge to meet future food supply
The current global agriculture revenue is approximately $4.8 trillion and only gets larger. If trends continue, caloric demand will increase by 70 percent in 2050 and crop demand for human consumption and animal feed will increase by at least 100 percent. However, at the same time, food production must reduce its ecological footprint and become sustainable in the very near future! Meeting this demand won’t be easy! According to the World Economic Forum a systemic transformation is needed to deal with the fast-growing demand for high quality food products while meeting the challenges imposed by sustainability goals. Science and technology are increasingly playing a role to implement the required changes to produce enough healthy, tasty and affordable food.

AgriFoodTech to drive sustainable and more efficient food production
As a result, a new science and research area is developing: the interdisciplinary field of AgriFoodTech, i.e., the crossovers between the agri & food domain and high-tech & digitization. The agri-food sector starts to benefit from new disruptive and interdisciplinary technology developments, e.g., in the field of life sciences, sensing & vision technologies, artificial intelligence (A.I.), and robotics and has become extremely data intensive. For example, there is a lot of potential value in data integration between the various steps and activities within the food value chain. Total integration of data and data analysis throughout the entire chain, from seeds and fertilizers, towards feedstock, livestock, agriculture, food processing and retail. As it stands, this is a moon-shot project and aimed at several long-term “Data Inside and Digitization” challenges in the food sector. Nevertheless, chain integration and consumer-driven chain reversal are becoming achievable by digital technologies and they might rewrite entire innovation and business models in the food sector.

Sustainable Food Initiative and Internet of Food Consortium
This PDEng research project of Niels Rood is part of the Sustainable Food Initiative (SFI). In this initiative, several companies and knowledge institutes teamed up to produce healthy and safe food products at the lowest possible environmental footprint. An important goal of the SFI program is to develop a shared roadmap for digitally enhanced innovation. This involves among others speeding up food related R&D and developing sustainable, flexible, and precise food production processes by fully exploiting modern sensing, vision and digitalization technologies, e.g., computer simulations, data fusion and advanced A.I.

The Internet of Food (INoF) consortium, which is part of SFI, aims to address the future food safety challenges by developing and exploring advanced data science solutions to make production processes more efficient and sustainable. Inter-organization collaboration is one of the main drivers of data-driven innovation by building on the strength of different organizations and the infusion of new ideas. Moreover, modern data fusion and smart data analytics technologies can reduce high development costs by sharing and reusing expensive proprietary food models in a secure way across chain partners. As anyone involved in cross organizational collaboration can attest, it is not without organizational, practical hurdles such as IT problems and miscommunications. Cross-type collaboration requires an appropriate and effective digital infrastructure that can maintain interoperability among diverse data formats from different sources, reducing data loss and miscommunication. An important condition for success is the ability to share data, models and services between companies without the need to share IP (Intellectual Property) or replicate corresponding execution environments.

Models as a Service (MaaS) and the follow-up Collaboration as a Service (CaaS) platform
In the first phase of the Internet of Food project Niels predecessor, Hossain Muctadir developed a platform prototype based on a layered, microservice architecture and FAIR (Findable, Accessible, Interoperable and Reusable) computer science principles. This previous research project was important to set up a baseline for an effective discovery platform. To achieve interoperability among various data sources and food models dynamic model parameter mapping and on-demand unit conversion were implemented in the first prototype. In this next project phase, Niels upgraded the prototype and software
architecture to permit sharing of arbitrary data and models in a much more general way by using a so-called hub and spoke approach with a thin hub. An architecture where both data and models describe themselves using innovative ontology-based concepts. Niels’ follow-up CaaS platform, extends sharing models by guaranteeing interoperability and (semi)automated integration between several food models and data sources in machine readable formats. It does this by using the ontology-based descriptions to adapt data sources, or the output of models, to a form that the models expect as input. In this way the new CaaS platform, users can expose and once exposed, access data and models from other companies intuitively, removing unwanted inter-organizational barriers such as different conventions for units, different formatting, different names, or codes for food ingredients. The ontology concepts were developed in close collaboration with Rogier Brussee of the Jheronimus Academy of Data Science (JADS & M&CS) and Hajo Rijgersberg and Gorkem Simsek-Senel of the Wageningen University.

**Creating value in practice by CaaS**

Besides extending the MaaS prototype and the underlying software architecture to exchange cross-organizational proprietary food models, Niels validated model and data exchange of the CaaS platform in real life situations. In close collaboration with other INoF team members he developed demos to show the practical use and benefits of data and model sharing and how this can create potential value in terms of data-driven innovation models. In these demonstrations, available models for nutrition, taste, pasteurization, and droplet size (sensor concepts) were combined to demonstrate that the new ways of “supplying data to models” in CaaS might speed up product development and can save costs by creatively connecting proprietary models in context of new product challenges and applications. Moreover, the flexibility of the CaaS platform allow combining computer simulations of two or more individual models to discover new scientific insights. Based on the demonstrations it becomes clear that the CaaS platform and the previous MaaS concept extends the potential to use computer models as actionable knowledge in the food industry.

**Advancing to the next phase**

The Maas and CaaS projects were mainly focused on developing the underlying software and data science technologies for data and model sharing and to prove that advanced and creative model exchange can create new value in terms of innovation. Discussions on developing a follow-up project exploring additional real-life implementations and digital technology developments are ongoing. The implemented digital infrastructure is sufficiently generic to support other chain integration & consumer-driven chain reversal opportunities in the food sector and beyond. Potential applications are e.g., in silico food product development, lights-out factories with sensor-based automated monitoring, and data-driven quality control using text mining and machine learning on vocabulary of sensory panels.

**Great Job and good teamwork**

Niels, you have done a great job, especially considering the difficult circumstances you had to work in due to the corona measures. From a platform that essentially could run one model, to a FAIR-based platform that can handle different data and models in a plug and play way. You are a good team player and worked closely together with scientists from WUR, TU/e, JADS, NIZO and Unilever. You also did an extensive problem analysis and demonstrated full knowledge of the complex Internet of Food topic at the software engineering level. Thank you very much for your excellent work and achievements to make the SFI- INoF moonshot project a success. I wish you all the best for your future career!

*Prof. dr. Jakob de Vlieg*  
Chair of the Applied Data Science (ADS) research group, M&CS, TU/e  
Lead AgrifoodTech at TU/e and JADS

Date: 14-09-2021
Preface

This report is written as part of the project *Collaboration as a Service - Extending the capabilities of the Model as a Service platform for the Internet of Food*. The report presents the activities and results of the Collaboration as a Service (CaaS) project. The CaaS project is my final graduation as the partial fulfillment of my graduation for the Professional Doctorate in Engineering (PDEng) in Software Technology (ST) program. The PDEng ST program is a two-year doctoral level technical designer program offered by the Stan Ackermans Institute at the Eindhoven University of Technology.

The CaaS project was executed in the context of the Internet of Food (INoF) project, commissioned by the Sustainable Food Initiative (SFI), who are also the clients of the CaaS Project. The author extended upon the previously developed Model as a Service (MaaS) prototype. The result is a more flexible platform, which supports all the use cases relevant to the INoF project.

Chapters 1, 2, and 3 provide an overview of the context and goals of the project. Chapters 4 and 5 describe the solutions that were created as part of the project. Chapters 6, 7, and 8 reflect on the results and process through which the results are achieved.

This report serves multiple purposes. It is first of all an overview of my graduation project, serving as a means of assessment of me by the Thesis Evaluation Committee (TEC) of the CaaS project. All chapters are relevant for this purpose since the combination of all aspects of the project provides a complete overview of my efforts. It can also be used as a reference for the client to gain a deeper understanding of the results of the project. For this purpose, Chapters 4, 5, and 8 are interesting in particular, as they focus on the project results. Lastly, the report can also be relevant to anyone interested in the technical developments within the food industry. Chapters 1, 2, and 8 are relevant for this audience since these chapters explain the project in the context of the food industry.

Niels Rood
September 2021
Acknowledgments

Many people have supported me throughout the execution of this project. These people have helped me perform at my maximum potential, and guided me through the times when things did not go as planned.

I would like to thank Jakob de Vlieg, my main academic supervisor, for providing much-appreciated input on the direction of the project, both from an industrial perspective and from an academic perspective. You helped me define the goals of the project, always finding new connections that could be made between my project and other projects you are involved with. These connections put my project into perspective, helping me improve my designs to fit in its context.

Many thanks also to Michiel Gribnau, my company supervisor, for showing me the value of my project in the context of the food industry. Your knowledge and examples of the applications of in-silico experimentation within the food industry helped me figuring out how I could optimally support these processes with the results of my project.

I would also like to thank Yanja Dajsuren and Désirée van Oorschot, who have helped me not only during my graduation project but also throughout my entire time in the PDEng ST program. You helped me keep on track towards graduation, especially when I was unsure how to proceed. Yanja even took upon herself the role of secondary academic supervisor for my final project, guiding me on improving my project from an academic perspective.

I am very thankful for the extensive collaboration I had during my project with Rogier Brussee and Hajo Rijgersberg, without whom the project would not be nearly as successful as it is now. Rogier, you brought all of your expertise and ideas for the future of the Internet of Food project, us together forming the fundamental concepts that supported many technical decisions in my project. Hajo, your experience with ontologies grounded our broad brainstorming sessions to result in a practical ontology vocabulary.

I am thankful to have been a client for the Flavouride Software Engineering Project (SEP). Many thanks to all the bachelor students for their dedication to make the Flavouride project a success, despite my inexperience in the role of SEP client.

Many thanks to Hossain Muhammad Muctadir, who conducted the Model as a Service project. Your groundwork and precise documentation gave my project a jump start, as I could quickly delve deep into the inner workings of your results.

Special thanks go to my parents, who did everything they could to allow me to focus maximally on my graduation project, which sometimes required taking some time to do something else together, allowing me to continue with a fresh mind.

I would also like to thank my PDEng ST colleagues, with whom I have shared our successes and hardships during our time working together. You have always been very open and inclusive, which gave me a lot of confidence. We have become more than just colleagues in this time, rather we have become friends for the years to come.

I am also thankful for all the professors and teachers who taught the courses and workshops that are part of the PDEng ST program. You have provided me with the knowledge and skills required to make this project a success.

Niels Rood
September 2021
Executive Summary

Under the Sustainable Food Initiative (SFI), multiple parties inside and outside the food industry want to improve the sustainability of the food industry by exploring innovative technologies. One such technology is in-silico food product development. To accelerate this kind of development, the SFI started investigating the possibilities for collaboration under the Internet of Food project. This resulted in the Model as a Service (MaaS) prototype: a model sharing platform showing the potential of sharing model results between organizations through one use case example. This example allows users to share and use a model that calculates the taste properties of a food product based on its ingredients.

The MaaS prototype has some shortcomings, which limit its applicability. Mainly, it was made such that it only supports one use case. The MaaS prototype also focuses on the sharing of models, while the possibilities for sharing data are limited.

The main goal of this project is to build upon the foundations set by the MaaS prototype. An updated prototype is created that mitigates the shortcomings of the MaaS prototype. This new version of the prototype is called the Collaboration as a Service (CaaS) platform. It takes a data-centered approach, changing the way data and metadata are used in simulations. Data sources are exposed through gateways in the same way that models are shared in the MaaS prototype. The gateways are also supplying metadata for the models or data sources they represent. Data source contents and model interfaces are both described in terms of data tables. This approach enables more use cases since any data source can be connected to any model input. Two new use cases are implemented to work with the CaaS platform.

As a part of this project, I was a client for a Software Engineering Project (SEP). In this SEP, ten Bachelor students comprising the Flavouride team were tasked to improve the usability of the MaaS prototype. During their ten weeks of work, these students made multiple improvements to the MaaS prototype. Their results have been partly integrated with the CaaS platform.

