

TU UB
Die approbierte Originalversion dieser Diplom-/
Masterarbeit ist in der Hauptbibliothek der Tech-
nischen Universität Wien aufgestellt und zugänglich.
<http://www.ub.tuwien.ac.at>

TU UB
WIEN Universitätsbibliothek

The approved original version of this diploma or
master thesis is available at the main library of the
Vienna University of Technology.
<http://www.ub.tuwien.ac.at/eng>



FAKULTÄT
FÜR INFORMATIK
Faculty of Informatics

Smart food sharing across smart cities: concepts, processes and infrastructure

MASTERARBEIT

zur Erlangung des akademischen Grades

Master of Science

im Rahmen des Studiums

Software Engineering & Internet Computing

eingereicht von

Konrad M. Steiner, BSc

Matrikelnummer 0927159

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: A.o. Univ. Prof. Dr. Dipl.-Ing. eva Kühn

Wien, 7. Dezember 2017

Konrad M. Steiner

eva Kühn

Smart food sharing across smart cities: concepts, processes and infrastructure

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Software Engineering & Internet Computing

by

Konrad M. Steiner, BSc

Registration Number 0927159

to the Faculty of Informatics

at the TU Wien

Advisor: A.o. Univ. Prof. Dr. Dipl.-Ing. eva Kühn

Vienna, 7th December, 2017

Konrad M. Steiner

eva Kühn

Erklärung zur Verfassung der Arbeit

Konrad M. Steiner, BSc
Leipziger Straße 14/22, 1200 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 7. Dezember 2017

Konrad M. Steiner

Acknowledgements

First, I want to thank my supervisor, Prof. Eva Kühn for suggesting such an interesting topic for my master thesis. Her various and manifold ideas guided me during the writing of my theses, but never restricted me. I am very glad that Eva, despite her full appointment calendar, has always managed to take time for a meeting with me. I want also to thank Dagmar Haier from Foodsharing for sharing her extensive knowledge about food sharing with me and giving me insights about Foodsharing which I would not have received otherwise. A big thank you belongs to my cousin Marlena Zeichner for proofreading this work.

At this point I also want to thank two of my fellow students and best friends Christian Kühmayer and Christian Schnitzer for their input in all the group works we did together during our studies. A very special thank you is for my parents Edeltraud and Hans-Peter Steiner for all the support they gave me during my extensive education. Last but not least I want to thank my girlfriend Gerda Mitterlehner for her mental support and kindness particular in stressful times of my studies.

Kurzfassung

Um überleben zu können, benötigt jeder Mensch Lebensmittel und daher ist eine angemessene Versorgung der gesamten Menschheit mit Essen wichtig. Offensichtlich ist das nicht der Fall - Nahrungsmittelabfälle und Mangelernährung sind die Folgen. In letzter Zeit versuchen immer mehr food sharing Initiativen das Problem zu lösen, indem sie überschüssige Lebensmittel weiterverteilen, sind dabei aber nur mäßig erfolgreich. Deshalb ist das Ziel dieser Arbeit einen smarten food sharing Prozess zu beschreiben. Sie beschäftigt sich mit vielen Aspekten von food sharing und bietet Vorschläge für Verbesserungen. Ganz besonders wird das Koordinationsproblem das food sharing innewohnt beleuchtet und mit dem Peer Model gelöst.

Zu Beginn gibt die Arbeit Einblicke in die globale Lebensmittelversorgung und schätzt das mögliche Potential von food sharing ab. Danach analysiert sie bestehende food sharing Lösungen und Forschung über food sharing und gibt Verbesserungsvorschläge. Anschließend werden mehrere Informations- und Kommunikationstechnologien beschrieben, die in einem verbesserten food sharing Prozess verwendet werden können. Dabei vergleicht sie gängige Koordinationsframeworks und kommt zu dem Ergebnis, dass für food sharing das Peer Model am besten geeignet ist. Deshalb wird das Koordinationsproblem von food sharing mit dem Peer Model modelliert, für seine Java Laufzeitumgebung implementiert und eine Simulation zeigt, dass das Model funktioniert. Abschließend beschreibt die Arbeit einen smarten food sharing Prozess und Storyboards veranschaulichen wie Benutzer damit interagieren.

Die Weltbevölkerung mit Essen zu versorgen wird auch zukünftig herausfordernd, vor allem wenn sie wie erwartet steigt. Mit food sharing können heute bereits einige Probleme in der Lebensmittelversorgung verbessert werden, aber aktuelle Lösungen sind nicht effizient genug. Ein smarterer food sharing Prozess, wie in dieser Arbeit beschrieben, sollte daher so bald als möglich eingesetzt werden.

Abstract

Everybody needs food to survive and therefore an appropriate supply of all people with foodstuff is of enormous importance. Obvious this is still not ensured - food waste and undernourishment are direct consequences. Lately, emerging food sharing initiatives try to solve these problem by redistribution of surplus food products, but they are not effective enough. So, the aim of this work is to describe an improved food sharing process. Therefore, the thesis deals with numerous aspects of food sharing and provides several prospects to improve current food sharing solutions. Especially the coordination problem that is inherent to food sharing is considered in detail and solved with a coordination framework called Peer Model.

The thesis starts with literature review to get insights about food waste as well as undernourishment and evaluates the possible potential of food sharing. Then it analyses current food sharing solutions and research about food sharing and names improvement opportunities. Afterward it describes several information and communication technologies that can be used for an enhanced food sharing process. At that it compares popular coordination frameworks and comes to the result that the Peer Model is suited best to deal with the coordination problem of food sharing. Thus, the thesis models the coordination parts of food sharing with the Peer Model, implements them for its Java runtime environment and a simulation shows that the model works. Finally, the work describes an advanced smart food sharing process and storyboards illustrate how users interact with it.

Supplying the world's population with food will be a challenging task in the future, particularly when the expected growth in population occurs. Today's food sharing solutions can improve some problems of the food supply chain, but current solutions are not efficient enough. Therefore, a smart food sharing process like the one described in this thesis should be implemented as soon as possible.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Aim of the work	2
1.3 Methodological approach	2
1.4 Structure of the work	3
2 Global food situation survey	5
2.1 Food losses and food waste	5
2.2 Food insecurity and undernourishment	8
2.3 Prospects of enhanced assignment	11
2.4 Summary	12
3 State of the art in food sharing solutions and research	13
3.1 Literature and research approaches	14
3.2 Deployed solutions for food sharing between households	23
3.3 Summary and open issues	27
4 State of the art in ICT suited for food sharing	29
4.1 RFID	29
4.2 Smart fridges	31
4.3 Autonomous goods transport	33
4.4 Others	35
4.5 Summary	35
5 Coordination models	37
5.1 Selection of a suitable coordination model	37
5.2 Peer Model	40
5.3 Summary	47
	xiii

6 S-FOSM: a Smart FOod Sharing Model	49
6.1 Modelling the S-FOSM	49
6.2 Implementation of the S-FOSM	59
6.3 Using S-FOSM in a food sharing simulation	64
6.4 Summary	70
7 S-FOSS: a vision of a Smart FOod Sharing Solution	71
7.1 Properties of the S-FOSS	71
7.2 The S-FOSS in action	76
7.3 Summary	83
8 Conclusion	85
8.1 Summary	85
8.2 Future work	86
List of Figures	89
List of Tables	90
List of Algorithms	91
Acronyms	93
Bibliography	95

Introduction

As being the first of the thesis this chapter introduces the topic. At the beginning in section 1.1 the motivation for this work is illustrated. Then section 1.2 describes the aim of the work. Afterwards section 1.3 names the methodological approach to reach these goals. Finally, section 1.4 will explain how the further work is structured.

1.1 Motivation

Worldwide each day unimaginable amounts of food get wasted because people cannot or do not want to eat it. Wasted food is a useless dissipation of resources and pollutes the environment. Paradoxically at the same time for lots of people there is no food available or they cannot afford it and hence they suffer from hunger. This undernourishment influences the whole life of the concerned people and makes them more susceptible for illness and less productive.

So, obvious there exists a problem of proper splitting the available food to the world population. These issues are already known and people and organisations have founded food sharing initiatives that redistribute surplus food to humans who need it. Food sharing solutions contribute in reducing food waste and feeding the ones in need. However, as there still so much food ends up in trash and there are lots of people undernourished, food sharing initiatives must be improved.

These days many cities develop to smart cities. They use many information and communication technologies (ICTs) to optimize resource utilisation, reduce costs and increase the quality of life of its residents. Smart city solutions are convenient to use and the people can enjoy their advantages. While at the beginning smart cities were mainly restricted to the energy and mobility domain now they also deal with topics like smart buildings, water, waste and pollution management and emerging concepts like urban farming. Food sharing would also fit into this concept. It increases resource utilisation and reduces

costs by decreasing food waste. Furthermore, increased food security would improve the quality of life for the inhabitants. Unfortunately, current food sharing processes are not convenient enough to use and therefore not smart enough for smart cities. Lots of people do not get the advantages of food sharing (e.g. reduced costs) because the high effort inherent to current food sharing solutions scare them off.

1.2 Aim of the work

This work has several goals that are all related to food sharing:

- The **1st goal** is to determine the extent of food waste and undernourishment as well as in what way food sharing can improve these problems.
- The **2nd goal** is find reasons why existing food sharing solutions are not effective enough (i.e. there is still so much food waste and lots of food insecure people)
- The **3rd goal** is to give examples of ICTs and how they could improve food sharing solutions and make them more effective and convenient for its users. Most important is to find a way to deal with the coordination problem that is inherent to food sharing.
- The **4th goal** is to provide a solution for the coordination problem for food sharing.
- The **5th goal** is to describe a smart food sharing solution. It must integrate various ICTs and the solution for the coordination part to be more convenient for its users.

1.3 Methodological approach

To reach the previously defined goals a couple of different methodological approaches have been applied:

- **Literature review:**

At the beginning of this work literature about food sharing and related topics (e.g. food waste) has been reviewed. Thereby the 1st, the 2nd and the 3rd goal of this thesis could be reached.

- **Modelling, implementation and simulation:**

The coordination problem of food sharing was solved using the Peer Model. For this task the first step was to model the food sharing process with the graphical notation of the Peer Model. Then this model was implemented and source code for the Java runtime of the Peer Model was written. Finally, a simulation environment was created where food sharing was simulated based on this model. Using these methodological approaches the 4th goal of the thesis was achieved.

- **Prototyping:**

For the description of the smart food sharing solution there was a focus on characterising the interaction of the user with the system. Therefore, several prototypes of the graphical user interface (GUI) of the smart food sharing solution have been created and helped to reach the 5th goal of this thesis.

1.4 Structure of the work

The remaining work is structured as follows:

- **Chapter 2** is a survey of the global food situation. It will deal with the main problems of food waste and food insecurity in more detail. Furthermore, it analyses how far an improved assignment can facilitate both of these problems and gives thereby the possible potential of food sharing.
- **Chapter 3** reviews literature about food sharing in the modern human society to find out which problems are already solved and what is open. In addition, already deployed food sharing solutions that facilitate food sharing between households are analysed to find out their weaknesses.
- **Chapter 4** describes various ICTs and how they can be used to make food sharing more convenient for users. These technologies should be integrated in food sharing solutions to make their processes smart and suitable for modern smart cities.
- **Chapter 5** states why a coordination model is essential to deal with such a complex coordination problem like food sharing. Then different coordination models are briefly described and compared. It is explained why the Peer Model will be used in this thesis and a detailed description of the Peer Model is given.
- **Chapter 6** proposes a solution for the coordination parts of food sharing between households using the Peer Model. At first the process of food sharing is modelled with the graphical notation of the Peer Model. Then this model is translated to executable source code for the Java runtime of the Peer Model. Finally, a simulation environment is created where simulated households use this process to share food among each other what proves that the model is a valid solution for food sharing.
- **Chapter 7** describes the vision of a smart food sharing solution for households. This smart food sharing solution uses various ICTs and concentrates on coordination to be as convenient as possible for its users. At first the properties of this system are specified and then storyboards give examples how fictional persons interact with the smart food sharing solution and enjoy its advantages.
- **Chapter 8** concludes this work. It provides a summary of the thesis and gives hints for future work in this area.

Global food situation survey

Everybody needs food to survive. Having enough and adequate food is one of the most important requirements for humans to be healthy and fit. Furthermore, food has a strong social and cultural component as it brings together people and has formed different local customs. Hence the supply of all people with sufficient and appropriate foodstuff is of extreme importance. Sadly, currently this does not work and there is a multitude of people suffering from hunger. Additionally, there are people that get more food than they can eat and thus waste it. These troubles in the global food situation lead to health problems, wastage of limited resources, environmental damage and has also negative economic impacts.

At first this chapter deals with two major problems in the global food supply: food losses and food waste (see section 2.1) as well as food insecurity and undernourishment (see section 2.2). Then section 2.3 analyses in what way an enhanced assignment would improve these matters. Finally, section 2.4 will sum up the most important facts about the global food situation.

2.1 Food losses and food waste

In 1981 the Food and Agriculture Organisation of the United Nations (FAO) had defined food loss as the weight of all food that was produced for human consumption, but do not get eaten by a human [FAO81]. In the literature (e.g. [GCS⁺11]) food losses are often divided into two parts: The term food loss is only used for decreases of quality and quantity of food due limitations in the food supply chain (FSC) and food waste relates to lost food because of wrong behaviour. Food losses rather occur at the beginning of the FSC (production, post-harvest and processing steps) and food waste mostly at the end (retail and final consumption). Other papers (e.g. [SJQM16] and [PBM10]) call all food losses food waste.

There is a multitude of studies about food loss and food waste. Most of them consider only a specific stage of the FSC and a bordered geographical region. As these studies often use different methods (e.g. questionnaire surveys, waste analyses and inferential analyses) as well as different definitions of food losses and food waste (e.g. inedible parts of the food like nutshells are sometimes considered as food loss, sometimes not and food that is used for alternative applications like pet feeding is sometimes counted as food waste and sometimes not) it is hard to compare them and get an aggregated view about food losses and food waste [SJQM16][PBM10][MFI⁺16]. Furthermore, for some regions or stages there is no data available. Therefore amounts of food losses and food waste in the literature always have to be handled with care and papers that provide an overall view have a high degree of uncertainty [GCS⁺11][SJQM16].

The next subsection (see subsection 2.1.1) gives details about the amount of lost and wasted food. Then subsection 2.1.2 describes some negative impacts of food losses and waste and finally subsection 2.1.3 gives some ideas to decrease the quantity of lost and wasted food.

2.1.1 Magnitude of food losses and food waste

As already denoted, exact data about global food losses and food waste are not available. Lundqvist et al. estimate in their often-cited work [LdFM08] from 2008 that about 50% of all produced food is lost or wasted. The newer and more detailed work [GCS⁺11] from 2011 by Gustavsson et al. comes to the result that each year about 1.3 billion tons of food produced for human consumption are lost or wasted. This is approximately one-third of the whole food production.

Lost and wasted food accrue all over the world, but the amount and the reasons differ between developing and developed regions of the world. As can be seen in figure 2.1 the per capita food loss and food waste of high income countries is considerably higher than in low income countries. For example, the per capita food loss and waste of North America and Oceania is with nearly 300 kg/year more than twice as high as in South and Southeast Asia with about 125 kg/year. Considering the food losses and food waste only at customer level the difference is even more major. The per capita food loss and food waste at customer level in Sub-Saharan Africa is only 6 kg/year while it is in North America and Oceania with 115 kg/year about 19 times higher [GCS⁺11]. Food losses and food waste at production and retailing stages of developed and developing regions are on a comparable level but the reasons are different. In developing countries mainly process limitations like missing cooling facilities and inefficient harvesting techniques cause a reduction of quality and quantity of food (i.e. the food gets brackish or is lost on the fields) [PBM10]. However, in medium and high income regions the reasons for food losses and food waste are mostly consumer behaviour (e.g. insufficient purchase planning) and a lack of coordination (e.g. overproduction) in the FSC.

Stenmarck et al. give in their work [SJQM16] more details about food losses and food waste for a high-income region, the former EU-28 states. The per capita food loss and

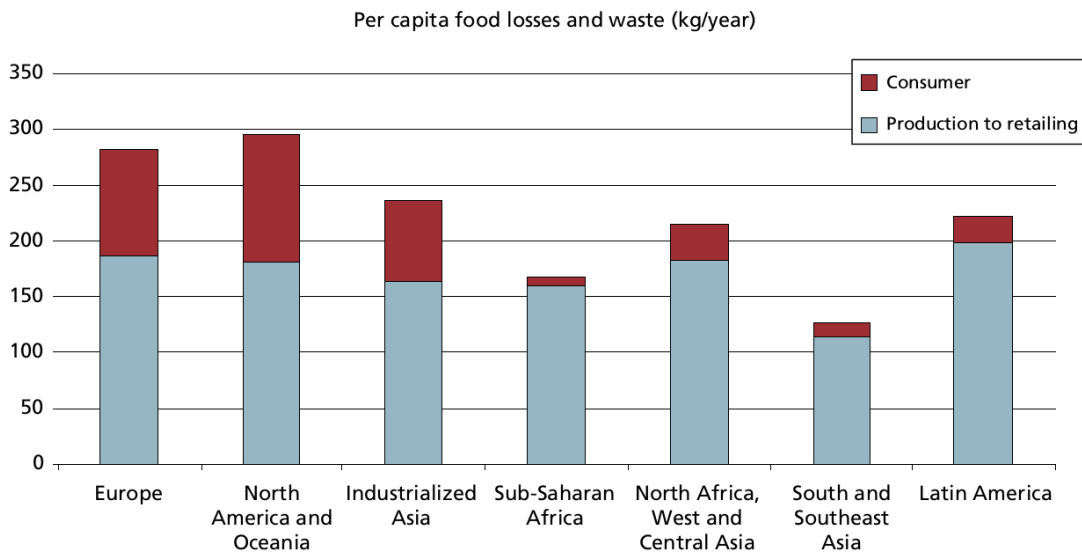


Figure 2.1: Per capita food losses for different regions (from [GCS⁺11])

food waste is different to [GCS⁺11] (173 kg/year in [SJQM16] for EU-28 vs. 280 kg/year in [GCS⁺11] for Europe). This is the case because [SJQM16] does not count food losses and food waste that are fed to animals as food losses and food waste. Furthermore [SJQM16] counts inedible parts of food also as food loss or food waste. In the former EU-28 households are responsible for the main part of food losses and waste. Their share in food losses and food waste is 53% (or per capita 92 kg/year) and about 60% are edible parts. Processing is responsible for 19% (50% edible), food service for 12% (59% edible), production for 11% (50% edible) and wholesale and retail for 5% (83% edible) of the food losses and food waste in the former EU-28 countries.

2.1.2 Negative impacts of food losses and food waste

Lost and wasted food causes several problems. The literature deals with the following drawbacks of food losses and waste particularly intense: The stress of the environment and the economic damage. But lost food and wasted food has also negative influence on the society as it decreases the food security of poor people and cumpers the efforts to combat the hunger in the world [GCS⁺11].

Food losses and food waste are a big dissipation of resources and stress the environment. For example, in the US more than 25% of the whole freshwater demand [HGDC09] and worldwide about 306 km³ of water per year is needed to produce edibles that get wasted [FAO14]. Both, the production of food and its rotting in landfills produce considerable amounts of greenhouse gases [HGDC09]. Only the production of all food losses emits greenhouse gases in the amount of 3.49 Gt CO₂e per year [FAO14] what is about 7 % of the global greenhouse gas emission of 2010 (49.5 Gt CO₂e) [VZA⁺14].

Wasting and loosing food costs a lot of money. In 2012 the edible parts of the food wasted by households of the former EU-28 states are estimated to 98 billion Euros, the overall edible food losses in these countries worth 143 billion Euros [SJQM16]. WRAP comes in [QJ09] to the result that in the UK each year edibles amounting 12 billion pounds are wasted by households and that an average UK family spends 480 pounds per year on food waste. In South Africa, the calculated cost of a ton of food waste is 592 US-dollar. The overall worth of wasted food in South Africa is 7.5 billion US-dollar per year what is about 2.2% of its gross domestic product (GDP) from 2013 [dLN15]. All food losses worldwide value about 936 billion US-dollars per year. Furthermore, the costs of the societal impacts (e.g. soil erosion and water pollution) of food losses are estimated to 1.2 trillion US-dollars annually [FAO14].

2.1.3 Countermeasures against food losses and food waste

In low-income countries, most food losses occur because of limitations in harvesting, storing, packing and transporting the food products [PBM10]. Therefore, for developing countries technical and managerial improvements at production stage are needed to reduce their food losses and increase their food security [GCS⁺11].

In contrast in medium and high-income regions mainly wrong behaviour and missing coordination in the FSC lead to wasting of food [GCS⁺11]. As customers in developed countries have the biggest contribution to food waste a change of their behaviour would be the biggest input for reducing food losses and food waste [SJQM16]. Different studies (e.g. [GFC13]) have already shown that these people in most cases do not want to waste food, it happens accidentally. Already small assistance (e.g. making the food waste visible) can have huge impacts [TCM⁺12]. Moreover, there are various initiatives with the goal to decrease food waste at different stages of the FSC (e.g. food banks or surplus food used as animal feed) [FAO14]. Additionally, currently there is an increasing interest in food sharing as one method that reduces food waste, especially at customer and retailer level [MFI⁺16].

2.2 Food insecurity and undernourishment

The term of food security was defined at the World Food Summit (WFS) in 1974 and redefined to be more precise over the years [FAO03]. The current definition is "Food security (is) a situation that exists when all people, at all times, have physical, social and economic access to sufficient, safe and nutritious food that meets their dietary needs and food preferences for an active and healthy life." and was introduced in [FAO01] in 2001. This widely accepted definition implies that four criteria must be fulfilled that a person can be called food secure: the availability of sufficient food (domestic production or imports), economic and physical access to the food, an adequate utilisation of the food by that person (is closely related to its health status) and the stability (i.e. that the person has access to enough food at all time) [FIW15][Gib12][FAO06]. People are food insecure if at least one of these conditions do not meet [FAO03].

It is very costly to determine food insecurity for a person or a household. Therefore, studies often use proxy measures to gain insights about some aspects of food security or use models to estimate food security for bigger groups of people based on existing data. The most often cited food insecurity data is from the FAO [Bar10]. They use the term undernourishment or hunger for people that do not get enough food to satisfy their energy requirements for at least one year. Based on national food balance sheets they estimate the number of undernourished people worldwide [FIW15]. The United States Department of Agriculture (USDA) uses the term food insecurity for people who do not get food whose energy amounts 2,100 kcal per day. They use two different models, one demand oriented and one supply oriented to calculate approximate values of food insecure people in 76 low and middle income countries [RKB16].

The next subsection (see subsection 2.2.1) deals with the amount of undernourished and food insecure people. Then subsection 2.2.2 describe some negative impacts undernourishment and finally subsection 2.2.3 briefly discusses some ideas to reduce food insecurity and undernourishment.

2.2.1 Magnitude of food insecurity and undernourishment

Although undernourishment was decreasing in the last decades it is still a big problem. According to the FAO from 2014 to 2016 there were about 795 million people suffering from hunger what was around 10.9% of the whole population [FIW15]. Some literature claim that this number is too low. They count people with micronutrient deficiencies (e.g. iron or vitamin A) as food insecure and therefore they set 2 billion people as lower bound of food insecurity [Bar10][PA09]. Hunger is rather a problem of developing countries, but there are also undernourished people in developed states [FIW15].

For developing countries, the absolute number of undernourished people decreased from 991 to 780 million and the prevalence of hunger dropped from 23.3% to 12.9% from 1990-1992 to 2014-2016, but the targets of WFS and Millennium Development Goal (MDG) were missed (see figure 2.2) [FIW15]. The USDA had only considered 76 low and middle income countries and used two different models to estimate the number of food insecure people. With their demand oriented model, they came to the result that in 2016 in this region 607.1 million people (16.9% of the population) were food insecure and that this value will drop to 250.7 million (6%) in 2026. On the other hand with the supply-oriented model the food insecure share is 12% (431.3 million people) in 2016 and will increase to 13.6% (570.3 million) in 2026 [RKB16].

In the developed regions of the world in 2014-2016 there were 15 million people undernourished [FIW15]. Furthermore, in the high-income country US in 2015 12.7% of the households (15.8 million) had at least at some time during the year problems to provide enough food for its members [CJRGS16].

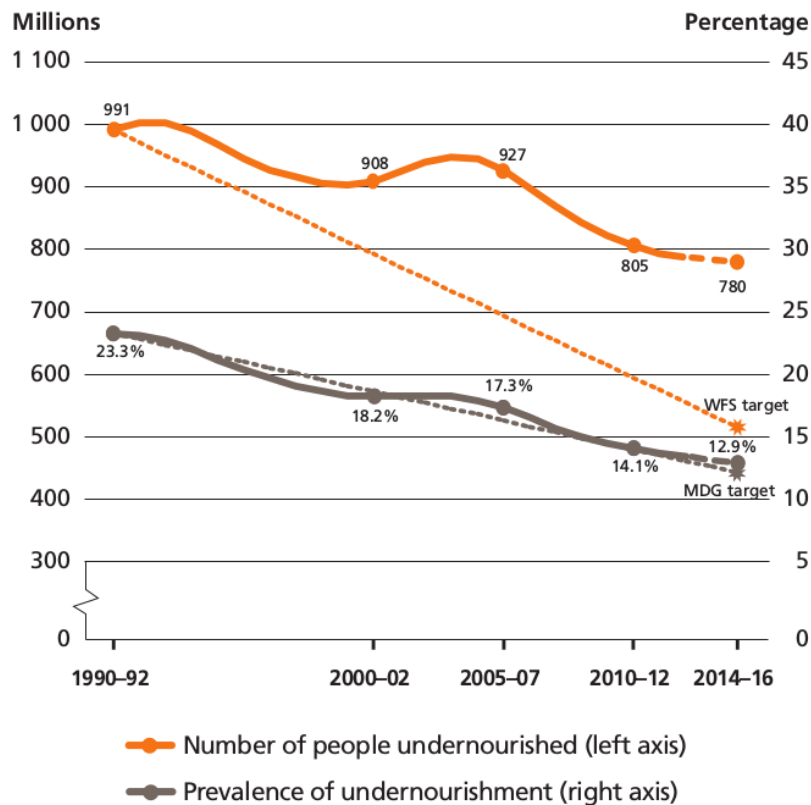


Figure 2.2: Change of the magnitude of undernourishment in developing countries over the time (from [FIW15])

2.2.2 Negative impacts of food insecurity and undernourishment

In case of food insecurity and undernourishment the concerned people feel the negative effects directly. The hunger reduces their health status. Furthermore, the undernourishment decreases their productivity and can even cumber the economic performance of a country.

Malnutrition is the most important risk factor for illness and death. In particular, people in developing countries cannot get sufficient and balanced food and therefore they are more susceptible for infections that cause illness, disease and death [MK05]. Young children are especially concerned by undernutrition. In 2011 malnutrition was the cause for 3.1 million child deaths (45% of all child deaths) [BVW⁺13].

Undernutrition hinders the development of children and influences them negatively for their whole life [BVW⁺13]. Bad nourished children have a lower household income, affluence and labour market activity in their adulthood. Moreover hunger reduces the performance of workers and so their income [LU15]. Hence food insecurity also causes an economical damage. It is estimated that worldwide the GDP has lost 6% because of the

impacts of undernourishment. In poor countries this loss is up to 12% [HS13]. It is not easy for the concerned people to get out of this situation. They earn few money because they are undernourished and they are undernourished as long as they earn few money. Furthermore, they cannot feed their children properly and therefore the children's risk of being food insecure in the adulthood is raised [LU15][Bar10].

2.2.3 Countermeasures against food insecurity and undernourishment

Food insecurity and undernourishment can be reduced in many ways. At the moment, there are mainly famine relief organisations that prohibit temporary hunger as a consequence of natural disasters by delivering emergency food aid to developing regions. Additionally, they support projects that increase food security in low-income countries long-term. Assuming that the world population continues to grow in future there will be worldwide efforts needed to supply all people with enough food.

There are lots of famine relief organisations which spend a lot of money to bring food to the hungry people [LW95]. The largest humanitarian agency, the World Food Program (WFP) [CRC12], got in 2016 donations in the height of nearly 6 billion US-dollar (from countries, private and public organisations and individuals) [WFP] to deliver emergency food aid and to conduct projects like school feeding or social protection and safety net programs [OGS10]. Unfortunately, currently emergency food aid deliveries are often to slow and therefore lose parts of their impact. Moreover, all projects to increase food security have to be planned and executed with great care. Otherwise they could negatively influence the local food supply (e.g. displacing of commercial food trade or affecting local prices) and therefore finally decrease food security [Bar10].

It is estimated that the world population will increase and in 2050 will reach 9 billion people. That means that the food demand will also rise and there must be something done to ensure food security for this amount of people. Increasing the productivity in food production (e.g. improved pest management), using crops with higher yield (e.g. disease-resistant varieties of wheat), a change of people's diets (e.g. meat and dairy products are less energy efficient) and expanding aquacultures (e.g. to Africa) are some possibilities [GBC⁺10]. Furthermore a reduction of half of the food waste would cover a quarter of this increased food demand [FAO14]. Hence food sharing can also help to increase food security, particularly in developed countries with high amounts of food waste [GMP14].

2.3 Prospects of enhanced assignment

Obviously there exists a problem of proper splitting the food to the world's population. Currently several people get more than they can eat and others cannot get enough to appease their hunger. An improved dispersal would reduce both the food waste and the hunger. But could an enhanced assignment eliminate the hunger in the world? Unfortunately, this question is not easy to answer, since the detailed composition of the

global food losses and the precise needs of undernourished people are not known. The next paragraph will use existing data to estimate if there would be enough food for all humans.

Equally dividing all food losses (1.3 billion tons per year) to all undernourished people (795 million) would mean that each of them would get about 1635 kg of food per year. An average German man eats about 565 kg per year [RifEuL08]. Under the premise that an average German man is well fed and the food losses have a similar composition than his diet one can suppose that there is enough food produced for all people. However, food losses also contain food that is lost during harvesting and processing (e.g. limitations in harvesting) and therefore cannot be redistributed. But if all customers in industrialised countries would hand over their surplus food (222 million tons per year [GCS⁺11]) to undernourished people instead of wasting it, in average each starving person would get about 277 kg per year what is nearly half of the food consumption of a German man. Depending on how much food they had before this could be crucial to stop their undernourishment. Moreover, only the food waste of the US contains an energy of 150 trillion kcal/year [HGDC09] and could be used to entire fulfil the USDA's nutrition goal of 2,100 kcal/day for more than 195 million people. In his famous documentary film "Taste the Waste" (see [Thu11]) Thurn even claims that only a third of the food wasted by Europe and North America would suffice to feed all the hungry people in the world. Unfortunately, the film does not say how this number comes about.

2.4 Summary

Summarising one can say that already more than enough food to feed all people over the world is produced. However, because of the great amount of food losses and food waste lots of humans are food insecure or undernourished. Since food losses and food waste as well as food insecurity and undernourishment have big disadvantages for the whole humanity all options to reduce them should be utilised, also to cope with the expected growth of population in future in a world with limited resources. Food sharing is one possibility to tackle both of these problems. Therefore, the further parts of this work will deal with food sharing and the next chapter (see chapter 3) will analyse the state of the art in food sharing solutions and research. However, what should not be forgotten is that food sharing cannot eliminate all food waste (e.g. quick perishable products that go bad before they can be redistributed) and that the transfer of surplus food products to people who eat them, needs resources. Therefore, a combination of several measures is needed to eliminate food losses and food waste and ensure food security and adequate nourishment for the world's population now and in future.

State of the art in food sharing solutions and research

For the animality Feister and McGrew defined 1989 in their article [FM89] food sharing as: "transfer of a defensible food-item from one food-motivated individual to another, excluding theft". Unfortunately, such an agreed definition of food sharing does not exist for the modern human society. Davies and Weyemes suggest in their briefing note [DW17] the very broad definition: "having a portion of food with another or others; giving a portion of food to others; using, occupying or enjoying food and food related spaces to include the growing, cooking and/or eating of food jointly; possessing an interest in food in common; or telling someone about food" that also includes collective cooking or exchange information about food. Most other literature (e.g. [CV16], [GFSG14] and [BKHM⁺16]) see food sharing rather only as form of surplus food management. Also for this work food sharing means to decrease food waste and in addition reduce food insecurity by redistributing excess food from private households as well as food producers and retailers, who give it to people who want it or to charitable organisations that need it for their mission. To have the most impact people get the food from others for free. In the further progress of the work this definition of food sharing will be used, except for the next section (see section 3.1) where also literature is analysed that has a broader definition of food sharing.

At the beginning (see section 3.1) this chapter reviews literature about food sharing. Then section 3.2 analyses two major food sharing solutions that are already in use. Finally, section 3.3 will sum up this chapter and point out important open issues in food sharing.

3.1 Literature and research approaches

There is a lot of research about food sharing in the animality (e.g. [JVS11]) and about food sharing in forager populations (e.g. [Mic04]). However, these results cannot be applied to the modern humans directly and therefore they are out of the scope of this work. Literature and research about food sharing in the modern human society was a long time very rare. Meanwhile it seems that more scientists engage with this issue as there is continuous publishing of new works about this wide topic. Also during the writing of this thesis new literature about food sharing in the modern human society appeared and was reviewed in this part of the thesis.

The next two subsections (see subsection 3.1.1 and subsection 3.1.2) will explain different types of food sharing and the worldwide distribution. Then subsection 3.1.3 will have a closer look at ICT supported food sharing initiatives in Dublin and subsection 3.1.4 will analyse a field report about developing and operating a food sharing solution in Italy. Afterwards (see subsection 3.1.5) a case study with users of a food sharing platform will be analysed and the role of food sharing as a way to reduce domestic food waste will be discussed in subsection 3.1.6. Thereafter subsection 3.1.7 deals with trust in food sharing apps and subsection 3.1.8 is about a research project with the goal to design cooling stations for food hand over in the urban area. Finally, subsection 3.1.9 will sum up the most important facts about this section.

3.1.1 Types of food sharing

There is not much research and literature about food sharing but in practice food sharing solutions are widespread and manifold. Davies analysed in her working paper [Dav16] a multitude of food sharing applications and tries different variants to classify them. Finally, she proposes a classification along two dimensions: Mode of sharing and what is shared. The dimension what is shared provides three classes: stuff (e.g. seeds, foodstuffs and food processing utensils), spaces (e.g. growing and preparation spaces) and skills (e.g. knowledge and experience food growing and preparation). The dimension mode of sharing is split in five categories: IIU (informal, illicit or unorganised activities such as foraging, gleaned and freeganism), gifting, bartering, not-for-profit (exchange of products and services against money, but not for profit) and for-profit (monetary exchange for profit)

Figure 3.1 describes each of these 15 types of food sharing (e.g. gifting and space results in providing space to grow food for free) and gives examples of existing food sharing solutions of this type (e.g. Fallen Fruits for Skills and IIU). Actually, it is not always easy to draw the line between gifting, bartering and monetary exchange. The food sharing solution Foodsharing (see subsection 3.2.1) supports its users to offer edibles to other users for free, but they can charge deposit (e.g. bottle deposit) from the acceptor of the food. This is probably the reason why Davis had classified Foodsharing as not-for-profit and not as gifting. In later work this typology of food sharing was slightly adopted and extended (see [DW17] and [DEM⁺17]), but it is essentially the same.