The CaaS platform was demonstrated to members of the Internet of Food project, with a positive response. The CaaS platform can be used to gather more partners to collaborate under the SFI or Internet of Food project. Just like the MaaS prototype, it can also serve as a starting point for a commercial product for collaboration in the food industry.
Table of Contents

Foreword................................................................................................................................................. i
Preface................................................................................................................................................... iv
Acknowledgments ................................................................................................................................ vi
Executive Summary ........................................................................................................................... viii
Table of Contents ................................................................................................................................. ix
List of Figures ...................................................................................................................................... xii
List of Tables ...................................................................................................................................... xiv

1. Introduction ..................................................................................................................................... 17
   1.1 Project context............................................................................................................................. 17
   1.2 Outline ......................................................................................................................................... 17

2. Problem Analysis ............................................................................................................................ 19
   2.1 Domain Analysis.......................................................................................................................... 19
      2.1.1. Food Production Domain ..................................................................................................... 19
      2.1.2. Data and Model Sharing Domain ........................................................................................ 20
   2.2 Problem Definition ...................................................................................................................... 21
      2.2.1. Tomato Soup Taste Use Case .............................................................................................. 22
      2.2.2. Nutritional Value Use Case ................................................................................................. 22
      2.2.3. Pasteurization-Shelflife Use Case ........................................................................................ 23
      2.2.4. Calibration-Droplet size Use Case ....................................................................................... 23
      2.2.5. Optimization Experiment Use Case ..................................................................................... 24
   2.3 Project Goals ............................................................................................................................... 25
   2.4 Assumptions and Constraints ...................................................................................................... 25

3. Requirements Elicitation ................................................................................................................ 27
   3.1 Elicitation process ....................................................................................................................... 27
   3.2 Use case scenarios ...................................................................................................................... 27
   3.3 Requirements ............................................................................................................................... 28

4. System Architecture and Design .................................................................................................... 31
   4.1 Design Decisions.......................................................................................................................... 31
      4.1.1. Data Storage and Exchange ................................................................................................. 31
      4.1.2. Data format .......................................................................................................................... 32
      4.1.3. Metadata format ................................................................................................................... 33
      4.1.4. Metadata Exchange .............................................................................................................. 34
   4.2 CaaS Architecture ....................................................................................................................... 34
      4.2.1. Gateways ............................................................................................................................. 36
      4.2.2. Sharing Platform .................................................................................................................. 37
List of Figures

Figure 1: Domain analysis diagram of the Food Product Development Domain ........................................... 20
Figure 2: Domain analysis diagram of the Data and Model Sharing Domain .................................................. 21
Figure 3: Situation schema of the Tomato Soup Taste use case. Company A has information about food ingredients and wants to know the taste of the resulting food product. Company B has a model that takes ingredients of tomato soup and returns the resulting taste. The model only supports ingredients known by Company B ........................................................................................................... 22
Figure 4: Situation schema of the Pasteurization-Shelflife use case ............................................................ 23
Figure 5: Situation schema of the Calibration-Droplet size use case ........................................................... 24
Figure 6: Use Case diagram. Yellow use cases have been modified compared to the previous MaaS project, while orange use cases have been added for this project ....................................................... 28
Figure 7: An example of a data table with its describing data schema .......................................................... 33
Figure 8: MaaS prototype component diagram ............................................................................................ 35
Figure 9: Block Definition Diagram of the top-level block decomposition of the CaaS platform ................. 36
Figure 10: Activity Diagram of the Data Gateway sharing process .............................................................. 37
Figure 11: Internal Block Diagram of the Sharing Platform block inside of the CaaS platform ................. 38
Figure 12: Activity Diagram for defining a simulation .................................................................................. 38
Figure 13: Example of column-based simulation bindings ............................................................................ 39
Figure 14: Sequence Diagram of the simulation execution process ............................................................. 41
Figure 15: Deployment Diagram of the MaaS prototype (derived from [1]) ................................................ 44
Figure 16: My Food Products view of the MaaS prototype ........................................................................ 45
Figure 17: Frontend pages (as presented in [1]) .......................................................................................... 46
Figure 18: Example of JSON format representing a data table ................................................................. 47
Figure 19: Example TURTLE file representing an ontology ....................................................................... 47
Figure 20: Class Diagram of the implementation of the model classes ...................................................... 48
Figure 21: CSV file and JSON data schema configuration for a recipe ....................................................... 49
Figure 22: Work Breakdown Structure of the CaaS project ...................................................................... 53
Figure 23: Global timeline of the project, showing the phases, milestones, and pacing .............................. 54
Figure A.24: Structural overview of stakeholders of this project. The diagram shows both the organization and work package each stakeholder is a part of ................................................................. 66
List of Tables

Table 1: Functional requirements that are added to the MaaS functional requirements to fit the CaaS project goals .......................................................... 29
Table 2: Decision criteria for data storage challenge ................................................................. 32
Table 3: Comparison of metadata-sharing storage methods ........................................................ 34
Table 4: Integration Test Results ............................................................................................... 51
Table 5: Milestones of the CaaS project .................................................................................. 55
Table 6: Risks and Risk Mitigation strategies ......................................................................... 55
Table A.7: Stakeholders of the project, grouped by company .................................................. 65
Table B.8: Use case Scenarios .................................................................................................. 67
Table C.9: Functional system requirements for this project .................................................... 69
Table C.10: Non-functional system requirements for this project ......................................... 70
Table C.11: Functional requirements for SEP assignment ..................................................... 70
1. Introduction

1.1 Project context

The population of the world is growing continuously, implying the need to continuously increase food production as well. Given the limited availability of fertile land and other resources and the changing climate conditions, this has become a serious challenge, asking for new and innovative approaches. The Sustainable Food Initiative (SFI) is an initiative composed of food industry partners and academia seeking to figure out ways to satiate the increasing need for food through sustainable innovation.

Within SFI, the Internet of Food (INoF) project was started to speed up innovation within the food industry by exploring advanced digital technologies, with the focus on collaboration-supporting technologies and sensor-based technologies. Efficient and trusted exchange of information by partners in the food value chain will help to maximize the value of the food value chain, whereas sensor-based technologies are essential for sustainable and resource-efficient food production. The key partners that work together in the INoF project are Unilever, NIZO, Symrise, Eindhoven Technical University (TU/e), and Wageningen University and Research (WUR). This PDEng graduation project, Collaboration as a Service (CaaS), is part of the Internet of Food project. Hence, this project also has stakeholders from each of these organizations. A complete overview of these stakeholders is provided in Appendix A.

The INoF project is split into multiple Work Packages, each focusing on different aspects of this problem. The CaaS project is part of Work Package 1, but it also incorporates results from Work Package 2.

Work Package 1 (WP1) aims to improve collaboration in food product development between companies. The goal is to define and set up a data and model sharing architecture that makes data related to product development Findable, Accessible, Interoperable, and Reusable (FAIR data). Last year, a microservice architecture was defined and implemented that allows users to 1) share food products and 2) share models that compute properties about these food products. The implementation is called the Model as a Service (MaaS) prototype [1]. The prototype includes a model ontology for automatic matching between food product properties and model inputs. It also uses the Ontology of units of Measurements (OM) to convert between units: food product properties are automatically converted into the right unit so they can be used by models.

Work Package 2 (WP2) focuses on finding better sensing technologies for food production processes. While the topic of this work package is not directly related to this project, it does provide a potential use case for the MaaS prototype: WP2 has created a sensing method that determines droplet sizes in an emulsion using near-infrared (NIR) light. This method uses a model that can be integrated with the (improved version of the) MaaS prototype.

1.2 Outline

The rest of this document dives deeper into the different aspects of the CaaS project, taking multiple perspectives to provide a complete picture. Chapters 2 and 3 discuss the context and preparation of the project: what is the problem at hand? What is required for the project to succeed? Chapters 4, 5, and 6 cover the architecture, design, and implementation of the platform created to solve these problems. Chapter 7 covers the project management aspects of the project. Chapter 8 concludes the report with a reflection on what was achieved during the project and what further steps could be taken in future work.
2. Problem Analysis

The first thing to do when starting a project is to figure out the actual problems that are to be solved. In this chapter, I approach the problem from both the inside and the outside. First, I present the domain of the problem. Then, the problem itself is dissected. The resulting in-depth problem description is then used to define the goals of this project. I also identify assumptions that were made and document constraints.

2.1 Domain Analysis

A project never stands on its own: it lives in its context, which needs to be considered when aiming for success. Part of this context is the domain of the project. This domain dictates the concepts and vocabulary used throughout the project. A project can live in multiple domains, coming from different aspects of the project. Analyzing the domain of a project ensures that the development process and project results are in line with the world of the client. Having a mental model of the domain of a project also makes communication with the client more effective.

This project lies in the middle of two domains: the domain of Food Production, and the domain of Data and Model Sharing. More precisely, the project aims to introduce Data and Model Sharing to the world of Food Production. Since both domains are important, the domain analysis covers the concepts of each of them in the next sections.

2.1.1. Food Production Domain

The Food Production Domain has concepts related to the development and production of food products. Food products are defined as a combination of ingredients, packaging information, and properties. The process that is used to create the food product is also a part of this definition, which is expressed in several process steps. The combination of the ingredients and process is called the recipe of the food product.

Production processes are often observed by sensors. These sensors produce sensor data, which can be used to monitor and optimize the observed process. Organizations can model their food products and related processes into food-related models, which can be used to optimize production processes. Figure 1 shows how the concepts in the Food Production Domain are connected. Some concepts from the Food Production Domain are already related to concepts from the Data and Model Sharing Domain.
2.1.2. Data and Model Sharing Domain

The Data and Model Sharing Domain has concepts related to the usage of and relations between data and models. Models have an interface that determines what input data the model accepts and what output data the model provides. Both models and data are owned by an organization. An organization can share data and models with other organizations. Quantities are specific kinds of data, which are represented by a combination of a unit and a value.

Both data and models can be described using ontologies. These descriptions can be used to make sure that a model receives data as described in its interface. Organizations can define simulations to run models with data. To specify how the data should be put into the models, simulations consist of bindings, which bind data to model interfaces. A simulation produces a result, which is the combination of the outputs of all the models defined in the simulation. Figure 2 provides an overview of all the concepts in the Data and Model Sharing Domain.
2.2 Problem Definition

The Internet of Food project aims to speed up innovation in the food industry by developing new ways for companies to collaborate. A step in that direction was the Model as a Service (MaaS) project [1]. The premise of this project was to make a system that allows organizations to share data and models, without giving them away. This resulted in the MaaS prototype.

An organization providing models can connect their models to the MaaS prototype, where they can be shared with other organizations. Organizations can also share food product information using the prototype. It then provides a list of models available to a user, based on the organization they are a part of. To use the models, users can define simulations using a food product and one or more models. The MaaS prototype can then run all models, providing information about the food product as input to the models. The results of these model runs can be viewed by the user when the models are completed.

The MaaS prototype allows users to see the potential of data and model sharing. However, it still has some limitations:

- It only accepts a limited amount of information about a food product (ingredients, production process steps, properties, and packaging information) in one specific format.
- It stores all food product information on a central server. This is not desirable for organizations, as this information is intellectual property.
- It also supports only one model, and adding models that have an interface different from this model is not possible without making changes in the design and implementation of the prototype.
- It offers limited functionality for combining models in a simulation. All models defined in a simulation are merely executed one after another, without any inter-model interactions.
2.2.1. Tomato Soup Taste Use Case

The original use case was about modeling the taste of tomato soup. This use case involves one model, which takes the ingredient mass-fractions in the recipe of tomato soup, and calculates a “taste score” for four different taste properties (sweetness, sourness, saltiness, and “tomato-taste”) the resulting tomato soup would have. See Figure 3 for an overview of this use case. This and subsequent figures show what data (in yellow) and models (in blue) are involved in the use case, and what company they belong to (dotted line). They show the flow of information throughout the use case (purple arrows), which always ends in a result (purple ellipse). Finally, the figures show what additional company knowledge is required for the use case (red clouds).

Figure 3: Situation schema of the Tomato Soup Taste use case. Company A has information about food ingredients and wants to know the taste of the resulting food product. Company B has a model that takes ingredients of tomato soup and returns the resulting taste. The model only supports ingredients known by Company B.

The calculation uses an ingredient reference table, which provides extra information on the taste-score contribution of different ingredients. This means that the model only accepts ingredients that exist in this ingredient reference table. To ensure this, the MaaS prototype only allows users to express their food products in terms of ingredients found in this reference table. This limits the platform to only be used with ingredients in a fixed list from the company that creates this tomato-soup-taste model. It also greatly reduces the future utility of the model for any company, as they are likely to want to use the same model but with a different list of ingredients. Meanwhile, more use cases have been recognized. These use cases are based on new example models, which show the need for more advanced functionalities than those supported by the MaaS prototype.

2.2.2. Nutritional Value Use Case

The first new use case is the Nutritional Value use case. This use case is almost the same as the tomato soup taste use case: it also is about a model that accepts ingredient mass fractions and uses a reference table to calculate food properties. The only difference is that this model uses a different reference table and calculates the predicted nutritional value of the food product. While this use case does not require changes in the MaaS platform to be demonstrable, it does show the need for a mechanism to support these kinds of reference tables from different sources; after all, every organization might have its reference tables that they use in their modeling tools.
2.2.3. Pasteurization-Shelflife Use Case

Another new use case is the Pasteurization-Shelflife use case. This use case uses two models (one on pasteurization, the other on shelf life), which have a similar purpose: both models take information on the production process and lifecycle of a food product and calculate the predicted number of microbes in the product in different phases of the product lifecycle. Each model takes the starting number of microbes and produces the predicted number after one lifecycle step. This makes for the interesting use case of chaining models together: using the output of one model (amount of microbes after pasteurization) as the input of another (amount of microbes before the product is put on the shelves). This would allow a user to predict the final number of microbes in their food product, directly observing the effect of changing parameters in the pasteurization process making use of both the pasteurization and shelflife models.

![Figure 4: Situation schema of the Pasteurization-Shelflife use case](image)

2.2.4. Calibration-Droplet size Use Case

Another model-based use case comes from work done in Work Package 2 (WP2) of the Internet of Food project. As part of WP2, a method is developed for determining the droplet size of an emulsion (mayonnaise in this case) by measuring its attenuation of Near-InfraRed (NIR) light as a function of droplet size. We call this the Calibration-Droplet size use case since the approach requires calibration based on reference data to determine the attenuation at different wavelengths for different known drop sizes before it can be used. This results in two models: a calibration model and a droplet size model. The calibration model uses reference sensor data, combined with known droplet sizes, to generate calibration data. The droplet size model requires this calibration data to determine the droplet size given new attenuation data for different wavelengths. This use case again shows the need for model chaining, as it allows for the combination of calibration and droplet size determination to be performed in one simulation. It also shows that there are more sources of data to consider than just the production
information of a food product: sensor data (in this case coming from NIR light intensity sensors) is an observation of the product, not a parameter that is set at the start of the production process.