Mode of sharing	I I U	Gifting	Bartering	Not-for-profit	For-profit
What is shared					
Stuff <i>From seeds, to unprocessed and processed foodstuffs including utensils, food waste or compost</i>	Sharing the foodstuff that has been 'liberated', foraged or gleaned e.g. 510 fruits, Berkeley, USA	Providing foodstuff for free e.g. FoodCloud.ie	Swapping foodstuff e.g. Adelaide Hills Produce Swap, Australia	Providing opportunities to offer or collect excess food on a not-for-profit basis e.g. Foodsharing.de	Selling homecooked food that generates income beyond the costs of production e.g. Cookisto, Athens
Spaces <i>From shared growing spaces to shared food preparation or shared eating spaces</i>	Guerilla gardening of public open spaces e.g. Elephant and Castle roundabout, London	Providing spaces for growing for free e.g. The Monroe Sharing Gardens, USA	Providing spaces where food can be acquired in exchange for labour e.g. Neighbourhood foodstores	Providing spaces for people to grow food on a not-for-profit basis e.g. Milwaukee Urban Gardens	Providing spaces for supper clubs e.g. The Underground Supper Club, Dublin
Skills <i>Including the sharing of knowledge and experiences around food from growing to eating and food waste disposal</i>	Identifying places where gleaned or foraging might occur e.g. Fallen Fruit, Los Angeles, USA	Providing skills around growing, e.g. 3000 acres, Melbourne, Australia	Providing opportunities to learn about growing food, swap seeds and produce with other gardeners near you. e.g. Grow stuff, Melbourne, Australia	Providing workshops around nutrition or growing e.g. Hunger mountain co-op, Montpellier, USA	Providing opportunities for travelers to experience homecooked meals with locals e.g. Eat With, global

Figure 3.1: Types of food sharing (from [Dav16])

3.1.2 Magnitude of food sharing solutions

Davies and Weyemes present in their briefing note [DW17] data about the number of food sharing initiatives world-wide. Therefore, the new research approach called creative construction by Davies et al. (see [DEM⁺17]) was used to create a database¹ with food sharing solutions. They identified more than 4000 food sharing initiatives in 100 cities in 44 countries around the world.

The food sharing initiatives are unequal distributed across these 100 cities. While the 10 most active food sharing cities (London, New York City, Melbourne, Berlin, Sydney, Barcelona, Philadelphia, Chicago, Buenos Aires and Vancouver) have 29% of all initiatives the 10 least active have only 2%. Furthermore, food sharing is more widespread in developed countries than in developing countries (see figure 3.2). The high-income region Australia and New Zealand have in average 76 food sharing initiatives per reviewed city, in Africa there are only 13 per examined city.

Most of the food sharing initiatives (54%) share stuff, 33% skills and the remaining ones (13%) space. Gifting is with 49% the most common mode of sharing, selling (for-profit and not-for-profit) is with 35% at the second place. Lots of the food sharing initiatives use ICTs and websites are frequently utilized to enable the food sharing. Smartphone apps on the other hand are only used by 9% of the food sharing initiatives, but these are more famous than food sharing solutions without an app.

3.1.3 ICT supported food sharing initiatives in Dublin

In his master thesis [Mur16] Murphy deals with ICT supported food sharing initiatives in Dublin. In the first step, he deals with the amount and the geographical distribution

¹<http://sharecity.ie/research/sharecity100-database/>

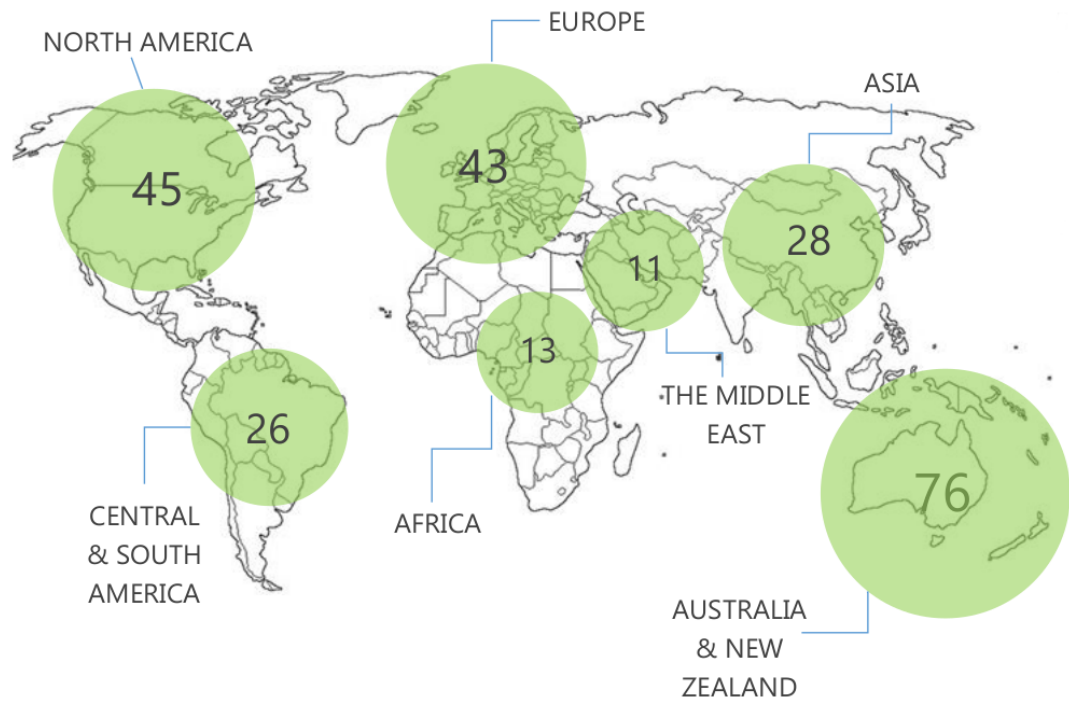


Figure 3.2: Average number of food sharing initiatives per city for different regions (from [DW17])

of them. In the following he selected four food sharing solutions and analysed them in more detail. The following paragraphs will briefly discuss his results.

Geospatial analysis of the food sharing initiatives

Murphy used the SHARECITY100² database to get information about food sharing initiatives and their position in Dublin. He combined this information with public available socio economic data about this city from the Irish Central Statistics Office³. The result was that more than the half of the 15 food sharing organisations are in the city centre. Furthermore, a higher percentage of people in the age between 20-40 years, an income that is marginally above average and a marginally deprived population make an area more likely to have more food sharing initiatives than other areas.

Case studies with four food sharing initiatives

For his case studies Murphy selected four different food sharing initiatives from Dublin: Urban Farm⁴ a non-profit organisation that mainly wants to educate people about food,

²<http://sharecity.ie/research/sharecity100-database/>

³<http://www.cso.ie/en/>

⁴<http://www.urbanfarm.ie/>

Social Hops⁵ where people can grow hops at home and exchange it at local brewers for beer, Urban Oyster⁶ sells kits to grow mushrooms on coffee grounds and the Hardwicke Street Community Garden⁷. Murphy conducted interviews with representatives of this food sharing initiatives and three other interviews with food sharing experts. Furthermore, he observed each of the food sharing organisations for two days to gain insights about them. Based on this data SWOT analyses were performed to identify their strengths, weaknesses, opportunities and threats. Additionally for each of these four organisations individual toolkits to check their economic, social and environmental sustainability indicators were created.

ICT was identified as a meaningful way to facilitate food sharing and create a larger community around a food sharing solution. Furthermore, educating the population about food sustainability was roundly denoted as major factor to further spread food sharing as well as support services for food sharing organisations. Whereas an official of the Irish Environmental Protection Agency mentioned that food regulations are needed, the food sharing initiatives see the current laws rather as obstacle. The toolkits were accepted well by the food sharing initiatives and they want to check their indicators in future from time to time.

3.1.4 Experiences with developing and operating a food sharing initiative

Ciaghi and Villafiorita describe in their conference paper [CV16] their more than five years' experience with BringTheFood⁸, a food sharing solution to share food products in Italy. At the beginning they recognised that only few edibles were exchanged with their web based food sharing platform. They related this to the following conditions:

1. **Low Degree of Recoverability (DoR) of donations:**

The DoR was introduced with the Availability-Surplus-Recoverability-Waste (ASRW) framework and is a function of the intrinsic recoverability (depends among others on the shelf life and the need for refrigeration of a food product) and the management intensity (the costs for a potential acceptor to get the food) of the surplus food. Generally, the DoR of products in the production and retail stage of the FSC is rather high (larger amounts per donation, proper wrapped, ...), at customer stage rather low (small amounts, close to expiration date, sometimes unpacked). Products at food service stage typically have a medium DoR. However, the implementation of the food sharing solution has also an impact on the DoR.

There were often few edibles (e.g. two cups of yoghurt) offered by households. BringTheFood cannot reduce the management intensity for these products sufficiently and hence they are often not collected by other members. However, at

⁵<http://www.urbanfarm.ie/social-hops.html>

⁶<http://www.urbanfarm.ie/urban-oyster.html>

⁷<http://dublincommunitygrowers.ie/gardens/hardwicke-street-community-garden/>

⁸<http://www.bringfood.org/landing/>

the beginning also large amounts of products (e.g 300 kg walnuts) from producers and retailers were not requested. This plenty was often too much for the small charities and private households to take it at once on the whole. Hence, they enabled that acceptors could request smaller portions of a huge food offer what noticeable increased the quantity of shared edibles.

2. **Missing support for logistic:**

At the beginning the users that were registered to accept food were mainly small charities and households. They often could not pick-up the food from the offering producers and retailers (e.g. because they had no refrigerated vans). BringTheFood started to cooperate with larger food banks with own transport fleet. These bigger charities help smaller ones by collecting and transporting foodstuffs for them.

3. **Lack of focus:**

The initial BringTheFood food sharing initiative was intended to facilitate sharing of food products among producers, retailers, charities and households. Because of this magnitude of possibilities, the users did not get the sense of BringTheFood: Households did not want to request food offers because they thought charities can use it more, charities thought it is a tool for food waste reduction and did not use it.

Hence, they decided to provide different solutions for different target groups. With the original BringTheFood web application now food producers give their surplus food to food banks. Food producers get easy and for free rid of their no longer needed food products. On top of that they even get tax reductions for this form of food donation. The charities get big amounts and just need to collect it at the producers site. Retailers can also use the original BringTheFood web solution to give surplus food to charities. In contrast to food producers there is an additional mediator that ensures that the quality of the donations is okay and that the retailers follow some rules. As for their solutions for food service and households the DoR was too low they offer other possibilities to reduce food waste at this stage. For canteens, they started the awareness project "ZeroAScuola", but if anyway food is left over it can be given to selected food banks (food from canteens must be eaten within 24 h) using the "BringTheFood per la Ristorazione" tool. At household level, they promote their Android app "QuantoSpreco?" that should reduce the food waste by monitoring the food stock of families. "QuantoSpreco?" provides food sharing for households too, but they do not expect much exchange of food. Also for other food sharing solutions (e.g. IFoodShare⁹ and Foodsharing) Ciaghi and Villafiorita claim limited effectiveness for food sharing at household level due to limitations in their models.

⁹<http://ifoodshare.org/>

3.1.5 Case study with food sharing users

Ganglbauer et al. performed a case study [GFSG14] with the Facebook group¹⁰ of the food sharing platform Foodsharing (see subsection 3.2.1). Foodsharing uses this Facebook group to discuss topics and invite its members to participate. The case study analysed demographic information about its members. The result was that the majority of their 1012 members is female (69.6%) and that 39% of all users are between 25 and 34 years old. Furthermore 3242 contributions to the group were analysed to find out why people participate in Foodsharing and how the Foodsharing community emerged. The main reason for people to give away surplus food was that they have felt better since they started to do something good and only few users mentioned that they take food because they cannot afford food. For the emerging of the Foodsharing community the members pointed out the following three reasons:

1. **Medial presence:**

When Foodsharing started in 2012 big German TV channels and major newspapers reported about it. Therefore, lots of people were informed about Foodsharing and some of them participated.

2. **Local communities:**

Food sharing is only productive and sustainable if there is a local community. You need a critical mass of members in an area that food sharing works.

3. **Global awareness:**

Food sharing brings together people, who want to reduce food waste. The more people are familiar with the disadvantages of food waste, the more will join the food sharing communities.

Ganglbauer et al. also points out that the Facebook group of Foodsharing might not be representative for Foodsharing, but it gives some insights about Foodsharing and its members. Not all Foodsharing members joined the Facebook group and people that do not use Foodsharing can also contribute to it. They plan to interview Foodsharing users to get more detailed results.

3.1.6 Food sharing to reduce domestic waste

Through the high amounts of food waste at this stage of the FSC food sharing at household level can be an important contribution to reduce food losses and food waste and increase food security. As food sharing within a household (e.g. flat share) and between households differ, they will be treated separately in the next paragraphs.

¹⁰<https://www.facebook.com/foodsharing.de>

Intra-household food sharing

Monore et al. conducted a framed field experiment [MFI⁺16] to examine the relationship between food sharing and domestic food waste. 20 students living in five flat shares (three to five persons) participated. At first all participants filled in a questionnaire with socio-demographic information and questions about their lifestyle and diet. Then in two succeeding weeks the organic food waste of the participants was collected and analysed. In the first week, they were asked to behave normal and in the second week to purchase, cook and consume the food collectively.

They observed an overall increase of the weight of the organic food waste in the second week, but with a closer look they saw that in three flat shares the organic food waste decreased and in two increased. Furthermore, they included data from the questionnaire, the food waste analyses and cross-inspections and recognized some patterns. The residents of the three flat shares that decreased their food waste indicated themselves to be more environmental aware and economic aware than the others. Moreover, the participants from the other two flat shares that increased their food waste lacked in domestic skills (e.g. problems with separating organic waste from other waste) and stand out through non-collaborative behaviours (e.g. leaving the flat share for several days during the experiment).

The conclusion of Monore et al. was that sharing practices could reduce the food waste if people have a certain degree of awareness, skills and collaborative behaviour. In addition, they suggest to repeat their experiment with more participants over a longer period of time to get better results.

Inter-household food sharing

For their conference paper [FWCF14] Farr-Wharton et al. examined the effect of smart phone apps on domestic food waste. In a three-week period 15 participants used three apps that have the goal to reduce food waste on household level: Fridge Pal¹¹ to monitor the food stock of households, LeftoverSwap¹² for food sharing with other households and their own app EatChaFood (see [FWFC13]) that combines both of these features. All participants had to answer a questionnaire regarding the usefulness and the usability of the apps at the end of every week.

The participants denoted that the food monitoring functions of Fridge Pal and EatChaFood helped them to organise their food inventory. It reduced the chance that people forget food and therefore it goes off and gets wasted. However, most participants mentioned that the process of adding to and deleting food products from the monitoring apps compromised to many manual steps and hence they will not use them after the experiment. The food sharing functionality of the apps was controversial for the participants. Lots of them did not want to meet with strangers to hand over food. Generally, the participants

¹¹<https://www.facebook.com/FridgePal/>

¹²<https://play.google.com/store/apps/details?id=com.greasedwatermelon.leftoverswap>

were more willing to give food to others than taking food from other members. However, from known persons or persons within a trusted community the taking of food was more likely. Furthermore, a good condition of food (e.g. packaged products) or a non-personal handover (e.g. in a communal fridge) increased food takings.

The conclusion of Farr-Wharton et al. was that food sharing needs trust and comfort. Only people that have a certain degree of trust in both, the offering partner and the offered product, will take it. The trust in the offering partner increases with the familiarity, but can also be raised through a strong community. Trust in a product depends on the condition of the product. For giving food to other people comfort is essential. People must feel comfortable about giving food to another person.

3.1.7 Trust in food sharing apps

In food sharing people eat food products they have taken from other people. Furthermore, for the handover of the edibles often a personal meeting is necessary. Hence trust is an essential factor for a food sharing initiative. People must have a certain degree of trust in other people and in the food products or they will not take or give edibles (see [FWCF14]). Wruß claims in his master thesis [Wru13] that, if the food sharing solutions are ICT supported, users must also trust in these techniques. Particular mobile applications that support food sharing have to keep trust in mind and therefore Wruß established 20 guidelines (e.g. "let the user control his/her data", "be careful with advertisements", "have features that really satisfy your users" and "use rating systems for users and their actions") for developing mobile applications that users trust in. Furthermore, he described an iOS app for food sharing he had created with respect to his guidelines.

3.1.8 Cooling stations for food handover

There is the ongoing research project UrbanFoodSpots¹³ by Österreichisches Ökologisches Institut. It was started in 2015 with the goal to design cooling stations for food handover in urban areas. These stations will be public available and support the local and private handover of food. The UrbanFoodSpots use ICTs and the operator can always follow who had given or taken what food. For the design of the user interface the diversity of its prospective users was considered so that as much people as possible can use this facility. The UrbanFoodSpots are not intended to replace current modes of food sharing, in fact they should support them and cooperate with local organisations. Another goal of the project is to raise awareness for food waste and food poverty in the population. Furthermore, they want to find feasible prospects for founding and supervision of these stations (see [BKHM⁺16]).

Figure 3.3 shows an early prototype of an UrbanFoodSpot. At the right side in the middle is the user interface where users can select the food product they want to have. Then the appropriate shelf at the left side can be opened and the product can be removed.

¹³http://www.ecology.at/urban_food_spots.htm

In a test run with more than 300 users the prototype worked well. 94% of the users denoted that the interface is comprehensible and 95% indicated that they would use the UrbanFoodSpots for food handover (see [Pla16]).



Figure 3.3: Prototype of a UrbanFoodSpot (from [Pla16])

3.1.9 Summary

This section has shown that besides the variant of food sharing, where surplus food is redistributed to people in need, there are various other forms that can be seen as food sharing. There are many food sharing initiatives in all parts of the world and ICT supports their further spread. Considering food sharing as means to redistribute surplus food there exist successful models that support charities to get excess food from producers and retailers. However, food sharing between households is more difficult. In contrast to food producers and retailers domestic homes have only small portions of food at once. Furthermore, compared to charities they can take smaller amounts of food. This increases the coordination effort. Current food sharing solutions use too simple models for this audience and therefore too little food is exchanged. However, since households have a great potential for food sharing (households are responsible for 53% of the food waste in the EU and 12.7% of the US households had problems to buy enough food in the last year), this thesis will concentrate on food sharing between households. The following section (see section 3.2) will analyse food sharing initiatives for food sharing between households that are already in use in more detail.

3.2 Deployed solutions for food sharing between households

Compared to research and literature about food sharing in the modern human society, creating food sharing solutions became popular earlier. Meanwhile world-wide there are more than 4000 food sharing initiatives with various forms of food sharing. Since this thesis concentrates on supporting the exchange of surplus food between households, this section will only consider food sharing solutions that match these conditions. Substitutional for all other food sharing initiatives that could be used for that task this section analyses two of the most successful ones: Foodsharing (see subsection 3.2.1 and OLIO (see subsection 3.2.2)

3.2.1 Foodsharing

Foodsharing¹⁴ has been provided by the registered association Foodsharing e.V. since 2012. Foodsharing started in Germany and now they have more than 200,000 registered members, mainly from Germany, Switzerland and Austria that prevented more than 8000 tons of food from getting wasted. Everybody can register at their website and become a member of Foodsharing, a Foodsharer as they call it. Foodsharers can create so called food baskets, a virtual representation of their surplus food, or request food baskets from other Foodsharers.