2.2.5. Optimization Experiment Use Case

Lastly, there is also the use case of Optimization Experiments. In an optimization experiment, the user tries to determine the effect on the food product when one parameter is changed. To do this, the user wants to try out multiple values for certain model inputs, to see how the different values affect the result of a simulation. This procedure is very tedious in the MaaS prototype since a user has to jump back and forth between changing the food product definition and running the simulation. Additionally, while an optimization experiment is temporary, the parameter changes have to be made directly on the food product definition, which can be critical information that a company does not want to change quickly.
2.3 Project Goals

The implementation and design of the MaaS prototype have some limitations that hold back its main function of showing the potential of model sharing in the food industry. These limitations of the MaaS prototype are:

- It only supports data about the ingredients, production process steps, properties, and packaging information of food products, storing all food product information on a central server.
- It only supports the Tomato Soup Taste use case, requiring design and implementation changes to support other use cases.
- It offers limited simulation functionality, only running all the models in a simulation in sequence.

The overarching goal of the Collaboration as a Service (CaaS) project is to extend the MaaS prototype to remove these limitations. The new CaaS platform should allow users to share data and models related to all the use cases defined in Section 2.2, according to the FAIR principles (Findable, Accessible, Interoperable, and Reusable). The result will be a new platform: the CaaS platform. I define the following goals for the CaaS project:

1) Update the MaaS prototype so that it increases the interoperability of data and models shared through it.
   To support all the use cases mentioned in the problem description, the CaaS platform should allow users to connect these different kinds of data with these models. However, not all models can accept every type of data. Whether data can be used as the input of a model depends on many properties of the data:
   
   a. The storage method of the data: is it stored in a file? Or a database?
   b. The way the data can be accessed: can the data be downloaded? Is the data available through an API? Can the data be fetched completely? Can it be queried?
   c. The format in which the data is stored: is the data a table? Or a graph?
   d. The availability of the data: is the data public? If not, what kind of credentials or access rights are required to access it?
   e. The content of the data: is the data relevant? Is it complete? Is it presented using the expected unit?

   The same can be said about the data expectations and accessibility of models. The CaaS project should allow any data to be used as input for a model, as long as the resulting output of the model is expected to be relevant to the actors in the use cases from Section 2.2. In other words, the CaaS platform should increase the interoperability of all data and models that are shared using the platform.

2) Provide a uniform method for describing data and models.
   To determine whether data can be used as the input of a model, users must be able to understand what exactly is being described in a dataset, or what exactly is expected and returned by a model. To accommodate this, the CaaS platform should allow users to describe their models and data in a uniform way. This description should be enough for the platform to determine whether a dataset is compatible with a model.

2.4 Assumptions and Constraints

The purpose of this project within the Internet of Food project is to create a system that can be demonstrated to show the power and feasibility of data and model sharing. Hence, the development should be focused on implementing and demonstrating the sharing functionality; other issues (payment, security, intellectual property protection) are out of the scope of this project. Moreover, while the
definition of models can be very broad, I assume only models that are part of industry-provided use cases have to be supported by the CaaS platform.

The CaaS project is a PDEng graduation project. Every PDEng graduation project has a fixed time of 9 to 10 months. This is also a limit for the CaaS project. After 10 months, the project should be completed and handed over to the client.

The CaaS platform is a continuation of the MaaS prototype created last year. While it would be an option to completely rebuild the system using my own choice of technologies, given the time constraint on this project I have concluded I should build upon the codebase, tools, and technologies that were used to create the MaaS prototype.
3. Requirements Elicitation

To create a system that conforms to all stakeholder wishes, it is essential to convert the general problem statements from Chapter 2 into use cases and requirements. Use case scenarios provide a complete overview of the interactions between the system and users that are expected by stakeholders. They also are a starting point for the definition of requirements. The requirements are a precise representation of the wishes of the stakeholders. Their precise definition allows them to be used to prioritize the different aspects of the project. To accomplish this, each requirement has a priority assigned to it using the MoSCoW system. The resulting list of prioritized requirements defines the scope of the project.

The use case scenarios and requirements for this project are found using the elicitation process laid out in Section 3.1. The final use case scenarios and requirements resulting from this process are discussed in Sections 3.2 and 3.3.

3.1 Elicitation process

To ensure I execute the project based on complete lists of use cases and requirements, I followed the following requirement elicitation process:

1. **Find all stakeholders that have some concerns with the project.** Since all stakeholder concerns should be reflected in the use cases and requirements, it is first of all important to have a complete list of stakeholders. This list can be found in *Appendix A*.

2. **Gather previously created use cases and requirements.** The MaaS prototype was created as part of a PDEng graduation project. This project had its own lists of use cases and requirements. These form a starting point for the use cases and requirements of this project.

3. **Meet with stakeholders to gather their concerns, using existing use cases and requirements as a starting point.** The goal is to find out how the current concerns differ from the existing requirements.

4. **Update the use cases and requirements based on stakeholder input.** This includes updating the use case scenarios and updating the existing priorities of the requirements.

5. **Discuss the updated use cases and requirements with stakeholders.** At the end of the discussion, either the use cases and requirements are accepted by the stakeholders, or the stakeholders have provided feedback on how to change the use cases and requirements even further (going back to the previous step).

3.2 Use case scenarios

Use cases describe the way that a system can be used. Each use case has a name and a scenario attached to it. Use cases are also connected to one or more user roles. Each user role corresponds to a role in which a user wants to use the system. The main user roles in the CaaS platform are:

- **Simulation User**: a user who wants to use the CaaS platform for running a simulation
- **Model Owner**: a user who owns one or more models and wants to share them through the CaaS platform.
- **Data Owner**: a user who owns data and wants to share them through the CaaS platform.

An actual user can fill more than one role at the same time: one user can be both a Data Owner and a Simulation User at the same time. Figure 6 shows an overview of the different use cases, how they are connected, and how user roles are connected to use cases.
Figure 6: Use Case diagram. Yellow use cases have been modified compared to the previous MaaS project, while orange use cases have been added for this project.

Each use case comes with a scenario. A scenario is a step-by-step description of the interaction between the user(s) and the system. Each step is either an action performed by a user or an action performed by the system. All scenarios are listed in Appendix B.

3.3 Requirements

From the use cases defined in the previous section, I derived the system requirements. There are multiple types of requirements for this project:

- Functional System Requirements
- Non-Functional System Requirements
- Functional SEP Requirements

The functional system requirements describe the functionalities that the system should support. Their priorities reflect how important the stakeholders deem these requirements to be. The non-functional system requirements describe desired properties of the system under development. Each of these properties should hold for all functions described in the functional system properties. The functional and non-functional requirements defined for the MaaS project [1] were used as a starting point of the elicitation process. I updated these lists such that they fit the new wishes of the client.
One of the main goals of the project is to make data and models more interoperable. While the starting list of requirements does cover the models, it does not cover data sources. The MaaS requirements are also very limited on the possibilities of connecting data with model inputs. The requirements in Table 1 were added to better reflect these needs. These requirements are about sharing data sources in a similar way as models, as well as providing more ways for data and models to be connected. The complete list of functional and non-functional requirements can be found in Table C.9 and Table C.10 respectively.

Table 1: Functional requirements that are added to the MaaS functional requirements to fit the CaaS project goals

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR14</td>
<td>The system shall provide a gateway with a URI that will allow interaction with a data source through a general REST API.</td>
</tr>
<tr>
<td>FR15</td>
<td>The system shall allow the user to connect a data source to the MaaS infrastructure by specifying the corresponding gateway URI and metadata about the data provided by the data source.</td>
</tr>
<tr>
<td>FR16</td>
<td>The system shall show a list of data sources that were connected by the current user.</td>
</tr>
<tr>
<td>FR17</td>
<td>The system shall show a list of data sources that are available to or have been shared with the current user.</td>
</tr>
<tr>
<td>FR18</td>
<td>The system shall allow the user to view data provided by data sources.</td>
</tr>
<tr>
<td>FR19</td>
<td>The system shall check whether the inputs provided to a model in a simulation are compatible with the defined inputs of the model using metadata describing the model input and the provided input data.</td>
</tr>
<tr>
<td>FR20</td>
<td>The system shall convert the unit of provided input to match the unit required by the model, using the Ontology of Measure.</td>
</tr>
<tr>
<td>FR21</td>
<td>The system shall allow the user to share or un-share a connected data source with other users who are from different organizations.</td>
</tr>
<tr>
<td>FR22</td>
<td>The system shall allow the user to export the results of a simulation.</td>
</tr>
<tr>
<td>FR23</td>
<td>The system shall allow the user to export data provided by data sources.</td>
</tr>
<tr>
<td>FR24</td>
<td>The system shall allow the user to directly enter input for a simulation.</td>
</tr>
<tr>
<td>FR25</td>
<td>The system shall allow the user to select data from a data source as an input for a simulation.</td>
</tr>
<tr>
<td>FR26</td>
<td>The system shall allow the user to select the output of a model as an input for another model.</td>
</tr>
<tr>
<td>FR27</td>
<td>The system shall allow the user to describe a data source using metadata.</td>
</tr>
<tr>
<td>FR28</td>
<td>The system shall allow the user to describe a model interface using metadata.</td>
</tr>
</tbody>
</table>

In addition to the regular functional system requirements, supplementary functional requirements were made for the Software Engineering Project (SEP) that was executed by bachelor students in parallel to this project. These functional requirements form the basis for requirements set up by the SEP team. More information on the SEP can be found in Chapter 7. The complete list of functional requirements related to the SEP can be found in Table C.11.
4. System Architecture and Design

A system architecture is a top-down description of a system. It provides an overview of the different elements that exist in the system, how they are connected, and how they work together to achieve the functionality as required by the use cases and system requirements. During the development of such a system architecture, there are always multiple challenges that need to be tackled. Each of these challenges can generally be solved in multiple ways, each approach having positive and negative consequences. To get to a complete architecture, multiple design decisions have to be made.

In this chapter, I present the system architecture of the CaaS platform, as well as the design challenges and design decisions that led to this architecture. I cover both the structure and the behavior of the system using SysML diagrams and textual descriptions. In these descriptions, I also show how the architecture changed compared to the MaaS prototype. These changes make the CaaS platform more flexible, allowing it to be used for more use cases.

4.1 Design Decisions

4.1.1. Data Storage and Exchange

One of the main goals of the CaaS project is to make the CaaS platform support more types of data. Of course, I could extend the database of the MaaS prototype to contain the data that is required for the new use cases. This poses a problem since it would require the platform to be updated every time a new use case is found. Since another main goal of the CaaS project is to allow any potentially relevant data to be used with models, alternative solutions should be considered. The following are the two alternatives that I considered for the CaaS platform:

- **Connect the platform to the storage solutions of the organizations owning the data.** The data that is used on the CaaS platform is owned by many different organizations. These organizations usually have these data stored on some kind of internal database software. To use these data in the MaaS prototype, these organizations would have to transform them to the supported format and enter the data on the web interface. This takes a lot of effort when handling large or continuously changing datasets.

  Instead of moving the data from the organization database software, I could also make the platform connect to the organization database to fetch the data. This removes the unnecessary moving of data and requires no extra effort from the data-providing organization. However, it significantly increases the number of interfaces and data formats that should be supported by the CaaS platform.

- **Create a gateway between the storage solutions and the platform, similar to the model gateways of the MaaS prototype.** In the MaaS prototype, gateways are used to bridge between the diverse interfaces of models and a general model execution interface. The same approach can be used to connect the diverse interfaces and data formats of database software with the CaaS platform. This takes away the downsides of the previous solution while keeping most of its benefits. However, the approach does require more effort to create the gateways, either by the organizations owning the databases or by the developers of the CaaS platform. Another downside of this approach is that having to convert from and to a general data format can make the platform less performant. However, network access is already intrinsically less performant and for the applications that I want to demonstrate; the performance hit is not very important.

The options can be compared based on their compatibility, scalability, implementation effort, and performance. Compatibility is about the amount of different data storage methods and data formats the platform can support. Use case scalability indicates whether the platform can handle an increasing
number of data sources and data sets. Implementation effort is about the amount of effort that would be required to implement it, either during the project or after. Performance is about the efficiency of passing around data from inside and outside the platform.

Table 2 shows how the different approaches compare given relevant design criteria. A minus sign indicates that the design criterion is negatively impacted by the approach, while a plus sign indicates a positive impact. Note that the “direct connection” approach hurts compatibility, as its compatibility is limited by the implementation of the platform, whereas the compatibility of the “data gateways” approach can always be extended with more gateways. Based on this comparison, I selected the “data gateways” approach for data storage.

<table>
<thead>
<tr>
<th></th>
<th>Direct connection</th>
<th>Data gateways</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compatibility</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Use Case Scalability</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Implementation Effort</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Performance</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

4.1.2. Data format

The use of data gateways requires an additional decision: what data format will be used for the general interface of these gateways? There are many formats in which data can be represented. Some examples are lists, matrices, tables, graphs, and objects. Most of the time it is also possible to convert data from one format to another. For example, a directed graph can be converted into a table with the columns “source”, “relation”, and “target”. The resulting table might be less practical in use since it is harder to query for relation-chains that would be trivial to find in the original graph representation. After multiple discussions with stakeholders, I settled on the use of data tables and data schemas. Figure 7 shows an example of a data table and a data schema that describes it.

A data table is a two-dimensional data structure that presents data in rows and columns. Each row represents an entity, while each column represents a property an entity can have. A single cell in a data table represents one property of one entity. Data tables are a very simple and ubiquitous data format, used by Comma-Separated Values (CSV) files, Pandas and R’s data frames, and relational databases.

A data schema is a description of all the columns that a data table contains. Each column has a name, but it can also have more properties. A column could also specify the unit in which all its rows are defined, or that the values in its rows refer to a column in another data schema. A data schema can be seen as a class of data tables: multiple data tables can be described by the same data schema if they have the same columns with the same metadata attached to them. Data schemas can even be used to describe an expectation of a type of data table. Hence, data schemas can also be used to describe the input and output interfaces of a model, as further described in Section 4.3.1.
4.1.3. Metadata format

A challenge for designing the CaaS platform is that it should provide a uniform method for users to describe the data and models they want to share. The platform requires this metadata to determine whether a dataset and a model input are compatible or how a dataset could be transformed such that it becomes compatible. Such a metadata definition method can have many forms, but for this project, I make a distinction between two different ones:

- **Custom metadata format based on domain analysis.** The domain analysis described in Section 2.1 can be used to determine what information is necessary for comparing datasets and model interfaces. All this information can then be stored in a custom metadata format. This way, the metadata format contains exactly the information needed by the CaaS platform. However, this approach is not future-proof, since the information needed by the CaaS platform might change in the future. It also requires all organizations that want to use the CaaS platform to understand the format.

- **Ontology definition.** Ontologies are a formal method of describing concepts and relations about a certain subject area. Such a description can also be used for data and model interfaces. A big advantage of using ontologies is that they provide a lot of flexibility: it is always possible to extend an ontology by adding new concepts and relations to it. They are harder to integrate into software applications since they are not natively supported by popular database systems. However, the MaaS prototype already uses the Ontology of units of Measure [2] for unit conversion, so the same systems can be reused.

Out of these two options, the ontology approach is used for the CaaS platform. In particular, an ontology vocabulary was developed for describing data schemas. Using ontologies for all metadata solves the flexibility problem of the MaaS prototype. If for any reason changes need to be made to the concepts or relations used to describe datasets and model interfaces, these changes can easily be added to the ontology definitions.
4.1.4. Metadata Exchange

The last design challenge of the CaaS platform is the method of storing metadata. The platform should use a metadata-storage approach that is maintainable, accessible, and efficient. The maintainability of a storage method is about how simple it is for a user to update their metadata. Accessibility is about the ease of access to metadata by other users. Efficiency is about the number of transfers required when the metadata is used in simulations.

Metadata could be stored either in one central location, or it could be stored in the gateways. Table 3 shows how these methods compare, showing positive and negative impacts on each criterium with plus signs and minus signs respectively.

Table 3: Comparison of metadata-sharing storage methods

<table>
<thead>
<tr>
<th></th>
<th>Centralized</th>
<th>Decentralized</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintainability</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Accessibility</td>
<td>-</td>
<td>+</td>
</tr>
<tr>
<td>Efficient</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

Storing the metadata in a central location makes it more efficient to access it when running simulations on the platform since no interaction with external servers is required. Maintenance and accessibility are hurt by this approach since organizations have to access the central server to update or view the metadata. The inverse is true when metadata is stored in the gateways, making it the more compelling alternative to make the sharing of metadata follow the FAIR principles.

4.2 CaaS Architecture

The architecture for the CaaS platform uses a “microservice architecture” approach, similar to the MaaS prototype. Every model is shared as a microservice, exposed through a model gateway. All model gateways have the same interface towards the outside, which allows users to run a model with specified input and fetch the results of a model run once it is completed. This uniform interface allows a wide variety of models to be connected to the sharing platform. On the inside, a model gateway runs its underlying model with the data it is supplied and stores the results of these model runs.

The sharing platform is a central hub of the microservice architecture. Next to being the point of access for users, it is in charge of the normalization and supply of data to the gateways for model execution. Figure 8 shows a component diagram of this approach for one model.
The CaaS platform reuses the microservice approach for the sharing of data. Instead of storing data on the sharing platform, data is stored in separate data source servers. Examples of such data source servers are relational databases, object stores, or any other server that stores data that could be used as input for a model. A data gateway is then added between the central server and the data source servers, similar to the model gateways from the MaaS prototype. As a result, the central CaaS sharing platform only stores information about the gateways, keeping both models and data located at their source. Unlike the MaaS prototype, the CaaS platform also moves ontology storage away from the sharing platform. The model- and data gateway owners become responsible for storing and providing the metadata of the models and data they serve. This is more sensible since they are the only ones who can make informed decisions in describing their models and data. Figure 9 shows an overview of the resulting architecture. The next sections provide more detail on the different blocks in this architecture.
4.2.1. Gateways

Data Gateways are an integral part of the CaaS platform. A data gateway exposes a Data Gateway API that can be used to access the data source it represents. The Data Gateway API is the same for all data gateways, even though the storage method and interface of the data source can vary. The data gateway is responsible for converting the data as provided by the data source into data tables. It is also responsible for providing metadata about the data that is stored in the data source. How this metadata is created is up to the implementation of the data gateway, which is further discussed in Chapter 5.

For the data gateway to be used within the CaaS platform, it should be registered on the Sharing Platform. The data gateway performs this registration by itself by contacting the Data Registration API of the sharing platform. This registration process uses an API key to register the data gateway for a specific owner. All users within the same organization can access the data directly, while the owner of the gateway can specify other organizations to share the data source with. All the steps of setting up and registering a data gateway are shown in Figure 10.
Figure 10: Activity Diagram of the Data Gateway sharing process

Model gateways were already an important part of the MaaS prototype. They are slightly updated in the CaaS platform to function similarly to data gateways. Just like data gateways, model gateways register themselves with the sharing platform through the Model Registration API. They also expose a Model Gateway API that provides metadata describing the inputs and the outputs of the model. The model gateway API also exposes endpoints for running the model and viewing run results, similar to the endpoints that were available in the MaaS prototype.

4.2.2. Sharing Platform

The Sharing Platform is the central component of the CaaS platform. It is a hub for users to share and use data sources and models. The sharing platform is split into multiple parts. The Frontend provides a web interface that can be used to view information about registered models and data sources, run simulations, and view the result of simulation runs. The Model Register and Data Register allow model gateways and data gateways to register themselves on the sharing platform. The registers also keep track of registrations, saving them on the Data Storage.

Data Storage is a relational database that is used for the storage of information on the sharing platform. It stores information about user accounts, organizations, gateway registrations, simulation specifications, and simulation run history. It does not store data used in simulations anymore. The Simulator can run simulations based on simulation definitions. It takes care of gathering the required data from data sources, converting these data if necessary, and running models using these data as input. Section 4.3 covers more about how users can run simulations on the CaaS platform. Figure 11 shows the different parts of the Sharing Platform, and how they are connected.
4.3 Simulation Workflow

A simulation is a combination of multiple models and multiple datasets. When a simulation is run, all the models are run with the datasets as their input. To perform a simulation, the CaaS platform needs to know what datasets should go to what models. It should also have a procedure to run all the models with the desired data inputs and in the desired order. Sections 4.3.1 and 4.3.2 show how the CaaS platform allows users to define simulations and how it executes simulations based on their definition.

4.3.1. Simulation Definition

To run a simulation, the CaaS platform must first know what exactly needs to be run. A user must be able to define what models have to be run and what data have to be fed into these models. The steps to define a simulation in the CaaS platform are shown in Figure 12.

The first step of defining a simulation is to specify what models and data sources should be part of the simulation. Any registered data source or model can be part of a simulation. But the CaaS platform also needs to know what data should be put into which model. The next step is therefore to bind model inputs to data sources. Bindings tell the CaaS platform to use a specific dataset as an input for a specific model. As mentioned in Section 4.1.2, the data in data sources are represented as data tables, while the inputs and outputs of a model interface are represented by data schemas. Since these both have columns,
bindings can be made on a column basis: every model input column can be bound to a data source column. Considering that model outputs can also be used as data, it is even possible to bind a model output column to an input column of another model. Figure 13 shows some column-based binding examples. The red arrows in the figure indicate bindings.

Figure 13: Example of column-based simulation bindings

The example binds two columns from a data source to the two inputs of Model A. Note that both the data source column Ingredient_name and the model input column Ingredient refer to the same entities from an external data table. Such references allow the CaaS platform to check for compatibility between data source columns and model input columns. Similarly, the units of the two Amount columns can be compared. If the units do not match, a unit conversion can be applied. The example also shows how an output column of Model A can be bound to an input column of Model B.

In some cases, it can be convenient to be able to enter data directly in the web interface: some data might just not be available in a database. The CaaS platform also allows users to define such direct inputs. When a model input is bound to a direct input, a user can enter the input data directly in the web interface.

Lastly, when the simulation is saved, the CaaS platform performs a check on whether all model inputs are bound. This is important since a model cannot be run when it is missing input data. When all inputs of all models are bound to some data source or direct input, the simulation definition can be used to run simulations.
4.3.2. Simulation Execution

When a simulation is defined, a user can choose to run it. When running a simulation, the CaaS platform is responsible for making sure that each model is run with the correct input data. In general, the following steps need to be taken to achieve this:

1) Determine what models need to be run
2) Pick a model to run
3) Gather all the data the model requires to run
4) Convert the data if needed and put it in a format the model interface accepts
5) Run the model with the converted and formatted input data
6) Repeat from step 2) until all models are completed

The MaaS prototype in theory follows a similar procedure. It loops through the list of models (even though only one model was implemented) in a simulation definition. For each model, it transforms the data from the food product selected in the simulation definition so that it can be sent to the model. The MaaS prototype then runs the model with this input data and moves on to the next model, until all models are completed.

The changes made in the CaaS platform require a more involved procedure for the simulation process to succeed. Figure 14 gives an overview of the simulation process used by the CaaS platform. First of all, the CaaS platform does not store the data itself anymore: it has to fetch the data from data gateways. Since a simulation definition also specifies the data sources that are used in the simulation, the platform can fetch all the data from this list and temporarily store them before running any models. When data is required as model input, it is transformed to fit the model interface. While this approach is not necessarily efficient concerning storage space, it makes sure that all data is available when it is required for running a model.

Secondly, the CaaS platform has to handle model dependencies. Since a model output can be bound to a model input, the CaaS platform has to temporarily store the results of model runs so they can be used as the input of another model. Even so, a model can be scheduled to run before the models it depends on are completed. The solution to this is to check whether all bound input data is available before running a model. If not, then the model is skipped. Then the list of models can be looped over multiple times until it is empty. If the list of incomplete models does not shrink in one iteration, this means there is a problem with the simulation definition and the simulation run cannot be completed.
Figure 14: Sequence Diagram of the simulation execution process
5. Implementation

The main goal of this project is to extend the functionality of the MaaS prototype. The updated architecture and design described in Chapter 4 are a roadmap for the changes that have to be made to create the new CaaS platform.

This chapter presents the implementation of the CaaS platform. Since the MaaS prototype is the starting point of this implementation, Section 5.1 gives a brief overview of the implementation of the MaaS prototype. Sections 5.2, 5.3, and 5.4 dive deeper into the implementation details of the CaaS platform. Finally, Section 5.5 covers the deployment approach of the CaaS platform.

5.1 MaaS Prototype

The MaaS prototype implementation is split into two parts: the frontend and the backend. The frontend is an Angular application that serves the web interface, while the backend contains all the other services. This split is reflected in the folder structure of the project: the frontend and the backend each have their Git repository. Sections 5.1.1 and 5.1.2 contain more information about these two parts.

Many of the microservices that comprise the MaaS prototype are deployed using Docker [3] containers. Each container is isolated from all the others, containing exactly the files and libraries required to run the microservice it represents. All the containers are connected through a virtual network, so they can communicate with each other freely. A Docker Compose [4] configuration is used to conveniently set up all microservice containers at the same time. An overview of the deployment can be found in Figure 15.
5.1.1. MaaS Frontend

The frontend was developed as part of a SEP, as part of the Model-as-a-Service project. It is developed using Angular [5], a TypeScript [6] based web development framework. The Angular app is deployed on a single Docker container that hosts all the files of the app using an Nginx [7] server.
The frontend is the main point of user interaction for the MaaS prototype. Users can use the frontend to view and manage their food products, models, and simulations. Models can be registered manually in the **My Models** view. Users can also view food products and models from other organizations than their own. The frontend also allows users to define and run simulations using the food products and models available to them. Figure 17 shows what views are provided by the frontend, and how they are connected. An example of the “My Food Products” view is shown in Figure 16.
5.1.2. MaaS Backend

The MaaS backend is a combination of multiple services. It contains the Model Sharing Backend, the Model Gateways, and an Ingredient Service. It also contains the Unit Translation Component: a library that uses the Ontology of units of Measure (OM) for unit conversion. All services are implemented using Python Flask, using the Marshmallow [8] library for serialization of Python objects. They store data on a PostgreSQL [9] database using the SQLAlchemy [10] library. Metadata is stored in a GraphDB [11] database using the SPARQLWrapper [12] library. Both the PostgreSQL and the GraphDB databases are run as Docker containers.

The Model Sharing Backend is the central service of the MaaS prototype. It handles user authentication, storage of food product information, storage of model metadata, and the execution of simulations. It also uses the Unit Translation Component for unit conversion during simulation runs. The Model Sharing Backend gets a list of available ingredients from the Ingredient Service. This list can be used by users in defining the ingredients of food products.

The Model Gateway service is a configurable implementation of a tomato taste model and a model gateway. The model can be configured to output a taste score for any combination of four tastes: sweetness, sourness, saltiness, and tomato taste. The model uses ingredient information from the Ingredient service to calculate taste scores. By default, the Docker Compose configuration describes two instances of this service: one model gateway outputs sweetness and sourness, while the other outputs saltiness and tomato taste. The results of completed model runs are stored on the PostgreSQL database container.
5.2 CaaS Gateway Implementation

The model gateway from the MaaS prototype implementation is the starting point for the new data and model gateways of the CaaS platform. Just like the MaaS model gateway, both the data gateway and the CaaS model gateway are implemented as single configurable services. The configuration determines what data is provided by a data gateway instance and what model is served by a model gateway instance.