To create a food basket with the Foodsharing website (see figure 3.4) the Foodsharer must describe the food and define if other Foodsharers can request the basket via the Foodsharing chat messages, by telephone or both. The address in the Foodsharers profile is set as the food baskets location and the user can provide some additional information like a photo or the weight of the products. Other Foodsharers can find the food basket on the map and can request it using Foodsharing's chat messages or telephone and arrange the handover of the food products.

Furthermore, some Foodsharing members operate so called Fair-Teiler¹⁵. That are public fridges where anyone can give off excess food or take food left by others. All Fair-Teilers are listed on the Foodsharing website and can be seen on the map. For each Fair-Teiler there is a page with details (e.g. address and opening hours) and a pinboard where Foodsharer sometimes write down what is in the Fair-Teiler at the moment. Figure 3.5 shows the details page of the Fair-Teiler Amtshaus Wien 7 in the seventh district of Vienna as example.

Foodsharing also cooperates with food companies to reduce their food waste. Therefore, Foodsharing and the food company agree on fixed dates where Foodsharing members come to the shop and take the food products that cannot be sold anymore. Food sharing

¹⁴<https://foodsharing.de/>

¹⁵The name Fair-Teiler is compound from the German words "fair" which means fair and "Teiler" which means someone who shares. Furthermore, the word Fair-Teiler is pronounced the same like the German word "Verteiler" which stands for distributor.

3. STATE OF THE ART IN FOOD SHARING SOLUTIONS AND RESEARCH

The screenshot shows a form titled "Essenskorb anbieten" (Offer food basket) with the following fields and options:

- Füge dem Essenskorb ein Foto hinzu** (Add a photo to the food basket): Includes a "Datei wählen" (Choose file) button.
- Beschreibung** (Description): A text input field with a character limit of "maximal 1705 Zeichen..." (maximum 1705 characters...).
- geschätztes Gewicht** (Estimated weight): A dropdown menu currently set to "3,00 kg".
- Wie möchtest Du kontaktiert werden?** (How do you want to be contacted?): Radio buttons for "Per Nachricht" (checked) and "Per Telefon-Anruf" (unchecked).
- Welche Arten von Lebensmitteln sind dabei?** (Which types of food are included?): A list of checkboxes for "Backwaren", "Obst & Gemüse", "Molkereiprodukte", "Trockenware", "Tiefkühlware", "Zubereitete Speisen", and "Tierfutter".
- Was trifft auf die Lebensmittel zu?** (What applies to the food?): A list of checkboxes for "sind Bio", "sind vegetarisch", "sind vegan", and "sind glutenfrei".
- Essenskorb veröffentlichen** (Publish food basket): A green button at the bottom right.

Figure 3.4: Create a food basket with the Foodsharing website

The screenshot shows the details page for "Fair-Teiler Amtshaus Wien 7". The page is divided into several sections:

- Options:** "Fair-Teiler nicht mehr folgen" (Unfollow Fair-Teiler).
- verantwortliche Foodsaver:** A list of users responsible for the food sharing.
- Follower:** A list of users who follow the Fair-Teiler.
- Beschreibung:** "Der Fairteiler befindet sich direkt im Amtshaus und ist zu den Parteienverkehrszeiten zugänglich. Diese sind Mo, Di, Mi und Fr 7:30-15:30, an Donnerstagen 7:30-17:30." (The Fair-Teiler is located directly in the office building and is accessible during party traffic times. These are Mo, Tu, We and Fr 7:30-15:30, on Thursdays 7:30-17:30.)
- Adresse:** "Anschrift: Hermanngasse 24-26", "PLZ / Ort: 1070 Wien".
- Message input:** A text field "Nachricht schreiben..." with "Bild anhängen" (Attach image) and "Senden" (Send) buttons.
- Recent posts:** Three posts from users sharing food items, such as "Noch immer etwas mehr als 2x 10 Liter Kübel Naturjoghurt da." (Still have a bit more than 2x 10 liter tubs of natural yogurt available.)

Figure 3.5: Details page of the Fair-Teiler Amtshaus Wien 7 (Photos and names of Foodsharing users have been made irrerecognisable)

members that pick up food at stores are called Foodsavers. A Foodsharer becomes a Foodsaver after passing a quiz and three introduction pick ups with experienced Foodsavers. When Foodsavers pick up food at cooperating companies they can either eat it themselves, give it to family, friends and charities or use Foodsharing to create a food basket.

3.2.2 OLIO

The OLIO Exchange Limited¹⁶ was founded in 2015 and launched an app for Android and iOS for food sharing. At the beginning, it was limited to some area in North London but now there are nearly 200,000 users (OLIOers) that share food in 41 countries. Anyone can become an OLIOer for free and use the app to share food and non-food products. Products pass over for free, though the provider can request that the acceptor donates money to a charitable organisation. OLIO's vision is to reduce the food waste to zero and that everyone has enough to eat. The OLIO App is available in English only, but the offered products are often described in other languages.

To share food with OLIO you must create a new listing. Therefore, the OLIOer must add a photo, choose offering food, choose whether the product is free or the acceptor must make a charitable donation, insert title, description, pick-up location and pick-up time and decide how many days the item is listed. Figure 3.6 shows the creation of a new listing with the Android app as example. Other OLIOers see the item in their list or on the map and can request them by writing a OLIO chat message. The provider and the acceptor of the food have to agree on time and location of the handover using these chat messages. If they do not want to meet in person, the provider can put the food in a drop box at one of OLIO's partner shops and the acceptor picks it up afterwards.

For people who want to engage themselves for OLIO and food sharing, OLIO provides three volunteer roles: Ambassador, Food Waste Hero and City Champion:

1. **Ambassador:**

Ambassadors promote OLIO in their neighbourhood and help new users. OLIOers must fill in a form and watch some information videos to become an Ambassador.

2. **Food Waste Hero:**

Food Waste Heroes pick up surplus food at stores that cooperate with OLIO and distribute it to other people via the OLIO app. OLIOers can write an email to OLIO to become a Food Waste Hero and they can keep up to 10% of the food they collect.


3. **City Champion:**

City Champions work closely together with OLIO's Head of Communication to launch and embed OLIO in a town. The role is limited for a four to six months'


¹⁶<https://olioex.com/>


3. STATE OF THE ART IN FOOD SHARING SOLUTIONS AND RESEARCH

Add Listing


Photo 


OFFERING

 Food


 Non-Food


WANTED

 Food

 Non-Food

Price

 Free

 Charitable Donation

Title

Yoghurt with sour cherry and poppy seed

Description

200g, best before 05.04.2017, organic food

Pick-up Location

Location set

Pick-up Times

Daily 7-9 pm

Keep Listed For

5 days

Submit




 Browse  Just Shared  Created with **Stitch & Share!**

Figure 3.6: Create listing with the OLIO Android app

period in which City Champions have to work at least 10 hours per week for OLIO. OLIOers can write an email with a covering letter to OLIO if they want to become a City Champion.

3.3 Summary and open issues

Literature has already engaged with some aspects of food sharing. Furthermore, there are various food sharing initiatives that provide food sharing solutions for users. Generally, it is easier to redistribute surplus food from producers and retailers to charities and for this case solutions are already in use that considerably reduce the amount of food waste. Food sharing initiatives for food sharing between households are also deployed, but only with moderate results. The more successful ones managed to create communities where people trust each other as far as that they can meet for food handover and eat food they received from others. Furthermore, primarily in cities they have reached enough active members so that people with food needs often find a suitable food offer and vice versa. Moreover, their members are aware of food waste and some of them not only share their surplus food, they additionally pick up excess food at retailers and producers and share it within the community.

Unfortunately, still the most surplus food of households ends up in trash. This is certainly the case because using these food sharing solutions is too costly and time intensive. Users have to enter all food information by hand, must transport the food themselves and monitor their food stock manually. Even worse is that the coordination between the food sharing people is only supported limited. People can indeed search for food offers of others, but when they want to request it they must write text messages or communicate outside the food sharing solution (e.g. make telephone calls) to arrange date and time for food handover. Therefore, ICTs should be integrated in food sharing processes to make them more comfortable. Then they would be more beneficial for users and better fit into the concept of modern smart cities. This thesis will analyse various ICTs and integrate them in a food sharing to get a smart food sharing process. The next chapter (see chapter 4) is about ICTs that are suited to improve food sharing. As coordination is of extreme importance for a smart food sharing process, coordination models will be analysed separately from other ICTs in an own chapter (see chapter 6).

State of the art in ICT suited for food sharing

As already described in the previous chapter (see section 3.2) current food sharing solutions to share food at household level are quite expensive to use. The users must enter long product descriptions by hand before they can give away food, get no support in monitoring their food stock and transport the edibles themselves to other users.

Therefore, this chapter will analyse three ICTs that support the food sharing process and together with a coordination model (see chapter 5) transform it to a smart food sharing process that is convenient to use and the users can enjoy the advantages of food sharing (e.g. less domestic waste). The next section (see section 4.1) is about radio-frequency identification (RFID)). Then in section 4.2 smart fridges will be discussed before section 4.3 deals with autonomous goods transport. Section 4.4 gives further examples of technologies that could improve the food sharing process and finally section 4.5 sums up the whole chapter.

4.1 RFID

RFID is a general term for many techniques that facilitate wireless short range communication between readers and transponders (also called tags) [Lan05]. It enables identification of objects without visual contact or physical access to them [Wan06]. The first application was military during World War II to differ enemy aircraft from the own planes [Dob12]. Since the 1990s RFID has become more popular and nowadays it is part of everyday life with applications like theft protection, access control and contactless payment [Lan05].

All RFID tags have an antenna to send and receive signals as well as a logic to process the data. There are different variations of RFID tags with varying properties. A detailed

description of all variants can be found in Finkenzellers RFID handbook (see [Fin15]). The following listing categorises RFID tags in two different ways that have the most influence on their properties:

- **Active vs. passive tags**

Passive RFID tags do not have an own power supply. They get the energy needed for sending and processing signals through receiving signals. These tags are smaller, cheaper, more durable and need less maintenance than active tags. Active RFID tags have an external power supply which gives them more range and more computational power than passive ones. Furthermore, there are semi-active RFID where only the computing microchip needs an external power supply but not the antenna [Dob12].

- **Frequency bands**

The used frequency for RFID has effects on the range and the transfer rate. The following frequency bands are used for RFID: Low frequency (LF), high frequency (HF), ultra high frequency (UHF) and super high frequency (SHF) [TT10]. The range of LF and HF RFID tags is comparable to their antenna size (typically a few centimetres up to a metre). Ranges of UHF and SHF tags are limited by transmit power and can reach hundreds of metres. The transfer rate of LF is very low, but therefore LF can go through thin metal and the data transfer is immune to water. The other frequencies provide faster communication [Dob12].

4.1.1 EPCGlobal

There are several standards that regulate RFID in various areas. For merchandise management, the initiative EPCGlobal¹ of GS1² develops standards for RFID supported processes.

EPCGlobal Class-1 Generation-2 is the newest standard of EPCGlobal to identify products or shipping units with RFID [Dob12]. To keep the costs for the tags low the standard defines passive RFID tags which need just 96 Bits of memory. To reach high ranges (up to 7 meters) it specifies the use of UHF frequency bands, but there is also a standard which uses HF and has a limited range (about 1 meter) for countries where UHF is not allowed [Flö05].

The memory of an EPCGlobal certified tag is used to store the Electronic Product Code (EPC), an identification value of the object that the tag is attached to. Currently the EPC is 96 Bits long but depending on its 8 Bit header it can vary in length and format [Dob12]. The EPC is backward compatible and current identification numbers like the Serialized Shipping Container Code (SSCC) for shipping units or the Global Trade Item Number (GTIN) for product classes can be converted to the SSCC-96 respectively the GTIN-96

¹<http://www.gs1.org/epcglobal>

²<http://www.gs1.org/>

format of the EPC [Flö05]. For objects that do not have any existing identification there is the General ID (GID) format of the EPC [TT10].

A EPCGlobal certified reader can read the EPC of all objects in its range. EPCGlobal suggests to use an Application Level Event (ALE) compliant middleware that connects readers and enterprise applications. The middleware receives, filters and groups the RFID events from all readers and hands them over to the enterprise applications. ALE is used to define when and what events the business information systems get [GB06].

The EPCGlobal tags just store the EPC (the identification of an object), but no detailed information. For this, the Object Naming Service (ONS) is used. It returns all sources that provide information for a specific EPC [GB06].

4.2 Smart fridges

Smart fridges are digitally upgraded versions of normal fridges which provide innovative features for its users. Most smart fridges have a touch screen mounted on the outside of their door that provides various information and multimedia content. Furthermore, they are often connected to different sensors and the internet [Rot07][HBSA16].

The following subsections will analyse research and literature about smart fridges (see subsection 4.2.1) as well as smart fridges that can be already bought by consumers (see subsection 4.2.2).

4.2.1 Research and literature

Several works introduced different smart fridges for different purposes. For example, Lou et al. describe in their article [LJL09] a smart fridge that helps its users to improve their nutrition and therefore the health of them. Based on the medical record and other information of the household's members as well as the current content of the fridge it generates shopping lists and meal suggestions for the users. Moreover, it provides detailed information (e.g. nutrition facts and allergens) of all of its current products and sends notifications if one of them will expire soon. The ZmartFRI described by Bucci et al. in their conference paper [BCC⁺10] is intended as information hub that connects the members of a household. They can collectively create a shopping list and post and read messages at the ZmartFRIs display. These interactions can also be done from remote. Furthermore, this smart fridge also monitors its content, the users can query its food stock and get notifications before a product expires. In his conference paper [Rou12] Rouillard shows that at least some functionality of expensive smart fridges can be also accomplished with a smart phone app. Using the Pervasive Fridge app, people can easily monitor their food stock. By scanning its barcode or entering its product information by text or speech they can add an edible to the list. The goal of the Pervasive Fridge app is to reduce food waste and therefore it reminds the users to consume their food before it expires.

The approaches and goals of research for smart fridges are manifold, but most of them have one feature in common: They monitor the food products that are stored inside. Thereby they can remind users to eat food before it goes off, propose recipes based on the available food and user's diets, order food before it goes out and other useful things to support their users. Using RFID the smart fridge can monitor the food products inside, without human interaction [XYL⁺13][HBSA16], but therefore all groceries must be equipped with RFID tags. To capture food without tags their bar code can be scanned or the information can be entered manually by the user [Rou12].

4.2.2 Consumer products

Already in the year 2000 LG launched the first smart fridge, called Internet Digital DIOS (or R-S73CT). It had a big screen at the outside, a web camera to photograph its inside and was connected to the internet. The users could for example monitor their food stock, write memos, play MP3 music or watch TV. Furthermore, it had an automatic ice maker, it needed only half of the energy than other comparable fridges and was with 23 decibels very quiet. On the market the Internet Digital DIOS was not very successful. On the one hand its high price of about 20,000 US-dollar scared the customers, on the other hand they did not see the advantages of these innovations (see [FS16]).

Probably the reduced prices and adapted features have opened be a bigger market for smart fridges. In any case, meanwhile there are more and more manufacturers selling smart fridges. The LG InstaView Refrigerator³ has a glass door and turns on its light when you knock twice on it. Therefore, you do not need to open the door to see what is inside. That saves energy and money. Saving money is also the intention of Whirlpools smart fridge⁴. It tries to shift the energy intensive defrost cycles to times where the current is cheap. Furthermore, the customer can install an app on her or his smart phone to get faster help if the fridge has some problems. However, the most features offers at the moment certainly the Samsung Family Hub⁵ (costs from 3,500 US-dollars). The customer can install various apps (e.g. calendar, recipes and web browser) and interact with a huge touch screen (see figure 4.1) with the fridge. It has a camera that photographs the inside of the fridge regularly. Furthermore, the fridge is connected to the internet and the user can access its information (e.g. the photo from the inside) from everywhere via a smart phone app. The Family Hub also reacts to voice commands and can read aloud media content (e.g. news and recipes). In addition, it can be used to order groceries at local supermarkets.

³<http://www.lg.com/us/refrigerators/lg-LFXS30796D-french-3-door-refrigerator>

⁴<https://www.whirlpool.com/kitchen/refrigeration/refrigerators/french-door/p.36-inch-wide-french-door-refrigerator-with-infinity-slide-shelves-32-cu.-ft.wrf995fifz.html>

⁵<http://www.samsung.com/us/explore/family-hub-refrigerator/overview/>

⁶Photo from the Samsung web page (<http://www.samsung.com/us/>)



Figure 4.1: Samsung Family Hub ⁶

4.3 Autonomous goods transport

Around 1950 companies started to automate their on-site goods transportation and introduced automated guided transport (AGT) systems to increase productivity. Automated guided vehicles Automated guided vehicleless (AGVs) enabled automated, driverless and partially autonomous transfer of goods for these systems. AGVs transported products on predetermined paths and needed markers that guide their way. Furthermore, AGV needed a connection to a central controller that coordinated them [Flä16]. Over the years AGVs got more intelligent and more autonomous. More decisions were taken by the vehicles and not by the central control unit [LADK06]. Nowadays fully autonomous transport vehicles for off-road applications are already available and in use [BCH16]. The following subsection (see subsection 4.3.1) will analyse autonomous driving in more detail.

4.3.1 Autonomous driving

Autonomous moving means of transport are the most important requirement for autonomous goods transport. In aviation autopilot functions have been in use for a long time and steer aeroplanes autonomous over long distances [Flä16]. Autonomous vehicles use radar, laser range finders, cameras, GPS, accurate maps of the environment and other techniques to find their way [GLU12]. Research projects like the Prometheus project or the Autonomous Land Vehicle project proved concepts for autonomous driving in the 1990s. The following paragraphs will examine the role of autonomous driving for cars, trucks and last mile delivery separately.