All gateways use the same formats when providing and accepting data and metadata. Data is transferred using a JSON format that represents data tables. A data table is presented as a one-dimensional array of JSON objects, as shown in Figure 18. A model interface might require multiple data tables as an input. These data tables can be combined into one JSON structure.

Figure 18: Example of JSON format representing a data table

The format used for transferring metadata between gateways and the CaaS sharing platform is TURTLE. TURTLE is a file format that stores triples in a human-readable format. The triples in a TURTLE file define a labeled graph, which represents an ontology of concept classes and directed relations between the classes. Figure 19 shows an example of an ontology and the TURTLE file that represents the ontology. A specific ontology vocabulary is used for defining data schemas with extra information. This vocabulary is further explained in Section 5.3.

Figure 19: Example TURTLE file representing an ontology

On the MaaS platform, users had to manually register model gateways by adding them to the system. This requires users to enter all information about the model gateway, including information about the model inputs and outputs. On the CaaS platform, gateway registration is performed through an API endpoint on the CaaS sharing platform service. Gateways can register themselves directly through this
endpoint, using an API key to authenticate as a specific user. Users can find this API key on the User Profile view and add it to the configuration of their gateways.

### 5.2.1. CaaS Model Gateways

CaaS model gateways function in a similar way to the model gateways of the MaaS prototype: they allow users to run a model and fetch the results of completed runs. The CaaS model gateway is still one configurable implementation. However, it can be configured to run any of the models that are part of the use cases described in Section 2.2. Each of these models is implemented as a subclass of an abstract class called `Model`. The `Model` class describes the methods that are common for each model. It also describes an abstract method for running the model, which is implemented differently for each of the model subclasses. The model gateway then instantiates the correct model class at runtime based on its configuration.

Each model has different input and output requirements. These requirements are documented in the model subclasses in the form of Marshmallow schema classes: each model subclass provides an input schema and an output schema. These schemas are used by the model gateway to verify that input and output data conform to the input and output requirements of the currently selected model. The model gateway also uses the input and output schemas of a model subclass to generate an ontology definition of the model interface. The gateway offers this ontology definition through a newly added API endpoint as a TURTLE file. Figure 20 shows how the models for the Tomato Soup Taste use case and Nutritional Value use case are implemented using this approach.

![Class Diagram of the implementation of the model classes.](image)

Figure 20: Class Diagram of the implementation of the model classes.
5.2.2. CaaS Data Gateways

The MaaS prototype stored food product data directly on the model sharing backend. The CaaS platform adds data gateways to decentralize data storage, as mentioned in Section 4.1.1. The CaaS data gateway service implementation is based on serving the contents of one or more files in the Comma-Separated Values (CSV) format. Such files are widely used as a simple representation of tabular data. In the configuration of the data gateway service, users can specify one or more CSV files. The service then registers and exposes a data source on the CaaS sharing platform for each of these CSV files. Each CSV file gets a unique URL based on its file name. The configuration also contains a data schema in JSON format, which describes the data tables in each of the CSV files. The data gateway service uses this information to generate ontology definitions of the metadata and expose this metadata through TURTLE files. Figure 21 shows a data schema configuration in JSON and a data table is represented in a CSV file.

![CSV file and JSON data schema configuration for a recipe]

5.3 Ontology Vocabulary

The CaaS platform uses ontologies to describe the metadata of data sources and model interfaces. Each gateway supplies the metadata of all the resources it represents. Each of these pieces of metadata must be expressed using the same concepts and connections. Hence, I collaborated with Dr. Rogier Brussee and Dr. Hajo Rijgersberg in developing an OWL vocabulary for describing data schemas. Appendix E provides an overview of the ontology vocabulary. This vocabulary supports the description of data schemas with additional column annotations. These column annotations come in two flavors: unit annotations and reference annotations.

Unit annotations specify the unit in which the values in a column are expressed. This information is used during simulations to perform unit checks and unit conversion: if there is a mismatch between the unit of a data source column and the unit expected by a model input column, the data in the data source column should be converted to the correct column. Since the CaaS platform uses the Ontology of units of Measure [2] (OM) for unit conversion, all units in OM are supported.
The vocabulary supports multiple ways for specifying the unit (or units) of the values in a column. First of all, a fixed unit can be specified for all values in the column. This can be used to specify a unit requirement for a model input column, though it can also be used for data sources when all the values in a column are expressed in the same unit. If the latter is not the case, then it is also possible to specify that the units of a column can be found in another column. Such a definition allows each row in a column to be expressed in a unique unit. For model input columns this approach is less useful since it would put the responsibility of unit conversion on the model instead of the CaaS platform.

Reference annotations indicate that the values in a column refer to values from another schema or concept. They are similar to foreign key relationships in a relational database system: when a column refers to a column in another table, the original column is expected to only contain values that exist in the other table. The other table can also serve as a source of extra data that is linked to the original table through the reference column. This mechanism allows models to express limits on input columns.

5.4 User Interaction

With the changes to data and model sharing that come with the CaaS platform, the simulation system also needs an update. Simulations need to gather data from different data gateways and pass it to model gateways using the updated APIs. Users also need to be able to define simulation bindings and enter data for direct inputs. Both the frontend and the sharing platform are updated to enable these new simulation requirements. Appendix F contains screenshots of all the pages available in the frontend.

Multiple changes are made to the views in the frontend. The Food Product pages are replaced by Data Source pages. These pages allow the user to view all the data sources that are registered to accounts in their organization, as well as data sources that are shared with their organization. Users can view the data schema and data table for each of the data sources available to them. The Model pages are also updated to fit the functionality of the new model gateways: users can only view the input and output data schemas of the model, instead of also being able to update them.

The add/edit simulation page of the CaaS platform allows users to enter all the information needed to define a simulation. Instead of selecting a single food product, users can select one or more data sources to use in the simulation. After selecting one or more models, users can bind data source columns to each column of the inputs of all the models. Model input columns can also be bound to a direct input (indicated as “input”); when a column is bound to a direct input, the user can finally enter data directly in the last tab of the add/edit simulation page.

5.5 Deployment

The CaaS platform is set up for deployment using multiple Docker images and a Docker Compose configuration file, similar to the deployment strategy of the MaaS prototype. Each Docker image contains all the dependencies required to run the service it represents. To build and run all the service instances defined in the Docker Compose, Docker Desktop [13] needs to be installed. Complete instructions on how to deploy and run the CaaS platform in a README file included with the implementation.
6. Verification & Validation

When working on a software project, it is important to make sure that the project results come out as expected. Verification and validation are the two overarching project activities that check that this is the case. Verification is about making sure that the software works according to specification. Validation is about checking whether the final project result meets client expectations. In this chapter, I show my approach to verification and validation in the CaaS project.

6.1 Verification

The goal of the verification process is to check whether the software implementation works according to the architecture and design specifications. All verification tests should pass if all features are implemented correctly. Due to the limited time available for the project, I focused on integration testing, as I deem this type of testing to be sufficient to show that all designed features are implemented correctly.

6.1.1. Integration Tests

The integration tests are about testing whether the functionalities work as intended when multiple modules are connected. They are closely related to the use case scenarios described in Section 3.2. Each integration test case is therefore connected to one or more use case scenarios. Table 4 presents the results of all the integration test cases. A complete description of all of the test cases can be found in Appendix D.

<table>
<thead>
<tr>
<th>Test ID</th>
<th>Test name</th>
<th>Related use case(s)</th>
<th>Test result</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIT-01</td>
<td>Register Data Source Gateway</td>
<td>Publish data, Authenticate as part of company(^1)</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-02</td>
<td>Register Model Gateway</td>
<td>Publish model, Authenticate as part of company(^1)</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-03</td>
<td>Edit Data Source Gateway information</td>
<td>Publish data, Define access rights</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-04</td>
<td>Edit Model Gateway information</td>
<td>Publish model, Define access rights</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-05</td>
<td>View Data Source data table</td>
<td>Publish data, Access shared data</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-06</td>
<td>View Data Source data schema</td>
<td>Publish data, Access shared data</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-07</td>
<td>View Model interface data schema</td>
<td>Publish model</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-08</td>
<td>Define simulation with direct input</td>
<td>Define simulation, Provide simulation input</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-09</td>
<td>Define simulation with input from a data source</td>
<td>Define simulation, Provide simulation input</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-10</td>
<td>Define simulation with chained models</td>
<td>Define simulation, Provide simulation input</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-11</td>
<td>Run simulation with direct input</td>
<td>Run simulation, Check simulation run status</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-12</td>
<td>Run simulation with input from a data source</td>
<td>Run simulation, Check simulation run status</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-13</td>
<td>Run simulation with chained models</td>
<td>Run simulation, Check simulation run status</td>
<td>Passed</td>
</tr>
<tr>
<td>CIT-14</td>
<td>View result of simulation with a single model</td>
<td>View simulation results</td>
<td>Passed</td>
</tr>
</tbody>
</table>

\(^{1}\) Specifically for authenticating the gateway using an API key
6.2 Validation

The validation process determines whether the CaaS platform is implemented in such a way that the client is satisfied. I validated the CaaS platform by comparing the implementation to the project goals and the system requirements. I also constantly validated intermediate project results during the bi-weekly review sessions. This section provides the results of the validation process.

6.2.1. Reflection on Goals

The main goals, as explained in Section 2.3, of the CaaS project are:

3) Extend the MaaS prototype to support the newly discovered use cases.
4) Update the MaaS prototype so that it increases the interoperability of data and models shared through it.
5) Provide a uniform method for describing data and models.

The first goal is achieved through the implementation and integration of the Nutritional Value use case, Pasteurization-Shelflife use case, and Calibration-Droplet size use case. The models and data sets related to the use cases were implemented and attached to a gateway.

The second goal is fulfilled by the addition of data gateways and the use of data schemas and data tables. When an organization implements a data or model gateway according to the specifications of the CaaS platform, they can connect the underlying dataset or models directly with other datasets and models that are shared on the CaaS platform.

Finally, the third goal is fulfilled by the introduction of the ontology vocabulary for data schemas. This vocabulary gives organizations a language to describe their data and models such that the CaaS platform, as well as other organizations, can understand and use them.

6.2.2. Reflection on Requirements

As established in Chapter 3, the complete set of requirements form a formal and more precise definition of the wishes of the client. Reflecting on the requirements is, therefore, a good method for determining whether all client wishes are fulfilled. I went over all of the functional requirements for this project and determined whether they are satisfied by the results of this project. The result can be found in Appendix C.

Based on this analysis, almost all functional requirements are satisfied by the project results. Requirement FR29 (The system shall allow the user to define a food product by specifying the recipe, processing, and packaging information) is no longer directly satisfied by the CaaS platform, but users can still define a food product with these properties using data sources and direct model inputs. The original method of defining food product information directly on the platform also no longer fits the rest of the requirements, since this data is supposed to be stored in the gateways.

6.2.3. Demonstrations

The best way to know whether the client is satisfied with the results of the project is to present the results to them. We presented the ontology vocabulary during a WP1 meeting. This presentation gained a positive response from the members of WP1. A final demonstration at a WP1 meeting is planned to show how the new use cases are supported by the CaaS platform.
7. Project Management

Every PDEng graduation project has a starting date and an end date. Many different activities are required to bring the project to success. In this chapter, I present how I managed these activities in the CaaS project. In Section 7.1 I summarize all activities that are important to this project using a Work Breakdown Structure (WBS) diagram. In Section 7.2 I put turn this list of activities into a project planning and project schedule. Lastly, I identify project risks and determine risk mitigations in Section 7.4.

7.1 Work Breakdown Structure

With many activities being required to make the project successful, it helps to structure them into a Work Breakdown Structure (WBS). Activities are broken down into smaller activities until the smallest activities are small enough for further planning. The larger activities can be used to create milestones, while the smaller activities can be used to create tasks.

The WBS in Figure 22 provides a quick overview of all the activities that are part of the CaaS project. The yellow activities are the top-level activities, which are broken down into the orange activities below them. The first two activities – Analyzing MaaS and Architecting CaaS – are about getting to know the results of the MaaS project and determining what should be done to improve it in the CaaS project. The implementation of the CaaS platform is split into multiple sub-activities, each of which can be used as a milestone. Lastly, the Executing SEP assignment activity contains all the sub-activities required to set up, run, and use the results of the SEP that was executed alongside the CaaS project.

![Figure 22: Work Breakdown Structure of the CaaS project](image)

7.2 Project Planning and Scheduling

A PDEng graduation project takes between 9 and 10 months, from January until October. I planned out all the activities from the WBS in Section 7.1 within this time frame. The resulting planning is
summarized in Figure 23. This planning is split into three phases, each covering a period of 3 to 4 months: the Current, the New, and the Future.

7.2.1. Project Phases
In the Current phase, I focused on analyzing the results of the MaaS project and working towards a solution direction for the rest of the CaaS project. To get acquainted with the MaaS prototype, I implemented and integrated the Nutritional Value use case. The results from these activities are described in Chapters 2, 3, and 4.

The New phase was all about implementing the architecture and designs created in the Current phase. It involves the implementation of the CaaS gateways and the updates to the CaaS simulation workflow. In this period I also was the client of the Flavouride project, the Software Engineering Graduation Project (SEP) of ten Bachelor students. This project spanned 11 weeks, starting on April 19th and ending with a final presentation on July 2nd. Their original assignment was to implement optimization experiments into the MaaS prototype, as described in Section 2.2.5. However, their project scope was increased after the first week to also implement the Pasteurization model as described in Section 2.2.3, as well as to improve the usability of the frontend user interface by improving searching and sorting functionality and adding export functionality.