Cars

By now lots of car manufacturers (e.g. BMW and Mercedes) have assistant systems available that enable partial autonomous driving (e.g. autonomous parking and adaptive cruise control). Furthermore, several car makers (e.g. Nissan and Volvo) and other companies (e.g. Google and Apple) are currently developing and testing autonomous cars and want to release them in the next years. Tesla introduced the AutoPilot system which can drive autonomously with little manual intervention [BCH16]. However, due legal constraints (e.g. the Vienna Convention on Road Traffic) on the roads of many countries there must be a driver that can overtake car control from the autonomous driving system.

Trucks

Truck manufacturers (e.g. Daimler and Scania) are also testing autonomous trucks on the road. On private property autonomous driving transport vehicles are already in use for applications like farming and mining [BCH16]. The loading and unloading of goods can also be done automatically [Flä16].

Last mile delivery

A great potential for autonomous transport vehicles is the delivery of products to customers in urban areas. Small, electric powered, autonomous robots can transport goods on the last mile cost-efficient [BCH16]. The company Starship Technologies developed such delivery bots (see figure 4.2). Their six wheeled robots can transport the content of two shopping bags. Using an app, the recipient can decide when to get the goods, follow the bots progress on the map and finally open it to get the products. Other companies like Amazon or Walmart try to use autonomous unmanned aerial vehicles (UAVs) for this purpose.

⁷Photo from the Starship Technologies web page page (<https://www.starship.xyz>)



Figure 4.2: Autonomous transport robot from Starship Technologies ⁷

4.4 Others

The three already described ICTs are certainly the most important to improve food sharing processes and together with a coordination model (see chapter 5) they could be used to create a smart food sharing solution. Of course, there are other technologies that could be embedded to a food sharing process to further improve it.

The German Fraunhofer institute has sent out a press information⁸ that they have developed a sensor film that can be packed with fresh meat and fish and changes its colour when the food gets bad. Also in press information^{9,10} the TU Vienna claims that they have developed test strips to detect unwanted substances in food. All you need to scent out poison, allergen and bacteria is a heating block and this special control strips. Furthermore, the real identities of the users could be checked. There are log in methods that can be used online and prove the real identity of people (e.g. the "Handysignatur"¹¹ in Austria)

4.5 Summary

This chapter analysed three ICTs that can be used to transform the food sharing process into a smart one. RFID allows to get product information from food easy. The necessary standards were already created, now it is up to food producers to equip their products

⁸<https://www.fraunhofer.de/de/presse/presseinformationen/2011/april/keine-chance-fuer-gammelfleisch.html>

⁹https://www.tuwien.ac.at/aktuelles/news_detail/article/8714/

¹⁰https://www.tuwien.ac.at/aktuelles/news_detail/article/9373/

¹¹<https://www.handy-signatur.at/hs2/>

with RFID tags. Smart fridges are already on the market and help their users to monitor their food stock. Additionally they should be endowed with RFID readers so that they can examine its content without manual intervention in future. Autonomous transport vehicles are already used in test operation. Further testing and legal changes are needed so that they can overtake the transport of shared food products. Further details about how these three technologies improve food sharing are in chapter 7. But first chapter 5 will analyse coordination models as further ICT to make the food sharing between households more convenient.

Coordination models

Food sharing can be seen as coordination problem with time and location constraints. People and companies with too much food must hand it over to other people or organisations who need it, before it goes off. When the food givers are food producers or food retailers that have bigger amounts of food products and the takers are charities that have use for various forms of food and own logistics to pick up food, current food sharing initiatives already work out well. However, food sharing at household level is different. Lots of small and heterogeneous food offers and complex time constraints (people must be at home for the handover of the food products) increase the coordination effort. Current food sharing initiatives have too simplistic models to deal with that and therefore too few food is exchanged (see subsection 3.1.4).

Coordination models help to deal with such complex coordination problems. They abstract the network communication and the developer can concentrate on coordination logic. Coordination models support the design and implementation of distributed and concurrent software that is robust, flexible and easy to adapt and extend in case of new requirements [KCS15].

Therefore, this thesis uses a coordination model to design and implement a food sharing process between households (see chapter 6). This chapter will deal with coordination models in general. First, it briefly describes and compares different coordination models and explains why the Peer Model is used in this thesis (see section 5.1). Then section 5.2 will describe the Peer Model in more detail (see section 5.2). Finally, section 5.3 will sum up the chapter.

5.1 Selection of a suitable coordination model

There are several coordination models that can be used to deal with coordinating food sharing between households. The sense of this section is to find the best coordination

model for this purpose. First, selection criteria are defined (see subsection 5.1.1) and then popular coordination models are compared according to them and the most suitable one is selected (see subsection 5.1.2).

5.1.1 Selection criteria

The coordination model used for modelling the coordination problem inherent to food sharing between households must meet the following criteria that were taken from [Küh16], [Küh17] and [KCJ⁺13]:

1. **Readable models:**

As already denoted, food sharing between households is a complex coordination problem. The used coordination model therefore must support these problems and keep the resulting model clear and readable so that all stakeholders can understand it.

2. **Graphical representation:**

It would make sense to discuss the resulting model with food sharing experts (e.g. representatives of food sharing initiatives). Since they are no computer scientists a graphical representation is indispensable.

3. **Runtime support for many platforms:**

In order that a model can be used to coordinate a food sharing solution it must be executed. Therefore, the coordination model should provide runtimes for various platforms where its models can be executed.

4. **Time constraints:**

As already mentioned, food sharing has to deal with complex time constraints (e.g. food is getting bad). The coordination model therefore must provide support for time constraints.

5. **Flow correlation:**

The processes for food sharing (e.g. the handover of a food product) can be seen as distributed work flow. In order that the concurrent flows do not affect each other in a undesirable way, correlation of flows must be supported by the coordination model.

6. **Exception handling:**

The work flows of food sharing can run into anomalous conditions (e.g. someone does not show up at the product handover). The coordination model must provide a suitable mechanism to deal with these exceptions.

7. **Transaction handling:**

Some actions of the food sharing processes must be carried out as distributed transaction (e.g. the reservation of a food product and its transport). To keep the

modelling as simple as possible the coordination model must provide support for distributed transactions.

5.1.2 Comparison and Selection

The most important coordination models are Petri Nets (see [Pet62]), Reo (see [Arb04]), the Actor Model (see [HBS73]) and the Peer Model (see [KCJ⁺13]). Kühn, Craß and Scherman in [KCS15], Kühn in [Küh16] and [Küh17] and Scherman in [Sch14] have already described the properties of those coordination models. Based on their work the next paragraphs find a suitable coordination model to deal with the food sharing coordination problem.

Petri Nets are well-founded and widely used to model concurrent processes. There exist a graphical notation and tools for modelling as well as for simulating and verifying activities. Petri Nets are universally applicable and its extension (Colored Petri Nets and Timed Petri Nets) provide support for time constraints. However, Petri Nets tend to become unreadable with increasing problem size. Furthermore, no support for correlating flows, handling errors, distributed transactions and no runtime environments are provided.

Reo is a data-driven coordination model that forces a clear separation of coordination logic from application logic. Same as in Petri Nets there are tools to model processes with its graphical notation and to verify them. Time constraints are supported as well. Unfortunately, it is too verbose and complex coordination problems are difficult to handle. It also does not support flow correlation, error handling, distributed transactions and has no runtime environment.

The Actor Model provides a mathematical model to deal with coordination problems. It supports distributed transactions and time constraints. Furthermore it has frameworks to execute the models on different platforms. These frameworks often provide additional features (e.g. flow correlation and error handling) that are not supported by the formal model. Unfortunately, the Actor Model does not have a graphical representation.

The Peer Model is the newest of these coordination models and was partially inspired by the others. It is well suited to deal with complex coordination problems, provides a graphical representation and runtime environments for different platforms. Furthermore, it supports time constraints, flow correlation and exceptions handling. A recent extension provides distributed transaction handling.

Table 5.1 compares the four most important coordination models with respect to the criteria defined above. A ✓ means that the formal definition of this coordination model sufficiently fulfils the corresponding criterion, a ✗ means it does not. ● means that the feature is not supported by the formal definition of the coordination model, but there are some frameworks for this coordination model that fulfil the criterion. As one can see, in contrast to the other three coordination models, the Peer Model satisfies all of the defined criteria. Therefore, it will be used to model the coordination parts of food sharing between households (see chapter 6). The next section (see section 5.2) will explain the Peer Model in more detail.

Coordination model \ Criterion	Petri Nets	Reo	Actor Model	Peer Model
Readable models	✗	✗	✓	✓
Graphical representation	✓	✓	✗	✓
Runtime support for many platforms	✗	✗	✓	✓
Time constraints	✓	✓	✓	✓
Flow correlation	✗	✗	●	✓
Exception handling	✗	✗	●	✓
Transaction handling	✗	✗	✓	✓

Table 5.1: Comparison of popular coordination models

5.2 Peer Model

The Peer Model was invented by Kühn as a novel programming model for modelling coordination between concurrent parts of a distributed software system. It is a data-driven approach and needs a tuple space-based middleware for data transfer among its participants [KCJ⁺13][KCJN14]. With the Peer Model Domain Specific Language (PM-DSL) the Peer Model provides a formal definition that allows automatic code generation for the runtime system of the Peer Model and therefore bridges the gap between design and implementation. Also, documentation (e.g. the graphical notation of the Peer Model) can be generated automatically [Ham15][KCH14]. The forced separation between coordination logic and application logic increases the readability and maintainability. Furthermore, it enables the reuse of configurable coordination components [KCS15][Sch14].

The Peer Model is independent of programming language and platform. Currently there exist runtime systems for Go, Java [Küh16], .Net [Rau14], ANSI C [Ham15][KCH14] and Android [Sch17][Til17]. In addition, there is a graphical monitoring tool that facilitates debugging for the Peer Model [Csu14].

The following subsections will describe the basic concepts of Peer Model (see subsection 5.2.1) and one of its extensions (see subsection 5.2.2) together with their graphical representation. As both, the Peer Model and the graphical notation, were refined over the time the latest version (see [Küh16] and [Küh17]) will be used.

5.2.1 Ground model

Using the, Peer Model coordination problems can be modelled in a convenient way. For this, its ground concepts are applied [Küh16]. These ground concepts are: Entry (data handled by the Peer Model), container (to store entries), peer (participants of the Peer Model), wiring (to describe the behaviour of a peer), link (to transport entries) and service (for application specific code) and will be explained in the next paragraphs.

Entry

All data that is transported in the Peer Model is encapsulated in entries. Entries are used to share information between processes and can represent among others messages, events, tasks, requests, acknowledgements and data. Each entry is structured in properties that have a name and a value. Besides the reserved system-defined coordination properties, arbitrary application-defined coordination properties can be added. For coordination in the Peer Model both system-defined and application-defined coordination properties can be used. Thereby the Peer Model is extensible.

The following listing contains the most important system-defined coordination properties and describes their meaning:

- **type:**
The **type** property stands for the coordination entry's coordination type and is the only mandatory property, because entries are selected on their type. The **type** property can be set arbitrary, only the name **exception** is reserved for exceptions.
- **time-to-live (ttl):**
Entries whose **ttl** is reached are ignored by the Peer Model and get eventually cleaned up. If the entry's **ttl_exc_dest** property was set to a peer, an entry of **type** exception is sent to this peer. The original entry is wrapped in the **entry** property of the exception entry and its type can be accessed via the **etype** property. The **ttl** allow a simple modelling of application timeouts.
- **time-to-start (tts):**
Entries whose **tts** is not reached are ignored by the Peer Model. This concept can be used to model timers as well as periodic repeating.
- **destination (dest):**
The **dest** property is used to specify an entry's destination. The entry is transported to the given peer's peer input container (PIC) by the Peer Model.
- **flow identifier (fid):**
The Peer Model supports the correlation of flows. If the **fid** property is set, the entry is treated separately from entries with incompatible **fids**.
- **data:**
With the **data** property application-specific data can be added to entries. The type of this data structures is set with the **data-type** property. Note that the **data-type** and the **type** property are independent of each other. The former indicates the data type of the application-specific information encapsulated in that entry and the later one is the type of entry used for coordination purposes.

As already described before, each type of information in the Peer Model is wrapped up in entries, what also applies to exceptions. When an exception occurs (e.g. a service fails,

an entry cannot be delivered or a **t**tl expires) an entry with coordination type **exception** is created. Exception entries are treated like any other entry in the Peer Model.

Container

Containers are used in the Peer Model to store entries. The containers are realized by extensible virtual shared memory (XVSM) (see [CKS09]), an extended and improved version of a tuple-space-based middleware. The Peer Model uses space containers that support transactions and blocking operations for a peer's PIC and peer output container (POC) as well as internal containers for wirings that do not support these features.

Peer

The participants of the Peer Models are called peers. Each peer has a unique name and two containers: The PIC for receiving entries and the POC for sending entries to other peers. Peers can read or take entries from and write new ones to their containers. The exact behaviour of a peer is defined by its internal wiring that can be activated by writing entries to the peers PIC. Furthermore, peers can have sub-peers that encapsulates logic only needed by them.

All peers of a site together form a so-called Peer Space. The Peer Space provides methods to add or delete peers and wirings. Furthermore each Peer Space contains an I/O peer that is responsible for transporting the entries between the peers. Entries that are transported on action links and whose **dest** property is set, are written to the I/O peers PIC automatically. The I/O peer delivers all entries of its PIC. If the **dest** values a local peer (i.e. in the same Peer Space) the entry is written to its PIC directly. When the **dest** property is set to a remote peer (i.e. the peer resides on another site) the I/O peer calls the `send` service to transmit the entry to this site. At the other Peer Space the `receive` service takes the entry and delivers it to the PIC of the appropriate peer.

Figure 5.1 shows the graphical notation of a peer. The low rectangle on the top contains the name of the peer, the slim grey rectangles on the left-hand and the right-hand side represent the PIC respectively the POC of the Peer. The rectangle in the centre is meant to define the peer's behaviour and can contain arbitrary many wirings and sub-peers. Peers are identified uniquely by the combination of the Peer Space and the peer name.

Link

Links are intended to transport entries between the containers of a peer as well as between the containers and services of a peer. Links are used by the wirings of a peer and a link is defined by the following parts:

- **source container:**

This attribute defines the source container for the link. It can be the PIC or the POC of a peer or its direct sub-peers as well as the internal container of one of the peer's wirings.

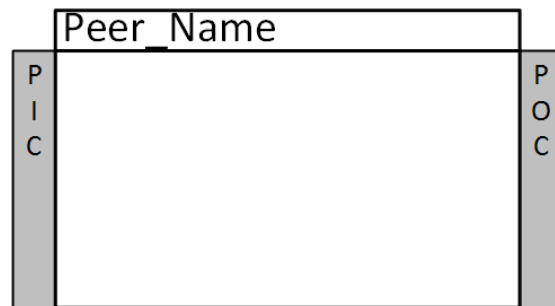


Figure 5.1: Graphical representation of a peer (without wirings and sub-peers)

- **target container:**

It defines the target container for the link. Identical to the source container it can be any of the peer's container.

- **operation:**

Operation can be one of the following seven values and defines what happens with the transported entries in their source respective their target containers: **Copy** (also called **read**) reads the entries at the source and copies them to the target, **take** (also called **move**) does the same, but deletes the entries at the source. **Create** generates new entries that do not exist in the source and writes them to the target, **delete** in contrast deletes them from the source and do not insert them in the target. With **test** entries are read at the source but not transported to the target. **Noop** does neither access source nor target and **call** is used to execute services.

- **query (optional):**

The query specifies which elements are selected in the source container and consists of three parts: type, count and selectors. A link transports solely entries of one coordination type that is defined by type. Count defines how many entries are selected by the query. It can be a precise number, a range or the keywords **ALL** and **NONE**. The default value is one. Selectors is a sequence of selectors connected by AND and OR. Selectors are based on the container selectors of XVSM and allow to define in which order (e.g. FIFO or random) and what entries (e.g. by SQL-like queries) are selected. If the query is not empty, the link is not executed before it is fulfilled.

- **expressions (optional):**

Links have access to the properties of the entries they transport and can manipulate them. Furthermore, they can create, read and write local variables that exist only in the context of their wiring and access system variables (e.g. `THIS_PEER` for the name of the current peer). Also functions (e.g. `fid()` to create a new **fid**) can be called. All these operations can be done in the expression part of the link.

- **properties (optional):**

A link can be configured using optional properties. The most important are: **tts** (the time when the link starts operating), **ttd** (how long an instance of that link is executed), **dest** (all entries of this link are transported to the given peer; only for action links), **flow** (if true (default value) the link only transports entries with the same or with no **fid**) and **mandatory** (if true the link must succeed so that the wiring does not fail).

Links can have the following four types: guard link, action link as well as input and output link of services. Guard links provide entries for the wirings and therefore their target is always the internal container (called entry collection (EC)) of the wiring. On the other hand action links transport the entries away from the wirings so their source is always the EC of the wiring. Service input and output links transport entries between a wiring and its services.

Figure 5.2 shows the graphical representation of a link. It is an arrow where the tail is connected to the source container and the head to the target container (the containers are not shown here). Above the line of the arrow there is the operation written in bold characters followed by the type, the count in square brackets, the selectors in double square brackets and the expressions in angle brackets. The space below the line is reserved to set the link's properties.

The input and output links of services can be omitted in the graphical notation. Then the service must be connected directly with the wiring and the service has access to the entire EC of the wiring and can read/add/delete/change all entries.

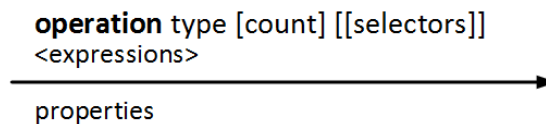


Figure 5.2: Graphical representation of a link (without source and target container)

Wiring

Wirings are the active part of the Peer Model and model concurrent processes. They are activated by entries in the space containers of a peer or its sub-peers. The wirings can fetch entries from these containers, process and change them, delete some of them or create new ones. These resulting entries are written to space containers and can activate further wirings. Basically, the execution of a wiring consists of three consecutive steps:

1. **Guards:**

Guards transport entries from space containers to the wiring's EC and are realised by guard links. A wiring can have arbitrary many guards, but at least one. All

guards of a wiring are numbered with a running value and executed in this order. Generally, the **mandatory** property of guard links is set to true. That means if a guard link fails (i.e. its query cannot be fulfilled) the execution of the wiring is halted until the guard link succeeds or the execution of the wiring fails (depending on the **rollback_on_failure** property of the wiring as well as the **ttd** of the wiring and the guard link).

2. Service execution (optional):

If a wiring needs to execute business logic it does it by calling one or more services. Like the guards all services have a running value and the execution order is defined by this number. Via input and output links a service can modify the content of its wiring's EC.

3. Actions (optional):

Actions use action links to transport entries from the wiring's EC to the space containers. Since the action part of a wiring is optional a wiring can have any amount (including zero) of actions. Same as in guards a running value determines their execution order. Usually the **mandatory** property of action links is set to false. Therefore, if its query cannot be fulfilled, this action link is skipped and transports no entries.

At runtime, each wiring of an instance of a peer can have several instances (bounded above by the wirings property **max_threads**). The **tts** property can be used to set the time until a new instance gets operative. The **ttd** property can be used to bound the operation time of a wiring instance. All instances of a wiring run in parallel and perform their tasks concurrently.

An instance of a wiring is executed in one single transaction (called wiring transaction (WTX)). For the execution, a wiring considers only entries with the same **fid** or no **fid**. The first entry that is fetched and has a **fid** defined, sets the **fid** for this instance of the wiring. Links whose **flow** property is set to false are excluded from this rule.

Figure 5.3 shows the graphical representation of a wiring with its guards, actions and a service. The white rectangle in the centre stands for the wiring and the wiring's name is denoted there. If the wiring sets some properties, they must follow the name after a colon in square brackets. The grey small rectangles at the left side represent the guards and the number inside their running value. Guard links (not shown in this example) are connected to them. Similar to this the small yellow boxes on the right-hand side represent the actions. The light blue box at the top of the wiring stands for a service and its name is written inside. In this example the service links are omitted and the service is directly connected to the wiring.

Service

Services are the only part of the Peer Model that contains application specific logic. However, the logic of a service is not modelled in the Peer Model. Through their input

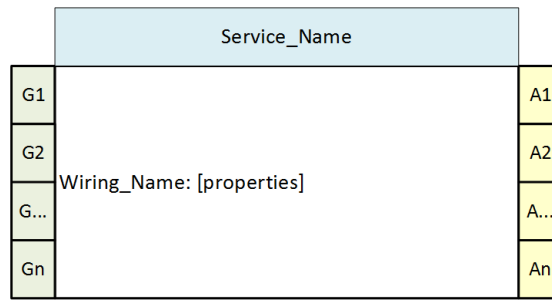


Figure 5.3: Graphical representation of a wiring (guard and action links are not shown)

links and output links respective their service links, services have access on the entries in the wiring's EC. Services can there create, read, update and delete entries.

Since application specific code is not modelled in the Peer Model, the graphical representation of services is very simple. As already shown in figure 5.3 a service is represented by plain light blue rectangle, labelled with its name.

5.2.2 FWTX

Extensive coordination problems often have dependencies between concurrent coordination steps and need to conduct distributed flows in transactions. The original version of the Peer Model does not support these requirements very well. All phases of a distributed transaction as well as the clean-up of outdated entries to manage the transactions must be modelled explicit. Furthermore, associated consecutive steps of a workflow must be split into several wirings. This leads to unneeded complicated models with too much wirings and links.

To overcome this problem Kühn provided in her conference paper [Küh17] an extension to the Peer Model called flexible wiring transactions (FWTX). FWTX is inspired by the Flex transaction model (see [BEK93] and [ELLR90]) where an early commit of sub-transactions is allowed. In case of an abort of its parent transaction a compensate action is called to undo the foregoing commit. Since really undoing this commit is not possible it is a semantic compensation. To make use of this concept in the Peer Model the WTX or wirings are extended to FWTX. FWTX provides the new link operation **wiring** to call a special form of wirings named passive wirings that take over the part of sub-transactions. Furthermore, new properties for wirings allow to define compensate, on-commit, on-top-commit, on-start and on-abort actions.

Passive wiring

Passive wirings are very similar to the ordinary wirings of the Peer Model. The greatest difference is that for an ordinary wiring there are always instances active that try to fulfil the guards and that a passive wiring creates a new active instance when it is called as sub-transaction by a guard link or as compensate, on-commit, on-top-commit, on-start

or on-abort action of a wiring. Furthermore, since passive wirings are called directly, parameters can be passed to them.

The graphical representation of a passive wiring is not much different from an ordinary wiring. The rectangle with the wiring's name has a dashed line to differ it from ordinary wirings. Furthermore, between the wirings name and the properties, the parameters are written in brackets.

Link operation wiring

One possibility to call passive wirings is via a guard. Therefore, the new link operation **wiring** was introduced. A link with this operation does not need a query to fetch entries, instead the called passive wiring is denoted by its name. The link is successful if the execution of the passive wiring was successful. To pass local variables to the passive wirings, the expression part of the link is used. The n-th parameter of the wiring is set addressed by \$n.

New properties for wirings

Using FWTX `compensate`, `on-commit`, `on-top-commit`, `on-start` and `on-abort` actions for wiring can be defined. These actions are set by wiring's properties with the same name and refer to passive wirings. The `on-commit` action is called automatically when the wiring commits, `on-top-commit` when the top-level wiring commits. `On-start` is called before the wiring is executed and `on-abort` when the wiring aborts. `Compensate` is executed when the wiring has already committed and it's calling wiring fails. Furthermore, the Boolean property `cascade` defines if the compensation is cascading or not.

5.3 Summary

This chapter has explained why a coordination model is necessary to deal with such a complex coordination problem like food sharing at household level in a reasonable way. In the following it shortly summarized properties of several popular coordination models and explained why this thesis will use the Peer Model to model the food sharing process. Then the Peer Model was described in more detail. At first its ground model and then one of its recent extensions: FWTX. In the next chapter (see chapter 6) this thesis uses the Peer Model to model the coordination inherent to food sharing between households.

S-FOSM: a Smart FOod Sharing Model

The previous chapter (see chapter 5) has already mentioned that food sharing between households is a complex, distributed coordination problem. Furthermore, it explained why a coordination model should be used to deal with this problem and why in this thesis the Peer Model will be used. This chapter translates this into action and uses the Peer Model to develop the S-FOSM, a model that provides a solution for the coordination parts of food sharing between households.

The next section (see section 6.1) describes how the S-FOSM is modelled with the Peer Model. Then the S-FOSM is implemented for the Java runtime of the Peer Model (see section 6.2). Afterwards a simulation is used to prove that the S-FOSM is indeed a solution for the food sharing problem (see section 6.3). Finally, section 6.4 will sum up this chapter.

6.1 Modelling the S-FOSM

In this section the S-FOSM is modelled with the graphical notation of the Peer Model. It is the first formal representation of a solution for the food sharing problem.

Unfortunately, up to now there is no specific modelling tool for the Peer Model, so this work used the general-purpose diagramming tool Microsoft Visio¹ and followed the latest version of the graphical notation as described in [Küh16] and [Küh17].

A prerequisite for food sharing is that there are people that give off and people that take food. The idea of the S-FOSM is that participants that have a food product they do not need, create a food offer (modelled by the `myFoodOffer` entry type) and people that

¹see <https://products.office.com/en/visio/flowchart-software?tab=tabs-1>

want edibles search for this food offers (modelled by the `foodSearch` entry type). For each new food search at the beginning is checked if there are some food offers that match the search. Then for each of the matching food offers is tested, if it can be transported from the offering to the searching food sharer. Finally, the searching partner selects one of the matching food offers whereby the transport is possible and the edible is shifted from one participant to the other.

Each participant of the S-FOSM has to run an instance of the `FoodSharingPeer` and can offer food products, search for food products or do both. To add a food offer, a `myFoodOffer` entry type with the properties `productName`, `pickupOptions`, `productDetails` and `foodOfferID`, has to be sent to the `FoodSharingPeer`. A `foodSearch` entry type with the properties `status="new"`, `searchTerm` and `deliveryOptions` must be sent to the `FoodSharingPeers` PIC in order to a search for food.

Figure 6.1 shows the `FoodSharingPeer` from a high level point of view (the wirings are omitted). As one can see, it has two sub-peers, the `FoodOfferManagerPeer` and the `TransportManagerPeer`. The following subsection (see subsection 6.1.1) describes the sub-peers, the 21 wirings of the `FoodSharingPeer` are explained in detail afterwards (see subsection 6.1.2).

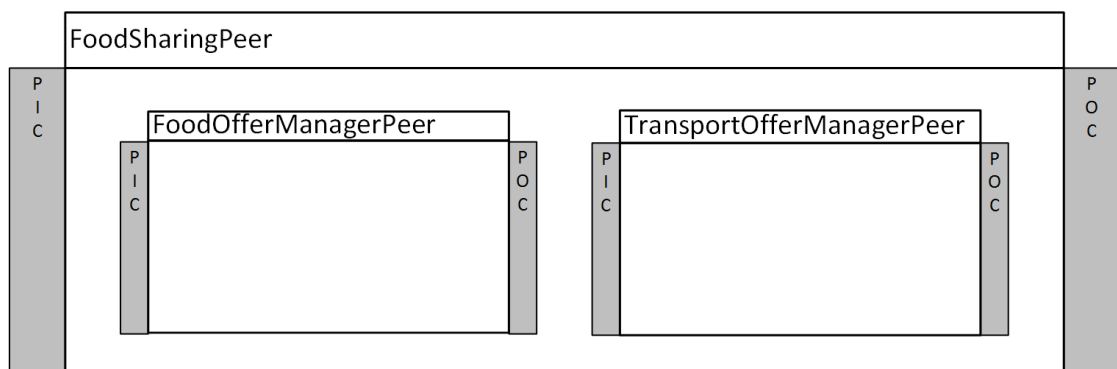


Figure 6.1: The `FoodSharingPeer` (from a high level point of view)

6.1.1 Sub-peers

The sub-peers of the `FoodSharingPeer` are used to abstract sub-problems of the food sharing problem and their behaviour (e.g. their wirings) is not modelled here. Finding optimal solutions for these sub-problems is outside the scope of this work (e.g. the `TransportManagerPeer` has to deal with the NP-hard scheduling problem). The next paragraphs describe what solutions for the sub-problems must do in order to solve them, but not how the sub-problems must be solved. However, for the simulation of the S-FOSM mock-up implementations for the sub-peers will be used (see subsection 6.3.2).

FoodOfferManagerPeer

The FoodOfferManagerPeer manages the food offers for the FoodSharingPeer.

In the PIC of its FoodOfferManagerPeer the FoodSharingPeer can find entries of type foodOffer representing the current surplus food products of all other food sharers it cooperates with. Moreover, the PIC of the foodOfferManagerPeer contains entries of type myFoodOffer representing the current surplus food of its FoodSharingPeer. The FoodSharingPeer can read, add and delete these entries of type myFoodOffer and the FoodOfferManagerPeers are responsible that they have an entry of foodOffer for each entry of myFoodOffer for all cooperating partners.

In addition, the FoodOfferManagerPeer is responsible for keeping an actual list of all other food sharers it cooperates with. Based on some criteria (e.g. the needed resources for the transport of products between them) it must decide if it cooperates with another food sharer or not.

TransportOfferManagerPeer

The TransportManagerPeer manages the transport of food products between cooperating FoodSharingPeers.

It receives entries of type transportFoodOfferRequest in the PIC and has to replace them with entries of transportFoodOfferResponse. The application-defined coordination properties pickupOptions (several options when and where the food can be picked up), productInformation (product information like size, weight and expiration date), deliveryOptions (several options when and where the food can be delivered), offeringPartner and FoodOfferID of the transportFoodOfferRequest must be set that the TransportManagerPeer can decide whether this transport would be possible or not and create the transportFoodOfferResponse. The application-defined property isFeasible of the transportFoodOfferResponse must be true or false and if it is true the properties selectedPickupOption (when and where the food is picked up), productInformation, selectedDeliveryOption (when and where the food is delivered), offeringPartner and FoodOfferID must be set as well.

When a reserveTransportRequest entry (properties: selectedPickupOption, productInformation, selectedDeliveryOption, offeringPartner and FoodOfferID) reaches the TransportManagerPeers PIC it has to take it, check if this transport is still possible and if so write a reserveTransportResponse back. This indicates that the TransportManagerPeer has reserved the resources for the transport. Then the offering partner has to add a foodRelease entry to the PIC of its TransportManagerPeer. When the pickup time of the selectedPickupOption is reached the foodRelease entry is deleted and when the delivery time of the selectedDeliveryTime is reached a foodDelivery entry added to the PIC of the TransportManagerPeer of the searching partner. In case a previous reserved transport is not needed anymore a cancelTransportReservation entry can be sent to the TransportManagerPeer and it tries to free the reserved resources for the transport.

6.1.2 Wirings

The FoodSharingPeer has to react to four main events: A new food product to share with other participants (denoted a myFoodOffer entry is in its PIC), a new search for food (denoted by a foodSearch entry in its PIC), the reservation of a previous offered product by another participant (its passive wiring reserveFoodWiring gets executed) and the delivery of a preliminary reserved edible (denoted by a foodDelivery entry type in the PIC of the TransportManagerPeer of the FoodSharingPeer). The following paragraphs describe the behaviour of the FoodSharingPeer by explaining its wirings.

The SendMyFoodOfferToManagerWiring (see Figure 6.2) gets executed when there are unreserved myFoodOffers (property reserved is not true) in the PIC of the FoodSharingPeer. This is the case when the FoodSharingPeer has a new food product to offer. It takes all unreserved myFoodOffers (guard G1) and sends them to the PIC of the FoodOfferManager sub-peer (action A1).



Figure 6.2: SendMyFoodOfferToManagerWiring

When the MyFoodOfferExpiredWiring (see Figure 6.3) gets executed the expiration date of a myFoodOffer was reached. The wiring takes an exception of **etype** myFoodOffer (guard G1) before the MyFoodOfferExpiredService is called to handle this problem (e.g. inform the user).

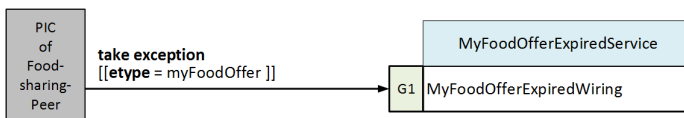


Figure 6.3: MyFoodOfferExpiredWiring

If the FoodSharingPeer receives a new foodSearch the GetMatchingFoodOffersWiring (see Figure 6.4) is executed to get all suitable food offers. It takes the foodSearch (guard G1) and reads all matching foodOffers from the FoodOfferManagerPeer (guard G2). Then it updates the status to gotFoodOffers, sets the numofTransportRequestsSent to zero, the numberOfMatchingFoodOffers to the correct value and sets the **fid** (action A1). Furthermore, the **fid** of all foodOffers is set before they are written to the FoodSharingPeers PIC.

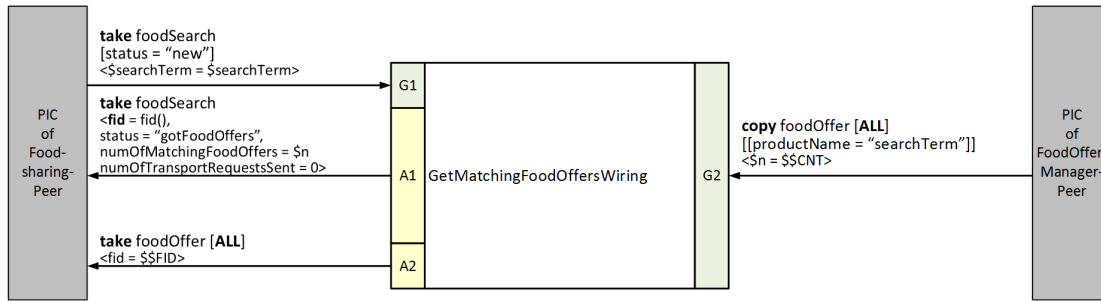


Figure 6.4: GetMatchingFoodOffersWiring

The `NoMatchingFoodOffersWiring` (see Figure 6.5) is only executed if there were no matching `foodOffers` for a `foodSearch`. It takes the `foodSearch` (guard G1) and executes the `NoMatchingFoodOffersService` to handle this case (e.g. inform the user) and possibly start a new food search (action A1).

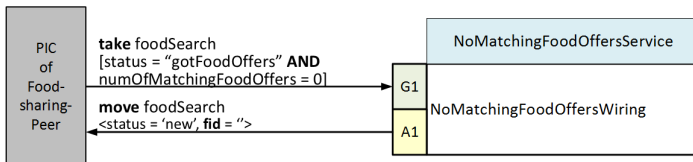


Figure 6.5: NoMatchingFoodOffersWiring

The purpose of the `RequestTransportFoodOfferWiring` (see Figure 6.6) is to create for each `foodOffer` a `transportFoodOfferRequest` and send it to the `TransportManagerPeer`. It takes the `foodSearch` (guard G1) and one `foodOffer` (guard G2) and creates a `transportFoodOfferRequest` and sends it to the `TransportManagerPeer` (action A2). Additionally, the `numOfTransportRequestsSent` is increased by one and the `foodSearch` written to the `FoodSharingPeers` PIC (action A1).



Figure 6.6: RequestTransportForFoodOfferWiring

After all `transportFoodOfferRequests` are sent the `UpdateStatusWiring` (see Figure 6.7) is executed to update the status of the `foodSearch`. The `foodSearch` is taken

(guard G1) and its status is set to `sentAllTransportRequests` before it's written back to the `FoodSharerPeers` PIC (action A1).

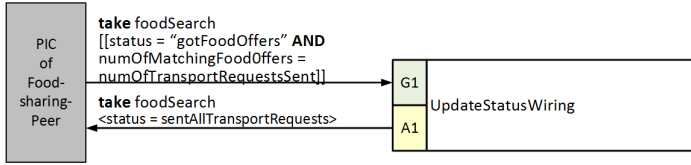


Figure 6.7: UpdateStatusWiring

When the `TransportManagerPeer` has created a `transportFoodOfferResponse` for each `transportFoodOfferRequest` and at least one of them is satisfiable the `SelectFoodWiring` (see Figure 6.8) is executed to select the best `foodOffer` that can be transported from the offering partner to the searching participant. The wiring takes the `foodSearch` with status `sentAllTransportRequests` from the `FoodSharingPeers` PIC (guard G1), takes all feasible (guard G2) and deletes all unfeasible `transportFoodOfferResponses` from the `TransportManagerPeers` PIC (guard G3). Moreover, it is only executed if there is at least one feasible `transportFoodOfferResponse` and one `transportFoodOfferResponse` was created for each `transportFoodOfferRequest` (guard G4). Then the `SelectFoodService` is executed to select the best `foodOffer` according to some algorithm. The selection is stored in the `foodSearch` and then it is written back to the `FoodSharingPeers` PIC.