Lastly, the Future phase was about utilizing the groundwork implemented in the New phase with new use cases. These new use cases are the Pasteurization-Shelflife and Calibration-Droplet size use cases described in Section 2.2.3 and 2.2.4. This time is also reserved for documenting and preparing for the final presentation and defense. While the previous two phases lasted three months, the Future phase also includes October as a buffer month.

7.2.2. Project Pacing
The phases described in Section 7.2.1 provide a very rough schedule for the entire project. At a more granular level, I used two-week sprints for consistent pacing throughout the project. The boundaries between sprints function as moments of reflection and planning: at the start of each sprint, I planned what I could achieve in that sprint, and at the end of each sprint I reflected on my actual progress. Each sprint ended with a meeting with my supervisors with additional input on my progress.
Milestones bridge between the sprint pacing and the phases of the project. Each milestone corresponds to the completion of multiple activities in the WBS. More information on each of these milestones is can be found in Table 5. Milestone deadlines were planned in-between sprints.

Table 5: Milestones of the CaaS project

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Activities to be completed</th>
<th>Deadline</th>
<th>Completion date</th>
</tr>
</thead>
<tbody>
<tr>
<td>M0</td>
<td>Analyzing MaaS, Architecting CaaS</td>
<td>April 24, 2021</td>
<td>April 24, 2021</td>
</tr>
<tr>
<td>M1</td>
<td>Implementing Data Gateway, Integrating Data Gateways</td>
<td>June 4, 2021</td>
<td>June 4, 2021</td>
</tr>
<tr>
<td>M2</td>
<td>Implementing CaaS gateways, Implementing CaaS simulation updates</td>
<td>July 16, 2021</td>
<td>August 25, 2021</td>
</tr>
<tr>
<td>M3</td>
<td>Implementing new use cases</td>
<td>August 27, 2021</td>
<td>October 22, 2021 (projected)</td>
</tr>
</tbody>
</table>

### 7.3 Communication Strategy

Consistent feedback is very important in a software project. To get feedback during the CaaS project, I set up a weekly timeslot for meetings with my supervisors. At the end of each sprint, this timeslot was used for P3G meetings and review meetings. On the other weeks, the timeslot was used for brainstorm sessions, the topic of which varied depending on the activities planned for the current sprint. Due to the COVID-19 pandemic, all meetings related to the CaaS project, including my final presentation, have been performed through email and Microsoft Teams.

Throughout the project, I worked in close collaboration with other members of Work Package 1 (WP1), especially on the development of the ontology vocabulary for data schemas. These collaborations required many meetings, which were planned at the moment a meeting was required. I also participated in WP1 team meetings and “full-team” meetings of the Sustainable Food Institute (SFI). During these meetings, I presented and discussed my intermediate results.

### 7.4 Risk Management

Risks in the CaaS project are reduced by determining potential risks throughout the project and coming up with a mitigation strategy. Whenever a risk came up, I recorded it and determined the impact the risk could have on the project, as well as the probability that the risk could occur. I also came up with a mitigation strategy. I then discussed the risks and mitigation strategies with my supervisors. The resulting list of risks can be found in Table 6.

Table 6: Risks and Risk Mitigation strategies

<table>
<thead>
<tr>
<th>ID</th>
<th>Risk</th>
<th>Impact</th>
<th>Probability</th>
<th>Mitigation Strategy</th>
</tr>
</thead>
</table>
| R1 | Communication issues with the client due to the COVID-19 pandemic | Medium | Medium      | - Schedule fixed time slot for meetings with supervisors over the complete duration of the project  
- Reschedule fixed time slot if a supervisor cannot attend  
- Schedule one-on-one meetings with supervisors who cannot attend one weekly meeting  
- Keep supervisors who cannot meet in a week up to date using email |
| R2 | SEP students deliver insufficient results | Low | Medium | - Set clear problem description  
- Thoroughly prepare the assignment to remove uncertainties  
- Steer SEP students towards results in weekly meetings |
| R3 | Change in scope or priorities in the middle of the project | High | High | - Illicit feedback from client and stakeholders to clarify and agree on scope and priorities  
- Make priorities explicit  
- Keep track of a list of prioritized requirements  
- Share the list of prioritized requirements with stakeholders |
| R4 | Ontology from WUR is not usable in time | Medium | Low | - Agree on the structure of the ontology early  
- Agree on the scope of the ontology within the system  
- Request recurring updates on ontology  
- Work on small example ontology as a drop-in replacement of WUR ontology |
| R5 | Implementation is not ready for demonstration | High | Medium | - Create new features such that they can be delivered in steps  
- Work on features based on priority to maximize the utility of the (potentially incomplete) architecture  
- Focus on demo potential, showing the use of different data sources and models in the system |
8. Conclusion

In this chapter, I reflect on the results achieved in the CaaS project. In Section 8.2 I make recommendations and suggestions on potential future work based on the results of this project. In Section 8.3 I reflect on my experiences of executing this project.

8.1 Results

At the start of this project, the MaaS prototype was a good starting point for a model-sharing platform. Where the MaaS project set out to show how an organization can share a model with other organizations without giving it away, this project expanded on this idea, looking into what is necessary to extend the possibilities of model and data sharing towards increasing sustainability of the food industry. Collaboration between organizations is key, hence the name of this project: Collaboration as a Service, or CaaS for short.

The CaaS platform removes the asymmetry that existed in the MaaS prototype between the sharing of models and the sharing of data: while organizations could share models without giving them away using the MaaS prototype, data had to be copied directly onto the sharing platform. The CaaS platform introduces data gateways to keep data ownership with the organization that provides the data. This approach also opens up new possibilities for data sharing in the future, since novel data storage methods can be supported just by implementing a new type of data gateway. Furthermore, The organization owning the data is in charge of what data they want to expose through their gateway.

The CaaS platform applies the data-focused perspective on model sharing as well. This resulted in the creation of the data schema and data table formats used for sharing metadata and data respectively. The resulting generalized approach for describing data and model interfaces enables many different ways to feed data into models, such as data mixing and model chaining. This makes both data and models that are integrated with the CaaS platform more interoperable.

The ontology vocabulary for data schemas is an important result of this data-focused approach. It provides a generally applicable language for organizations to describe their data and models. Since other organizations can use the same language, these other organizations can also directly use these data and models. The vocabulary also future-proofs the platform, since it can be extended as data and model interfaces used in the food industry become more complex.

Another important part of the results of the CaaS project is the platform implementation. This implementation includes example use cases that come directly from industry partners. The fact that these use cases are supported brings the CaaS platform close to the food industry. It also fulfills the goal of extending the data and model support of the MaaS prototype.

8.1.1. Deliverables

The CaaS project has multiple deliverables:

- **CaaS platform implementation**: all the code that is part of the CaaS platform implementation. It is structured in the same way as the MaaS prototype, composed of a frontend git repository, a backend git repository, and a Docker Compose configuration file.
- **CaaS ontology vocabulary**: the TURTLE files describing the ontology vocabulary that can be used to describe data schemas. It is hosted in a public Github repository at [https://github.com/Nielesonimusso/internet-of-food-ontologies](https://github.com/Nielesonimusso/internet-of-food-ontologies). This repository also contains the ontology definitions for all the use cases described in Section 2.2; these serve as examples of how the ontology vocabulary can be used.
- **Flavouride results (SEP):** the Flavouride Software Engineering Project (SEP) has produced many results, not all of which have been integrated with the CaaS platform implementation. Since these are still valuable results, the complete results of the Flavouride project are also included with the deliverables.

- **Final report:** this final report is also a deliverable, explaining in detail the results in terms of architecture, design, and implementation, as well as the CaaS project that led to these results.

### 8.2 Future Work

Throughout the project, I came up with many ideas that could improve the CaaS platform. The timeframe of the project limited the scope, so not all of these ideas are included in the execution of the CaaS project. Some of these ideas were also not in the scope of the project anyway due to the assumptions and constraints mentioned in Section 2.4. Even though they did not fit into the scope of this project, these ideas are still potential future steps that could be taken within the Internet of Food project. In this section, I list all the ideas that did not make it into the CaaS platform.

#### 8.2.1. Broadened Functionality

The use cases that were provided as part of the Internet of Food project indicate what features are required from the CaaS platform. The data-focused sharing approach could potentially support many more use cases, like using mixing columns from multiple data sources for one model input. The current CaaS platform implementation focused on at least supporting the provided use cases; hence some features – like the one mentioned above – are not yet implemented, even though the data schema format could support them. It would be interesting to gather more use cases and see what features the CaaS platform requires to support them.

Another feature that could require more research is the unit conversion feature. In the Tomato Soup Taste and Nutritional Value use cases, ingredients in a recipe are proportioned with a quantity with the unit “percentage”. The models also take a “dosage” input, which is considered to be the “whole” to the percentages of each ingredient. While conversion of percentages to another unit is possible when the “whole” is known, the connection between the “whole” and the “parts” is not supported by the CaaS prototype. Additionally, the Ontology of units of Measure (OM) does not directly support the units that are used in the Pasteurization-Shelflife use case. More work is required to completely implement the unit conversion feature, as well as figure out what other kinds of unit conversions are relevant to food-industry-related models.

#### 8.2.2. Performance Improvements

The CaaS platform was not designed with performance in mind. This decision sometimes hurts the model running performance. For example, all models are executed in sequence to check before running each model whether all required data is available. Most of the time this is not needed since there are no dependencies between models, so the platform could benefit from running multiple models in parallel. This approach would require a more advanced scheduling mechanism for when model dependencies do exist.

In the current CaaS platform implementation, all data is fetched by the platform before it is sent to models for a model run. This is inefficient since data is essentially copied twice. A possible alternative is to have model gateways directly contact data gateways to gather the data they need for a model run. The platform should then instruct the model gateway where it should fetch the data, and how it should convert this data to its model inputs. While this method is potentially much more efficient, it complicates access control. This could be solved using an authorization framework like OAuth 2.0 [14].

Metadata sharing could also be made more efficient than the current approach. The CaaS gateways send a TURTLE file describing all the metadata about the data or model they represent. Most of the time only a fraction of this information is needed to answer a simple query. An alternative would be to have...
the gateways implement a SPARQL query endpoint [15]. However, this would increase the effort required to implement new model gateways.

8.2.3. Usage Incentives

The topic of “usage incentives” for organizations is purposefully left out of the scope of the CaaS project since the CaaS platform is mostly to be used as a demonstration platform. Payment models and the protection of Intellectual Property (IP) are still interesting topics to consider to increase the interest of external parties.

Payment for the use of data or models of an organization can be difficult to get right. In general, it would be possible to just track how many times each organization has used your model or data source, and send periodic invoices to all companies based on their usages of your resources. For data sources, this already becomes a bit less clear: when do you consider a data source to be used? When its data is accessed? What if a data source is used multiple times within one simulation?

IP protection is also an interesting topic. Data that is used in a simulation is still passed around to every model that is part of the simulation. This means that data could accidentally be shared with organizations it should not be shared with given the access rights defined on the CaaS platform. Even if this would not be the case, the owner of the data has no control over the actions that are performed on the data; the creator of a model could easily add a side-channel to their model that just stores all data inputs. Data sanitation and model certification could be solutions to this problem.

Another solution to leaking IP could be to have models be executed by a third party. This party receives all data and models that are part of a simulation and is only allowed to communicate back the results of the simulation. This way, no organization involved in the simulation can see anything other than the simulation results. However, this third party also has to be trusted, and it needs to have the resources in place to run any of the models shared on the CaaS platform.

Even models suffer from an intellectual property leaking issue: even though they are hidden behind a gateway, it is still possible for an external organization to reverse-engineer a model given enough model runs or the right set of input data. To do this, the model-owning organization has to share the model with the external organization. The model runs used in the reverse-engineering process can also be traced back to the external organization, such that the organization can be held liable for its actions. Additionally, the external organization has to pay for every model run, which, depending on the number of model runs required for reverse-engineering, can become costly.

8.2.4. Decentralization

The microservice architecture used for the CaaS platform is very decentralized: the complete “system” is composed of many different services existing at many different locations (organizations). The Sharing Platform module is the only module that is centralized. This is not ideal, since this module still requires a single owning organization or entity, which becomes a single point of failure for the entire ecosystem.

A completely decentralized CaaS platform would be an ideal solution. One way to achieve this is to use blockchain technology to turn the Sharing Platform module into a decentralized app (dApp). Such an app could store information about gateways on the blockchain while authorizing model runs and data access using transactions. These transactions could even be used to carry the payments. This approach would most likely require direct connections between data and model gateways as mentioned in Section 8.2.2 since data transfer and storage on blockchains are generally expensive.

8.2.5. User Experience

While the user experience of the CaaS platform was important in a way, it was not the main priority throughout the project. Especially the visual aspect of the user interface could be improved in some cases. More importantly, the user experience for creating bindings and entering direct inputs could be made more intuitive than it is now. The current interface is made to give the user the ability to define
simulations and enter direct inputs at all. It still allows for some incorrect inputs, and it is not always clear where the user should enter what information. An ideal user experience is hard to achieve, which is why I think more research can benefit the experience significantly.

The results from the Flavouride project also include multiple user experience improvements. Mainly, they added support for optimization experiments to the simulation workflow. This means that users can vary a value in a defined range, after which the simulation will perform model runs for each of the values in that range. This reduces the amount of back-and-forth the user has to do to perform an optimization experiment. While this could be implemented into the CaaS prototype, it introduces a new challenge for the results viewer: with the introduction of multiple outputs per model, the results viewer already has to deal with showing multiple tables per model. If the optimization experiments were added, this would also add another layer of outputs to display.

The Flavouride project also added improved search and sort functionality for all of the tables that exist in the MaaS prototype, making navigation through the available food products and models easier as their numbers increase. Such a feature can keep the CaaS platform manageable as the number of supported use cases increases.