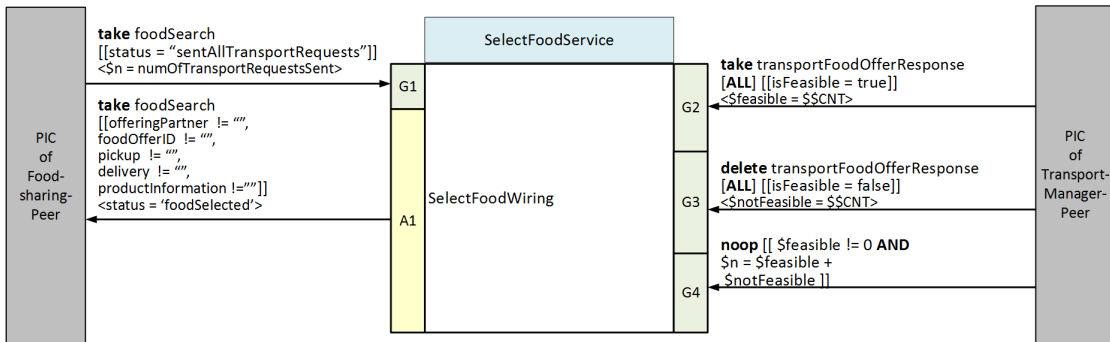


Figure 6.8: SelectFoodWiring

The `NoFeasibleTransportWiring` (see Figure 6.9) is only executed when the `TransportManagerPeer` has created all `transportFoodOfferResponses` and none of them is feasible. It takes the `foodSearch` with status `sentAllTransportRequests` (guard G1) and deletes all the `transportFoodOfferResponses` (guard G2). Then the `NoFeasibleTransportService` is executed to handle this case (e.g. inform the user). Furthermore, a new `FoodSearch` could be started (action A1).

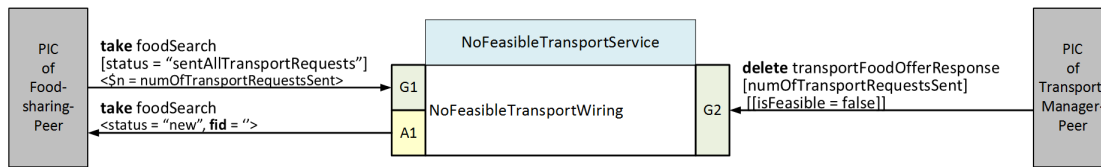


Figure 6.9: NoFeasibleTransportWiring

Whenever the foodOffer is selected the `ReserveFoodAndTransportWiring` (see Figure 6.10) is performed to reserve the foodOffer and its transport. The wiring takes the foodSearch with status `foodSelected` (guard G1) and calls the `reserveFood` wiring of the sharing partner to reserve the food (guard G2) and its own `reserveTransport` wiring to reserve the transport of the food. Both reservations must be completed within a specified time, otherwise the `ReservationTimeoutWiring` is executed as on-abort action.

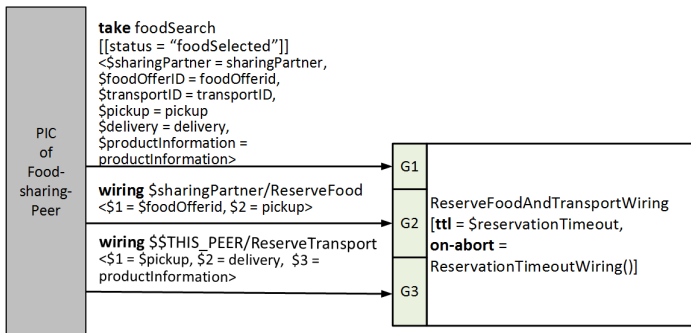


Figure 6.10: ReserveFoodAndTransportWiring

If the reservation of a foodOffer or its transport times out, the `ReservationTimeoutWiring` (see Figure 6.11) is activated by the `ReserveFoodAndTransportWiring`. Since it has no guards the `ReservationTimeoutService` is executed immediately to handle this problem (e.g. inform the user) and potentially starts a new foodSearch (action A1).



Figure 6.11: ReservationTimeoutWiring

The passive wiring `ReserveFoodWiring` (see Figure 6.12) reserves a foodOffer for a searching partner. Therefore, the `myFoodOffer` with fitting offerid is taken from the `FoodOfferManagers` PIC (guard G1) and written to the `FoodSharingPeers` PIC (action A1). The wiring has defined the `CancelFoodReservationWiring` wiring as compensate and the `PrepareFoodReleaseWiring` wiring as on-top-commit action.



Figure 6.12: ReserveFoodWiring

To revoke the reservation of a foodOffer the passive wiring CancelFoodReservationWiring (see Figure 6.13) is needed. It takes the myFoodOffer from the FoodSharingPeer (guard G1) and writes it to the FoodOfferManagerPeers PIC.

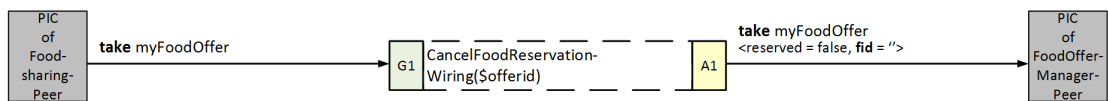


Figure 6.13: CancelFoodReservationWiring

When the wiring PrepareFoodReleaseWiring (see Figure 6.14) is activate, the reservation of the foodOffer is committed and the food product is prepared for its release. The wiring takes the myFoodOffer (guard G1) and writes back a foodRelease (action A1). The **tts** of foodRelease is set to pickup.minTime that the food is not released too early and the **tvl** of the foodRelease is set to pickup.maxTime that the FoodSharingPeer recognises when the foodRelease is not picked up by the FoodOfferManager in time.

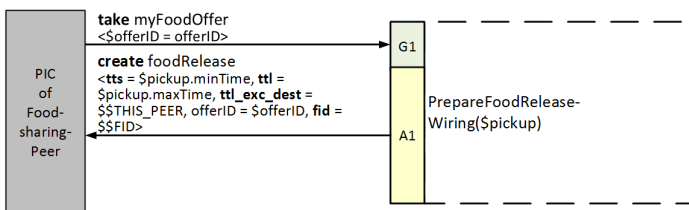


Figure 6.14: PrepareFoodReleaseWiring

The ReleaseFoodWiring (see Figure 6.15) is responsible to release a food product to the TransportManagerPeer. Therefore, the wiring takes the foodRelease (guard G1) and writes it to the TransportManagerPeers PIC.

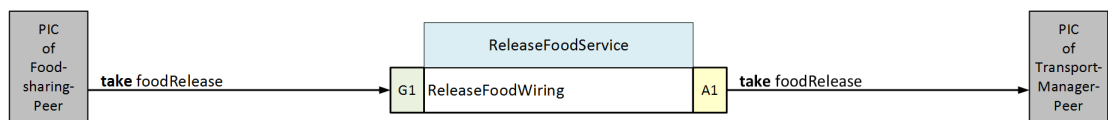


Figure 6.15: ReleaseFoodWiring

In case that the `foodRelease` is not picked up in time by the `TransportManagerPeer` the `FoodNotPickedUpWiring` (see Figure 6.16) is called. It takes the `foodRelease` within the exception (guard `G1`) and executes the `FoodNotPickedUpService` to deal with this condition (e.g. inform the user). The service can create a `myFoodOffer` that is written to the `FoodOfferManagers PIC` (action `A1`).

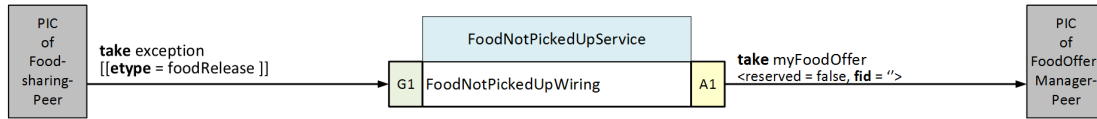


Figure 6.16: FoodNotPickedUpWiring

Using the `ReserveTransportWiring` (see Figure 6.17) the transport of a food product can be reserved. It takes a `transportReservationRes` from the `TransportManagerPeer` (guard `G1`) and writes it to the `FoodSharingPeer` (action `A1`). Furthermore, the `GetTransportReservationWiring` is defined as `on-start`, the `CancelTransportReservationWiring` as `compensate` and the `PrepareForFoodReceivingWiring` wiring as `on-top-commit` action.

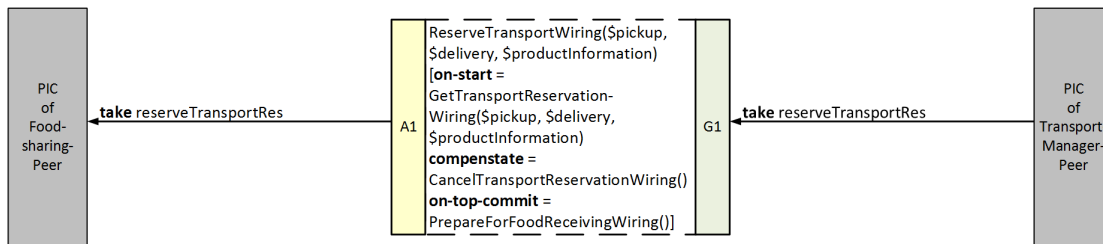


Figure 6.17: ReserveTransportWiring

With the passive `GetTransportReservationWiring` (see Figure 6.18) the transport of a food product is prepared. The wiring sends a `reservateTransportReq` to the `TransportManagerPeer` (action `A1`).

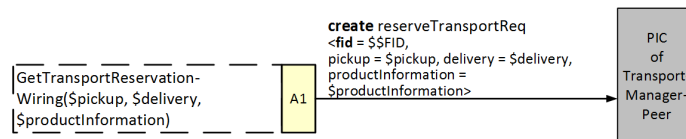


Figure 6.18: GetTransportReservationWiring

By calling the passive `CancelTransportReservationWiring` (see Figure 6.19) a transport reservation can be revoked. Therefore, the wiring takes the `reservateTransportRes` from the `FoodSharerPeer` (guard `G1`) and writes a `cancelTransportReservation` in the `TransportManagerPeers PIC`.



Figure 6.19: CancelTransportReservationWiring

When the transport of the food is committed the `PrepareForFoodReceivingWiring` (see Figure 6.20) wiring is executed to prepare the `FoodSharingPeer` for receiving the food. It takes the `reservateTransportRes` (guard G1) and writes back a `foodReceiving` with set `tts` and `tvl` that marks the time of the delivery.

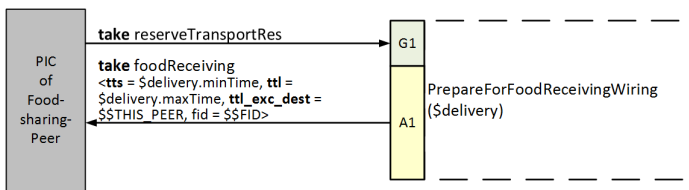


Figure 6.20: PrepareForFoodReceivingWiring

The `FoodReceivingWiring` (see Figure 6.21) is activated when the reserved food reaches the searching Partner. The wiring takes the `foodDelivery` (guard G1) and the `foodReceiving` (guard G2).

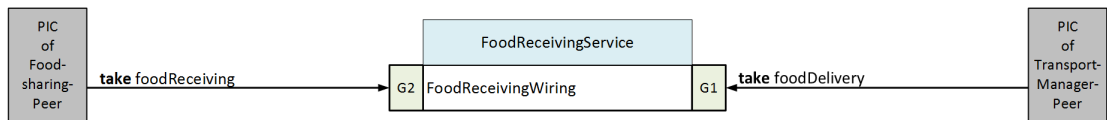


Figure 6.21: FoodReceivingWiring

If a requested food is not delivered in time the `FoodNotReceivedWiring` (see Figure 6.22) is activated. It takes the exception (guard G1) and executes the `FoodNotReceivedService` to handle this situation (e.g. inform the user). The service could also create a new `foodSearch` which is written to the `FoodSharerPeers` PIC.

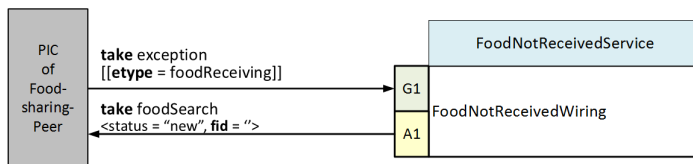


Figure 6.22: FoodNotReceivedWiring

6.2 Implementation of the S-FOSM

In the previous section (see section 6.1) the S-FOSM was presented with the graphical notation of the Peer Model. In order to execute the S-FOSM, source code for a specific runtime environment of the Peer Model has to be created. In this thesis the Java runtime of the Peer Model is used, because it supports the most features. This section explains the translation of the graphical model of the food sharing problem to source code for the Java runtime of the Peer Model in two steps: At first features that are not supported by the Java runtime of the Peer Model must be replaced (see subsection 6.2.1) and then the peers and wirings of the S-FOSM are translated to Java source code (see subsection 6.2.2).

6.2.1 Adaption of the S-FOSM

Both, the graphical representation and the formal definition of the Peer Model provide features that are not implemented in the Java runtime yet. To represent the food sharing problem this work used two features of the Peer Model that are currently not supported by the Java runtime: Variable assignments at wirings and FWTX. The following paragraphs describe how the S-FOSM is adapted in order to be executable with the Java runtime.

Variable assignments at wirings

The variable assignments at the wirings are removed and their behaviour is overtaken by the wiring's service. As the process is similar for all wirings it will be explained only for one of them, the `GetMatchingFoodOffersWiring`.

In the original S-FOSM with variable assignments at wirings (see figure 6.4) the `GetMatchingFoodOffersWiring` at first takes a `foodSearch` entry with status `new` and saves its `searchTerm` property to a local variable `$searchTerm` (guard G1). Then it copies all `foodOffers` whose `productName` match the `$searchTerm` and saves the number of copied `foodOffers` to a local variable `$n` (guard G1). As there is no service directly after the guard links the actions are performed. Action A1 sets the `fid` to a new generated one, the status to `gotFoodOffers`, the `numOfMatchingFoodOffers` to `$n` and the `numOfTransportRequestsSent` to 0 and returns the `foodSearch`. Finally, it sets the `fid` of all `foodOffers` and returns them (action A2).

In the adapted S-FOSM for the Java runtime of the Peer Model (see figure 6.23) the `GetMatchingFoodOffersWiring` takes a new `foodSearch` entry (guard G1) and all `foodOffers` (guard G2). Then the `GetMatchingFoodOffersService` is called. It must remove all `foodOffers` whose `productName` does not contain the `searchTerm` of the `foodSearch` and set the `fid` of the remaining ones. Furthermore, the `fid`, the status, the `numOfMatchingFoodOffers` and the `numOfTransportRequestsSent` of the `foodSearch` are set. Then the wiring returns the `foodSearch` (action A1) and all `foodOffers` (action A2).

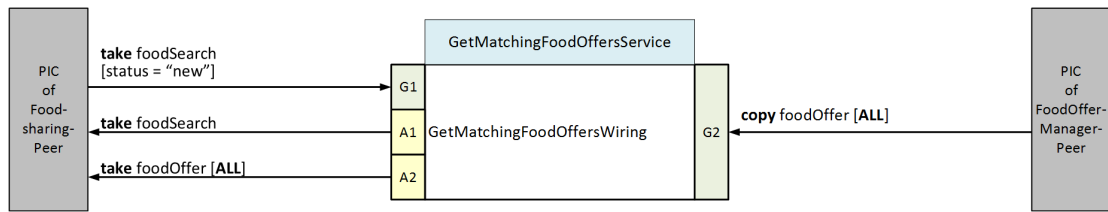


Figure 6.23: GetMatchingFoodOffersWiring (adapted S-FOSM)

FWTX

In the original S-FOSM FWTX was used to reserve the food and its transport. To adapt it for the Java runtime, passive wirings are replaced by regular ones, the call of a passive wiring is replaced by sending entries to the corresponding regular wirings and the implicit activation of passive wirings by actions (e.g. on-top-commit and compensate) has to be done explicit by sending entries to regular wirings.

The passive wirings `ReservationTimeoutWiring` (see Figure 6.11), `ReserveFoodWiring` (see Figure 6.12), `CancelFoodReservationWiring` (see Figure 6.13), `PrepareFoodReleaseWiring` (see Figure 6.14) and `PrepareForFoodReceivingWiring` (see Figure 6.20) of the original S-FOSM are replaced by regular wirings with the same name but different internal logic in the adapted S-FOSM (see Figure 6.26, Figure 6.29, Figure 6.30, Figure 6.31 and Figure 6.32). The adapted S-FOSM needs three new wirings, the `CommitReservationsWiring` (see Figure 6.25), the `CleanUpFoodReservationWiring` (see Figure 6.27) and the `CleanUpTransportReservationWiring` (see Figure 6.28) to control the commitment respectively the cancellation of reservations. Furthermore, in the adapted S-FOSM three wirings, the `ReserveTransportWiring` (see Figure 6.17), the `GetTransportReservationWiring` (see Figure 6.18) and the `CancelTransportReservationWiring` (see Figure 6.19), are not needed anymore since its wirings are directly communicating with the `TransportManagerPeer`.

In the original S-FOSM the reservation can be summarised like this: The `ReserveFoodAndTransportWiring` (see Figure 6.10) activates the `ReserveFoodWiring` (see Figure 6.12) and the `ReserveTransportWiring` (see Figure 6.17) to reserve the food and its transport. Both of these passive wirings try to perform their reservation within a given timeout. If one of them fails their compensate wirings are called to revoke possible performed reservations, when both of them are successful their on-top-commit wirings are activated to commit the reservations.