### 8.3 Project Retrospective

In this section, I reflect on my experiences of executing the CaaS project. This project is my first project where I filled many different roles at the same time: I was the architect, the engineer, but also the product owner and the project manager. I had to figure out the goals of the project, turn those into required activities, and then plan these activities such that they were all completed in nine to ten months. Filling all these roles gave me plenty of opportunities to apply the skills I had learned in the first year of the PDEng project.

Normally, a PDEng graduation project would be executed on-site at the company. In my case, that would have meant I would work at either the TU/e campus, the WUR campus, or at the Unilever office. However, due to the COVID-19 pandemic, I have executed the entire project from home. On the one hand, this worked out well for me: I did not have to travel, so I could be flexible on my working time while still filling my required weekly hours. This flexibility requires extra effort to keep structure in the project process though. Having regular contact with my supervisors helped me a lot, especially with keeping up with my bi-weekly sprint pace.

As part of the CaaS project, I also became a client for the Flavouride SEP project. This was the first time for me as a client for any project, which gave me an interesting perspective on the difficulties of this role. I prepared well for this role by defining in detail the requirements for this project. It turns out it is very difficult to make requirements that are both clear and not too strict: on the one hand, they should be clear enough such that the project team would know what I wanted, but on the other hand, they should not be so strict as to limit the creativity of the team. And even though I defined the requirements at the start, they still changed throughout the first week of the Flavouride project. Overall, this experience made me understand the role a client has and can have in a project.
## Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>PDEng</td>
<td>Professional Doctorate in Engineering</td>
<td></td>
</tr>
<tr>
<td>TKI</td>
<td>Topconsortium voor Kennis en Innovatie</td>
<td><a href="https://topsectoragrifood.nl/">https://topsectoragrifood.nl/</a></td>
</tr>
<tr>
<td>SFI</td>
<td>Sustainable Food Initiative</td>
<td><a href="https://www.sfifood.nl/">https://www.sfifood.nl/</a></td>
</tr>
<tr>
<td>INoF</td>
<td>Internet of Food</td>
<td><a href="https://www.sfifood.nl/projects/the-internet-of-food">https://www.sfifood.nl/projects/the-internet-of-food</a></td>
</tr>
<tr>
<td>MaaS</td>
<td>Model as a Service</td>
<td></td>
</tr>
<tr>
<td>CaaS</td>
<td>Collaboration as a Service</td>
<td></td>
</tr>
<tr>
<td>SEP</td>
<td>Software Engineering Project</td>
<td><a href="https://studiegids.tue.nl/opleidingen/bachelor-college/majors/computer-science-and-engineering/curriculum/final-project/">https://studiegids.tue.nl/opleidingen/bachelor-college/majors/computer-science-and-engineering/curriculum/final-project/</a></td>
</tr>
<tr>
<td>TU/e</td>
<td>Eindhoven Technical University</td>
<td><a href="https://www.tue.nl/en/">https://www.tue.nl/en/</a></td>
</tr>
<tr>
<td>WUR</td>
<td>Wageningen University &amp; Research</td>
<td><a href="https://www.wur.nl/">https://www.wur.nl/</a></td>
</tr>
<tr>
<td>ISPT</td>
<td>Institute for Sustainable Process Technology</td>
<td><a href="https://ispt.eu">https://ispt.eu</a></td>
</tr>
<tr>
<td>WP1</td>
<td>Work Package 1</td>
<td></td>
</tr>
<tr>
<td>WP2</td>
<td>Work Package 2</td>
<td></td>
</tr>
<tr>
<td>NIR</td>
<td>Near-InfraRed</td>
<td></td>
</tr>
<tr>
<td>FAIR</td>
<td>Findable, Accessible, Interoperable, Reusable</td>
<td><a href="https://www.go-fair.org/fair-principles/">https://www.go-fair.org/fair-principles/</a></td>
</tr>
<tr>
<td>Simulation</td>
<td>Combination of connected data and models</td>
<td></td>
</tr>
<tr>
<td>Synonym: model run job</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MoSCoW</td>
<td>Must have, Should have, Could have, Won’t have</td>
<td></td>
</tr>
<tr>
<td>CSV</td>
<td>Comma-Separated Values</td>
<td></td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
<td></td>
</tr>
<tr>
<td>SQL</td>
<td>Structured Query Language</td>
<td></td>
</tr>
<tr>
<td>SPARQL</td>
<td>SPARQL Protocol and RDF Query Language</td>
<td><a href="https://www.w3.org/TR/rdf-sparql-query/">https://www.w3.org/TR/rdf-sparql-query/</a></td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
<td><a href="https://www.json.org/">https://www.json.org/</a></td>
</tr>
<tr>
<td>TURTLE</td>
<td>Terse RDF Triple Language</td>
<td><a href="https://www.w3.org/TR/turtle/">https://www.w3.org/TR/turtle/</a></td>
</tr>
<tr>
<td>OWL</td>
<td>W3C Web Ontology Language</td>
<td><a href="https://www.w3.org/OWL/">https://www.w3.org/OWL/</a></td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
<td></td>
</tr>
<tr>
<td>SysML</td>
<td>Systems Modeling Language</td>
<td><a href="https://sysml.org/">https://sysml.org/</a></td>
</tr>
<tr>
<td>WBS</td>
<td>Work Breakdown Structure</td>
<td></td>
</tr>
<tr>
<td>dApp</td>
<td>Decentralized Application</td>
<td></td>
</tr>
</tbody>
</table>
Bibliography

References


## Appendix A: Stakeholder Analysis

Table A.7: Stakeholders of the project, grouped by company

<table>
<thead>
<tr>
<th>Eindhoven Technical University</th>
<th>Role</th>
<th>Interest</th>
</tr>
</thead>
</table>
| Niels Rood                    | PDEng Software Technology Trainee   | - Deliver successful results for this PDEng final project  
|                               |                                     | - Deliver project results that satisfy the customers’ needs  
|                               |                                     | - Gain experience in managing and executing a complex design project                                 |
| Yanja Dajsuren                | Software Technology PDEng Program Director | - Make sure the project is executed and completed under PDEng Software Technology program standards  
|                               |                                     | - Provide feedback on technical and non-technical aspects of the project                            |
| Jakob de Vlieg                | Academic Supervisor                 | - Demonstrate the potential of a data and model sharing infrastructure  
|                               |                                     | - Attract more organizations to participate in the Internet of Food project                        |
| Hossain Muhammed Muctadir     | Ph.D. Candidate, Previously PDEng Software Technology Trainee | - Provide information on the execution and results of the Internet of Food implementation created in the previous year |
| SEP Team                      | Developers                          | - Deliver successful results for their final Software Engineering Project  
|                               |                                     | - Align their results with the needs of their customer (Niels Rood)                                |
| Rogier Brussee                | Additional Supervisor               | - Provide feedback and knowledge regarding the requirements and possible solutions for this project |

<table>
<thead>
<tr>
<th>Wageningen University &amp; Research</th>
<th>Role</th>
<th>Interest</th>
</tr>
</thead>
</table>
| Görkem Simsek-Senel              | Ontology Expert and Developer      | - Develop ontologies for the available data and models  
|                                  |                                     | - Discuss on integration and usage of ontologies within the Internet of Food                      |
| Hajo Rijgersberg                 | Ontology Expert and Developer      | - Develop ontologies for the available data and models  
|                                  |                                     | - Discuss on integration and usage of ontologies within the Internet of Food                      |
| Puneet Mishra                    | WP2 Developer                      | - Develop calibration and sensor models for integration of WP2 results                              |

<table>
<thead>
<tr>
<th>Unilever</th>
<th>Role</th>
<th>Interest</th>
</tr>
</thead>
</table>
| Michiel Gribnau                  | Company Supervisor                | - Make sure that the project results are in line with the needs of the industry  
|                                  |                                     | - Demonstrate the potential of a data and model sharing infrastructure  
|                                  |                                     | - Attract more organizations to participate in the Internet of Food project                        |
### Stakeholders

<table>
<thead>
<tr>
<th>Stakeholder</th>
<th>Role</th>
<th>Interest</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kevin van Koerten</td>
<td>Model Developer</td>
<td>- Create model implementations for Internet of Food use cases</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Provide support on the use of said model implementations</td>
</tr>
</tbody>
</table>

Figure A.24: Structural overview of stakeholders of this project. The diagram shows both the organization and work package each stakeholder is a part of.
## Appendix B: Use cases

**Table B.8: Use case Scenarios**

<table>
<thead>
<tr>
<th>Use Case</th>
<th>Scenario</th>
<th>Related user roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access shared data</td>
<td>1. (User) Request list of accessible data sources</td>
<td>User</td>
</tr>
<tr>
<td></td>
<td>2. (System) Show a list of data sources that the user is allowed to access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. (User) Select data source to access</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. (System) Show information about the data source</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. (User) Access data of data source through provided URL</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Alternate:</strong> 1a. Create user account</td>
<td></td>
</tr>
<tr>
<td>Authenticate as part of a</td>
<td>1. (User) Log in using a user account</td>
<td>User</td>
</tr>
<tr>
<td>company</td>
<td>Alternate: 1a. Create user account</td>
<td></td>
</tr>
<tr>
<td>Define a simulation</td>
<td>1. <code>&lt;extend&gt;</code> Define access rights</td>
<td>Simulation User, Model Owner,</td>
</tr>
<tr>
<td></td>
<td>2. (Simulation User) Navigate to the page for adding simulation</td>
<td>Data Owner</td>
</tr>
<tr>
<td></td>
<td>3. (System) Show available data sources and models</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. (Simulation User) Select models to run and data sources to use</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. <code>&lt;include&gt;</code> Provide simulation input</td>
<td></td>
</tr>
<tr>
<td>Provide simulation input</td>
<td>1. (Simulation User) Connect data sources to model inputs</td>
<td>Simulation User</td>
</tr>
<tr>
<td></td>
<td>2. <code>&lt;extend&gt;</code> Define model chain</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. <code>&lt;extend&gt;</code> Define optimization experiment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. (System) Verify all inputs are provided with data</td>
<td></td>
</tr>
<tr>
<td>Define model chain</td>
<td>1. (Simulation User) Connect model outputs to model inputs</td>
<td>Simulation User</td>
</tr>
<tr>
<td></td>
<td>2. (System) Automatically determine a (partial) mapping between connected model outputs and inputs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. (Simulation User) Complete mapping between connected model outputs and inputs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. (System) Verify all connected model outputs and model inputs are correctly mapped</td>
<td></td>
</tr>
<tr>
<td>Define optimization</td>
<td>1. (System) Show input values which can be specified as to-be-optimized</td>
<td>Simulation User</td>
</tr>
<tr>
<td>experiment</td>
<td>2. (Simulation User) Specify input values to optimize</td>
<td></td>
</tr>
<tr>
<td>View simulation results</td>
<td>1. (Simulation User) Navigate to the simulation page</td>
<td>Simulation User</td>
</tr>
<tr>
<td></td>
<td>2. (System) Show performed runs for the simulation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. (Simulation User) Select run to view</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. (System) Show results from all iterations of the selected run</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. <code>&lt;extend&gt;</code> Export simulation results</td>
<td></td>
</tr>
<tr>
<td>Export simulation results</td>
<td>1. (Simulation User) Request export of simulation results</td>
<td>Simulation User</td>
</tr>
<tr>
<td></td>
<td>2. (System) Prepare and provide results of all iterations of the simulation</td>
<td></td>
</tr>
<tr>
<td>Run simulation</td>
<td>1. (Simulation User) Navigate to the simulation page</td>
<td>Simulation User</td>
</tr>
<tr>
<td></td>
<td>2. (System) Show available simulations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. (Simulation User) Select simulation to run</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. (System) Perform run iterations required for simulation</td>
<td></td>
</tr>
<tr>
<td>5. &lt;extend&gt; Check simulation run status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>--</td>
<td></td>
</tr>
</tbody>
</table>
| **Check simulation status** | 1. **(Simulation User)** Navigate to simulation run page  
2. **(System)** Fetch and show current simulation run status | Simulation User |
| **Publish model** | 1. **(Model Owner)** Implement model provider to use Model Sharing API  
2. **(Model Owner)** Model provider registers using Model Sharing API  
3. **(System)** Model Register stores registration for listing  
4. <include> Define access rights | Model Owner |
| **Publish data** | 1. **(Data Owner)** Implement data provider to use Data Sharing API  
2. **(Data Owner)** Data provider registers using Data Sharing API  
3. **(System)** Data Register stores registration for listing  
4. <include> Define access rights | Data Owner |
| **Define access rights** | 1. **(User)** Open model/data definition in Web Interface  
2. **(System)** Show a list of companies that the model/data can be shared with  
3. **(User)** Select from a list which companies the model/data needs to be shared with | Model Owner  
Data Owner |
# Appendix C: Requirements

Table C.9: Functional system requirements for this project

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR01</td>
<td>The system shall provide a gateway with a URI that will allow interaction with the computational model through a general REST API.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR02</td>
<td>The system shall allow the user to connect a model to the MaaS infrastructure by specifying the corresponding gateway URI and the input as well as output parameters of the model.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR03</td>
<td>The system shall allow the user to share or un-share a connected model with other users who are from different organizations.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR04</td>
<td>The system shall show a list of models that were connected by the current user.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR05</td>
<td>The system shall show a list of models that are available to or have been shared with the current user.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR06</td>
<td>The system shall allow the user to define a simulation by specifying one or more models and data sources.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR07</td>
<td>The system shall show the results of successfully executed models and error messages if the model execution fails.</td>
<td>Should</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR08</td>
<td>The system shall allow the user to execute a simulation.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR09</td>
<td>The system shall allow the user to view the data resulting from a simulation.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR10</td>
<td>The system shall annotate the information regarding the input and output parameters of a model using ontology.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR11</td>
<td>The system shall allow the user to log in to their account using username and password.</td>
<td>Should</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR12</td>
<td>The system shall allow the user to create an account.</td>
<td>Should</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR13</td>
<td>The system shall allow the user to link their account to an organization.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR14</td>
<td>The system shall provide a gateway with a URI that will allow interaction with a data source through a general REST API.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR15</td>
<td>The system shall allow the user to connect a data source to the MaaS infrastructure by specifying the corresponding gateway URI and metadata about the data provided by the data source</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR16</td>
<td>The system shall show a list of data sources that were connected by the current user.</td>
<td>Should</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR17</td>
<td>The system shall show a list of data sources that are available to or have been shared with the current user.</td>
<td>Must</td>
<td>Satisfied</td>
</tr>
<tr>
<td>FR18</td>
<td>The system shall allow the user to view data provided by data sources.</td>
<td>Should</td>
<td>Satisfied</td>
</tr>
</tbody>
</table>
FR19: The system shall check whether the inputs provided to a model in a simulation are compatible with the defined inputs of the model using metadata describing the model input and the provided input data.

FR20: The system shall convert the unit of provided input to match the unit required by model, using the Ontology of Measure

FR21: The system shall allow the user to share or un-share a connected data source with other users who are from different organizations

FR22: The system shall allow the user to export the results of a simulation

FR23: The system shall allow the user to export data provided by data sources

FR24: The system shall allow the user to directly enter input for a simulation

FR25: The system shall allow the user to select data from a data source as an input for a simulation

FR26: The system shall allow the user to select the output of a model as an input for another model

FR27: The system shall allow the user to describe a data source using metadata.

FR28: The system shall allow the user to describe a model interface using metadata.

FR29: The system shall allow the user to define a food product by specifying the recipe, processing, and packaging information.

Table C.10: Non-functional system requirements for this project

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>NF01</td>
<td>Bug fixes, performance improvements, or feature implementations in the backend or in the gateway will not change their provided interface</td>
<td>Should</td>
</tr>
<tr>
<td>NF02</td>
<td>The backend shall be decoupled from the frontend</td>
<td>Must</td>
</tr>
<tr>
<td>NF03</td>
<td>The implementation of the backend and the gateways shall be decoupled from each other</td>
<td>Must</td>
</tr>
<tr>
<td>NF04</td>
<td>The system shall allow user interaction at all times</td>
<td>Should</td>
</tr>
<tr>
<td>NF05</td>
<td>The system shall be able to handle a model execution request when the provided data is invalid</td>
<td>Should</td>
</tr>
</tbody>
</table>

Table C.11: Functional requirements for SEP assignment

<table>
<thead>
<tr>
<th>ID</th>
<th>Requirement</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>FRO31</td>
<td>The system shall allow the user to define an optimization experiment by specifying a simulation with data inputs to optimize</td>
<td>Must</td>
</tr>
<tr>
<td>FRO32</td>
<td>The system shall allow any numeric value in data input to be specified as an input to be optimized</td>
<td>Should</td>
</tr>
<tr>
<td>FRO33</td>
<td>The system shall allow the user to specify the range of values to try out for a to-be-optimized data input in an optimization experiment</td>
<td>Must</td>
</tr>
<tr>
<td>FRO34</td>
<td>The system shall perform an iteration for every possible combination of values for all to-be-optimized data inputs in an optimization experiment</td>
<td>Should</td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>FRO35</td>
<td>The system shall allow the user to enter the specific values to be used in an optimization experiment for a to-be-optimized data input</td>
<td>Could</td>
</tr>
<tr>
<td>FRO36</td>
<td>The system shall allow the user to enter the lower bound, upper bound, and step size for defining the range of values to try out for a to-be-optimized data input in an optimization experiment</td>
<td>Could</td>
</tr>
<tr>
<td>FRO37</td>
<td>The system shall allow the user to execute an optimization experiment involving multiple simulation iterations</td>
<td>Must</td>
</tr>
<tr>
<td>FRO38</td>
<td>The system shall allow the user to view the data resulting from each iteration executed in optimization experiment</td>
<td>Must</td>
</tr>
<tr>
<td>FRO39</td>
<td>The system shall allow the user to export the results from all iterations executed in an optimization experiment</td>
<td>Could</td>
</tr>
<tr>
<td>FRO40</td>
<td>The system shall perform iterations of an optimization experiment one at a time</td>
<td>Could</td>
</tr>
</tbody>
</table>
Appendix D: Integration Test Cases

All integration test cases assume the CaaS platform has already been started and the user has already created an account on the CaaS platform.

CIT-01: Register Data Source Gateway
This test case makes sure that a data source gateway can automatically register itself when it starts up.

**Test Procedure**
1) Navigate to the “My Data Sources” page.
2) Check if there are data sources listed with your account as their owner. If this is the case, then the data source gateways were able to register themselves and the test case is complete. If not, continue with the next steps.
3) Navigate to the “Profile” page.
4) Copy the text under “API Key”.
5) Paste this text in the file called “.env” at the root of the project. Replace the text after “API_TOKEN=” with your API Key.
6) Restart the CaaS platform.
7) After the CaaS platform has restarted, there should be data source gateways registered with your account as the owner.

CIT-02: Register Model Gateway
This test case makes sure that a model gateway can automatically register itself when it starts up.

**Test Procedure**
1) Navigate to the “My Models” page.
2) Check if there are models listed with your account as their owner. If this is the case, then the model gateways were able to register themselves and the test case is complete. If not, continue with the next steps.
3) Navigate to the “Profile” page.
4) Copy the text under “API Key”.
5) Paste this text in the file called “.env” at the root of the project. Replace the text after “API_TOKEN=” with your API Key.
6) Restart the CaaS platform.
7) After the CaaS platform has restarted, there should be model gateways registered with your account as the owner.

CIT-03: Edit Data Source Gateway information
This test case tests whether a user can edit the general information about a data source. Only the information in the “General Info” tab can be edited.

**Test Procedure**
1) Navigate to the “My Data Sources” page.
2) Click on the pencil icon next to one of the data sources.
3) Edit some of the general information.
4) Click “SAVE”.
5) Check whether the information has been successfully updated.
CIT-04: Edit Model Gateway information
This test case tests whether a user can edit the general information about a model. Only the information in the “General Info” tab can be edited.

Test Procedure
1) Navigate to the “My Models” page.
2) Click on the pencil icon next to one of the models.
3) Edit some of the general information.
4) Click “SAVE”.
5) Check whether the information has been successfully updated.

CIT-05: View Data Source data table
This test case checks whether a user can view the data that is provided by a data source.

Test Procedure
1) Navigate to the “My Data Sources” page.
2) Click on one of the data sources.
3) Navigate to the “DATA” tab.
4) Confirm that the data provided by the data source is now visible.

CIT-06: View Data Source data schema
This test case checks whether a user can view the metadata that is provided by a data source.

Test Procedure
1) Navigate to the “My Data Sources” page.
2) Click on one of the data sources.
3) Navigate to the “ONTOLOGY” tab.
4) Confirm that information about the different columns provided by the data sources is now visible.
   a. Also, check whether the metadata shown corresponds to the data provided in the “Data” tab.

CIT-07: View Model interface data schema
This test case checks whether a user can view the metadata that is provided by a model.

Test Procedure
1) Navigate to the “My Models” page.
2) Click on one of the models.
3) Navigate to the “INPUTS” tab.
4) Confirm that information about the different model input arguments is visible.
5) Navigate to the “OUTPUTS” tab.
6) Confirm that information about the different model output arguments is visible.

CIT-08: Define simulation with direct input
This test case makes sure that a user can define simulation with one model, while directly entering the input for the model in the simulation definition interface.

Test Procedure
1) Navigate to the “My Simulations” page.
2) Click on “ADD SIMULATION”.
CIT-09: Define simulation with input from a data source
This test case makes sure that a user can define simulation with one model while binding a model input to a data source.

Test Procedure

1) Navigate to the “My Simulations” page.
2) Click on “ADD SIMULATION”.
3) Enter arbitrary text in the “Name of simulation” and “Description of simulation” fields on the “GENERIC INFO” tab.
4) Navigate to the “DATA SOURCES” tab.
5) Select the “Recipe” data source by clicking on the “Select” button next to the data source.
6) Navigate to the “MODELS” tab.
7) Select the “NutritionModel” model by clicking on the “Select” button next to the model.
8) Navigate to the “BINDINGS” tab
9) Copy the following image:

10) Click “SAVE”.
11) If no errors occurred, the test has passed.
a. Navigate to the “DATA INPUT” tab.
10) Enter the following data:
   a. Dosage: Input size 1
      i. Dosage: 250
12) Click “SAVE”.
13) If no errors occurred, the test has passed.

CIT-10: Define simulation with chained models
This test makes sure that a user can define a simulation that contains a model chain.

Test Procedure
1) Navigate to the “My Simulations” page.
2) Click on “ADD SIMULATION”.
3) Enter arbitrary text in the “Name of simulation” and “Description of simulation” fields on the “GENERIC INFO” tab.
4) Navigate to the “MODELS” tab.
5) Select the “PasteurizationModel” model and the “ShelflifeModel” model by clicking on the “Select” button next to the models.
6) Navigate to the “BINDINGS” tab.
7) Make sure everything is set to “input, except for the “StartingMicrobes” under “ShelflifeModel”. Copy the following image:

```
<table>
<thead>
<tr>
<th>Column</th>
<th>Source type</th>
<th>Bound column</th>
</tr>
</thead>
<tbody>
<tr>
<td>amount</td>
<td>data</td>
<td>Recipe.Amount</td>
</tr>
<tr>
<td>name</td>
<td>data</td>
<td>Recipe.Ingredient</td>
</tr>
</tbody>
</table>
```

8) Navigate to the “DATA INPUT” tab.
9) Fill in at least one row for each of the input tables
10) Click “SAVE”.
11) If no errors occurred, the test has passed.

CIT-11: Run simulation with direct input
This test checks whether a simulation that is defined with direct input can be run.
Test Procedure

1) Perform all the steps from CIT-08.
2) Navigate back to the “My Simulations” page.
3) Click on the play button next to the simulation you created.
4) If no errors occurred, the test has passed.

CIT-12: Run simulation with input from a data source
This test checks whether a simulation that is defined with a model input bound to a data source can be run.

Test Procedure

1) Perform all the steps from CIT-09.
2) Navigate back to the “My Simulations” page.
3) Click on the play button next to the simulation you created.
4) If no errors occurred, the test has passed.

CIT-13: Run simulation with chained models
This test checks whether a simulation that is defined with a model chain can be run.

Test Procedure

1) Perform all the steps from CIT-10.
2) Navigate back to the “My Simulations” page.
3) Click on the play button next to the simulation you created.
4) If no errors occurred, the test has passed.

CIT-14: View result of simulation with a single model
This test checks whether a user can view the results of a model run.

Test Procedure

1) Perform all the steps from CIT-11.
2) Click on the simulation.
3) Navigate to the “RUNS” tab.
4) Click on one of the runs.
5) Confirm that the results of the model run are now visible.
Appendix E: Ontology Vocabulary for Data Schemas
Appendix F: CaaS platform Page

Screenshots

My Data Sources
Data Source – General Info

Inspect Data Source: Recipe

General Information

- Created By: Niels Roed
- Owned by: Technical University Eindhoven
- Price: T€/call

Data Source Permissions

- This data source is not shared

Connection Details

- Gateway URL: https://ingredient-access-gateway:5020/api/recipe
- Status: CONNECTED

Data Source – Data

Inspect Data Source: Recipe

Data provided by data source

<table>
<thead>
<tr>
<th>Amount</th>
<th>Ingredient</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>Sugar</td>
<td>%</td>
</tr>
<tr>
<td>11</td>
<td>Vinegar 12%</td>
<td>%</td>
</tr>
<tr>
<td>1</td>
<td>Salt</td>
<td>%</td>
</tr>
<tr>
<td>10</td>
<td>Tomato puree</td>
<td>%</td>
</tr>
<tr>
<td>15</td>
<td>Tomato paste</td>
<td>%</td>
</tr>
<tr>
<td>52</td>
<td>Water</td>
<td>%</td>
</tr>
<tr>
<td>1</td>
<td>Molasses</td>
<td>%</td>
</tr>
</tbody>
</table>
Data Source – Ontology

My Models
Model – General Info

Model – Input
Add Simulation – General Info

Add a new simulation

General info

Name of Simulation
Get all tastes

Association of simulation
Calculate all the tastes given a recipe

Add Simulation – Data Sources

Add a new simulation

Data Sources to use

Recipe

Search Model

Ingredients database

Ingredient property database

Hex

My tests
Add Simulation – Models

Add Simulation – Bindings
Add Simulation – Data Input

Simulation – General Info
Simulation – Runs

Simulation Run
About the Author

Niels Rood received both his bachelor's degree in Software Science (2016) and his master's degree in Computer Science and Engineering (2019) at Eindhoven University of Technology (The Netherlands). During his master’s program, he spent one year on the student team Blue Jay Eindhoven (2017) working on an application of drones to support lifeguards at indoor swimming pools. For his master’s thesis, *Functional Safety Analysis and Safety Pattern Application on i-CAVE*, he designed a safety analysis and improvement method for the software architectures of cooperative vehicle systems based on the ISO 26262 standard. His interests are in Embedded Software and Software Architecture.