In the adapted S-FOSM the reservation process works like this: The `ReserveFoodAndTransportWiring` (see Figure 6.24) sends a `reserveFoodOfferRes` to activate the `ReserveFoodWiring` (see Figure 6.29) and a `reserveTransportRes` to the `TransportManagerPeer` to reserve the transport. If both reservations were successful (denoted by a `reserveFoodOfferRes` in the `FoodSharerPeers` PIC and a `reserveTransportFoodOfferRes` in the `FoodOfferManagerPeers` PIC) the `CommitReservationsWiring` (see Figure 6.25) is executed and activates the `PrepareFoodReleaseWiring` (see Figure 6.31) by

sending a `commitFoodReservation` and the `PrepareForFoodReceivingWiring` (see Figure 6.32) by sending a `commitTransportReservation` to commit the reservation. If one of the reservation has not been performed in the given time the `CleanUpFoodReservationWiring` (see Figure 6.27) and/or the `CleanUpTransportReservation` (see Figure 6.28) are called to revoke already performed reservations.

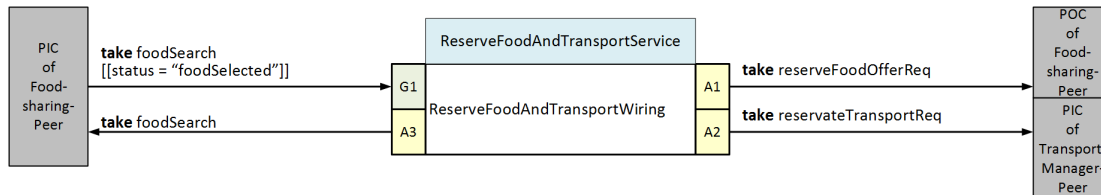


Figure 6.24: ReserveFoodAndTransportWiring (adapted reservation process)

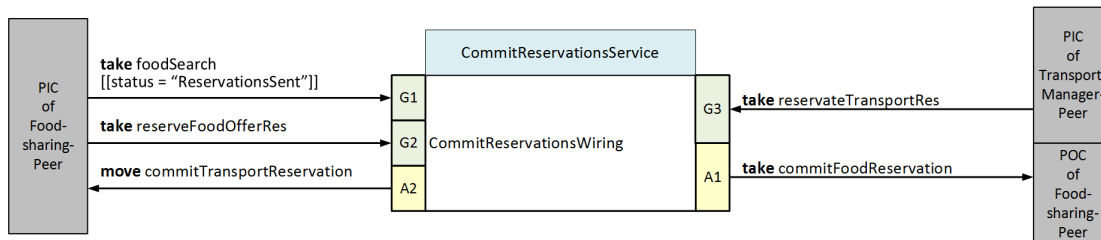


Figure 6.25: CommitReservationsWiring (adapted reservation process)

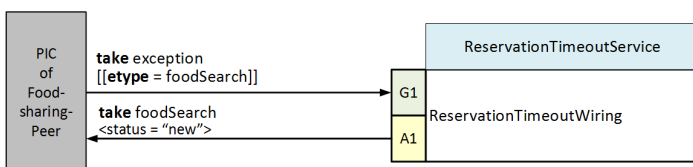


Figure 6.26: ReservationTimeoutWiring (adapted reservation process)

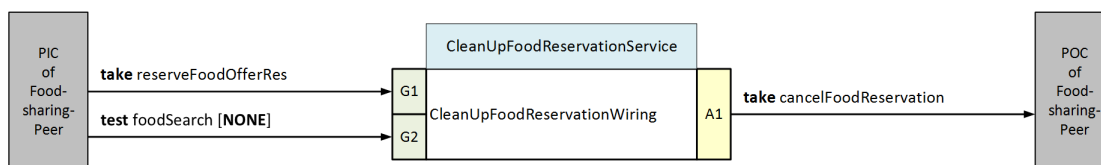


Figure 6.27: CleanUpFoodReservationWiring (adapted reservation process)

6. S-FOSM: A SMART FOOD SHARING MODEL

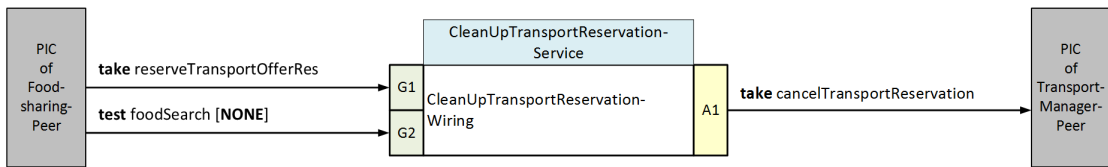


Figure 6.28: CleanUpTransportReservationWiring (adapted reservation process)

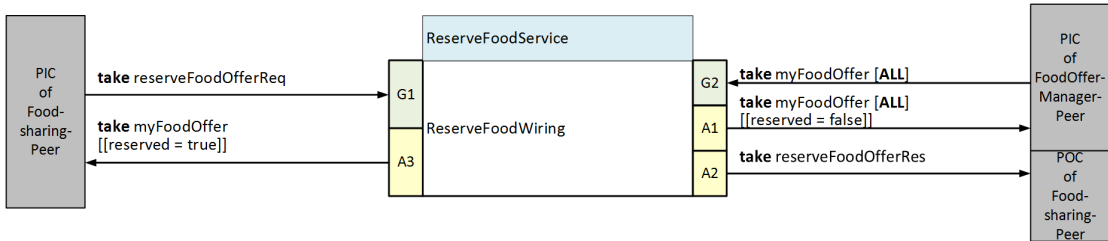


Figure 6.29: ReserveFoodWiring (adapted reservation process)

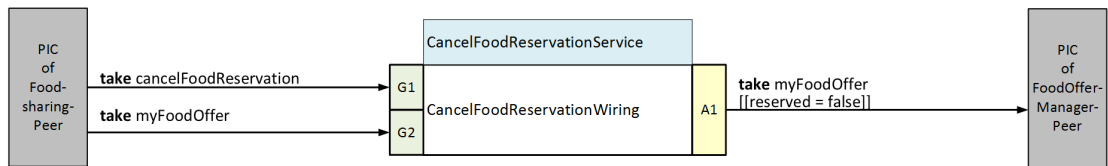


Figure 6.30: CancelFoodReservationWiring (adapted reservation process)

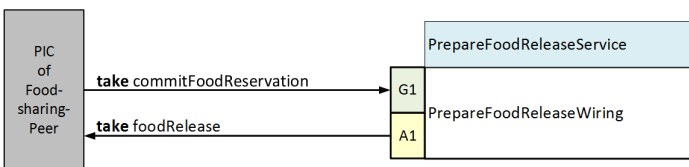


Figure 6.31: PrepareFoodReleaseWiring (adapted reservation process)

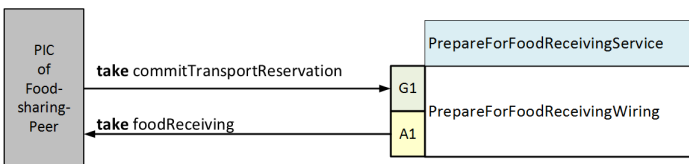


Figure 6.32: PrepareForFoodReceivingWiring (adapted reservation process)

6.2.2 Deriving the Java source code of the S-FOSM

To run the S-FOSM with the Java runtime of the Peer Model, Java code source for the Java runtime must be created. Unfortunately, up to now there is no code generator that does this job automatically, so the code has to be written by hand. The following paragraphs briefly describe how the Java code for the peers and wirings can be derived from the graphical model.

Peers

With the `PeerEntryBuilder` class a new peer can be created. Its constructor takes one `String`, the name of the peer, as parameter. The `dest` method sets the destination of the new peer which must be the `PSCAddress` of its superpeer. Using the `build` method the new peer is created. To give an example algorithm 6.1 shows the Java code for the `FoodSharingPeer`. As the `FoodSharingPeer` has no superpeer the `runtimepeer` is set as its superpeer.

Algorithm 6.1: Java code of the `FoodSharingPeer`

```
new PeerEntryBuilder("foodSharingPeer").
  dest(new Address(Address.RUNTIME_PEER_ADDRESS,
    Address.PSCAddress)).
  build();
```

Wirings

To implement a wiring the `WiringEntryBuilder` class can be used. The name of the wiring is set via the constructor. Furthermore, there are the methods `guard` to add a guard link, `service` to add a service, `action` to add an action link, `dest` to set the destination of the wiring and `build` to finally create the wiring.

The `guard` method takes six arguments to set the entry type, the source container, the link operation, the link query, the link count and the flowdependency of the guard. `Action` takes five arguments to set the entry type, the destination container, the link operation, the link query and the link count of the action.

To call the `service` method a `WiringEntryBuilder.ServiceBuilder` object has to be delivered. One constructor for the `WiringEntryBuilder.ServiceBuilder` takes a `String` for the services name and a `Class` for the service class. Furthermore, it has `guard` and `action` methods to add the services guard and action links.

To give an example algorithm 6.2 shows the Java code for the `GetMatchingFoodOffersWiring`.

Algorithm 6.2: Java code of the `GetMatchingFoodOffersWiring`

```
new WiringEntryBuilder("GetMatchingOffersWiring").
  guard(EntryType.getEntryType("foodSearch"), Address.PIC,
```

```
Link.LinkOperation.TAKE, (coData) -> {
    return (coData.containsKey(status) &&
        coData.get(status) ==
        FoodSearchStatus.NEW);
}, LinkCount.EXACTLY_ONE, true).
guard(EntryType.getEntryType("foodOffer"),
    "/" + "foodOfferManagerPeer" + ":PIC",
    Link.LinkOperation.TAKE, LinkQuery.ALL,
    LinkCount.EXACTLY_ONE, true).
service(new WiringEntryBuilder.ServiceBuilder
    ("GetMatchingOffersService", GetMatchingOffers.Class)).
guard(EntryType.getEntryType("foodSearch"),
    Link.LinkOperation.TAKE, LinkQuery.ALL,
    LinkCount.EXACTLY_ONE).
guard(EntryType.getEntryType("foodOffer"),
    Link.LinkOperation.TAKE, LinkQuery.ALL,
    LinkCount.ALL).
action(EntryType.getEntryType("foodSearch")).
action(EntryType.getEntryType("foodOffer")).
action(EntryType.getEntryType("foodSearch"), Address.POC,
    Link.LinkOperation.TAKE, LinkQuery.ALL,
    LinkCount.EXACTLY_ONE).
action(EntryType.getEntryType("foodOffer"), Address.POC,
    Link.LinkOperation.TAKE, LinkQuery.ALL, LinkCount.ALL).
dest(new Address(new Address.PeerAddress("foodSharingPeer"),
    Address.WSCAddress)).
build();
```

6.3 Using S-FOSM in a food sharing simulation

This section proves that the S-FOSM is indeed a valid solution for the food sharing problem. For this purpose, a food sharing simulation environment is created, where the S-FOSM is used for the coordination of surplus food and food needs.

In the next section (see subsection 6.3.1) the implementations of the services used for the simulation are described. Then Subsection 6.3.2 explains how the sup-peers are implemented for the simulation. Afterwards subsection 6.3.3 gives some details about the simulation environments and finally subsection 6.3.4 discusses the results of the simulation.

6.3.1 Implementation of the Services

In the Peer Model only coordination logic is modelled, business logic can be injected in abstract services. The S-FOSM uses nine services that must be implemented for the

simulation. The next paragraphs describe how these services are implemented for the simulation.

The `SelectFoodService` should select the best food product for a food search. The implementation for the simulation selects one of the food products with minimal transport time.

The `NoMatchingFoodOffersService`, the `NoFeasibleTransportService`, the `ReservationTimeoutService`, the `FoodNotReceivedService`, the `FoodNotPickedUpService` and the `MyFoodOfferExpiredWiring` are called when problems occur during a search or the offering of food (e.g. no matching food offers found) in order to handle them. A good way to cope with these problems would be to inform the user and provide some possibilities to continue (e.g. start a new food search). However, for the simulation it is sufficient that these services just log the problem for the summary after the simulation run.

The `ReleaseFoodService` respectively the `FoodReceivingService` are called when a food product is released respectively delivered. In a real application scenario, these services must monitor the handover of the food. For the simulation, it is again sufficient to log these events.

6.3.2 Mock-ups of the sub-peers

In the S-FOSM the sub-peers `FoodOfferManagerPeer` and `TransportManagerPeer` were not modelled with the Peer Model, their behaviour was only described textual to leave space for different implementations of these sub-peers. However, for the simulation of the S-FOSM concrete versions are needed. The following paragraphs describe how the sub-peers were implemented for the simulation. The used implementations for the sub-peer are not optimal solutions for a productive food sharing setting, but adequate for the simulation.

FoodOfferManagerPeer

During one simulation run the number of food sharers is constant (i.e no food sharers leave, no food sharers join). Furthermore, each `FoodSharingPeer` cooperates with each other `FoodSharingPeer`. So, the idea of the `FoodSharingPeer` is that it informs all other `FoodSharingPeers` about all changes of its `myFoodOffers`. In order to do that, the `FoodOfferManagerPeer` (see Figure 6.33) needs three wirings: the `DistributeMyFoodOffersWiring`, the `AddFoodOfferWiring` and the `DeleteFoodOfferWiring`.

The `DistributeMyFoodOffersWiring` is responsible for informing other `FoodOfferManagerPeers` about changes of its `myFoodOffers`. First it takes a `scheduleDistributeFoodOffers` (guard G1) and reads `myFoodOffers` (guard G2). Then the service determines if the `myFoodOffers` have changed since the last execution and for each new/missing `myFoodOffer` for all other `FoodSharingPeers` a `addFoodOffer/deleteFoodOffer` entry

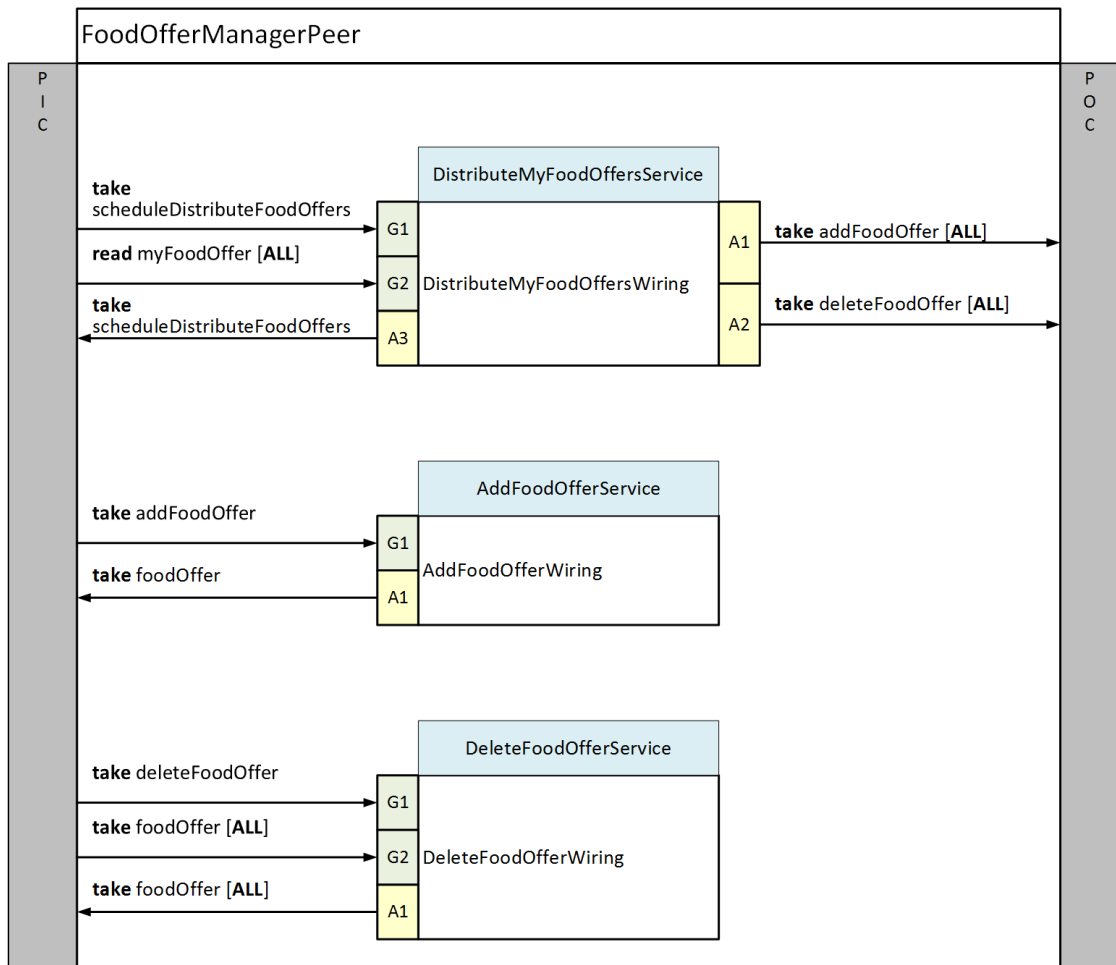


Figure 6.33: FoodOfferManagerPeer

is created and sent to the POC (action A1 and A2). Furthermore, the `tts` of the `scheduleDistributeFoodOffer` is set before it is written to the PIC (action A3).

All `addFoodOffer` entries from other `FoodOfferManagerPeers` are processed by the `AddFoodOfferWiring`. It takes the `addFoodOffer` (guard G1) and executes the service to create a new `foodOffer` which is written to the PIC (action A1).

When a `FoodOfferManagerPeer` receives a `deleteFoodOffer` entry the `DeleteFoodOfferWiring` is activated to remove the `foodOffer`. The wiring takes the `deleteFoodOffer` (guard G1) and all `foodOffers` (guard G2) before the service deletes the `foodOffer` denoted in `deleteFoodOffer`. Finally, all other `foodOffers` are written to the PIC (action A1).

TransportManagerPeer

The idea behind the implementation of the TransportManagerPeer (see Figure 6.34) is that there is a fixed number of transporters that have a base station. From there, each of them can reach all FoodSharingPeers, but only transport one food product at once. For the implementation of the TransportManagerPeer four wirings are used: the HandleTransportFoodOfferRequestWiring, the HandleReservationTransportRequestWiring, the HandleCancelTransportReservationWiring and the TransportFoodWiring.

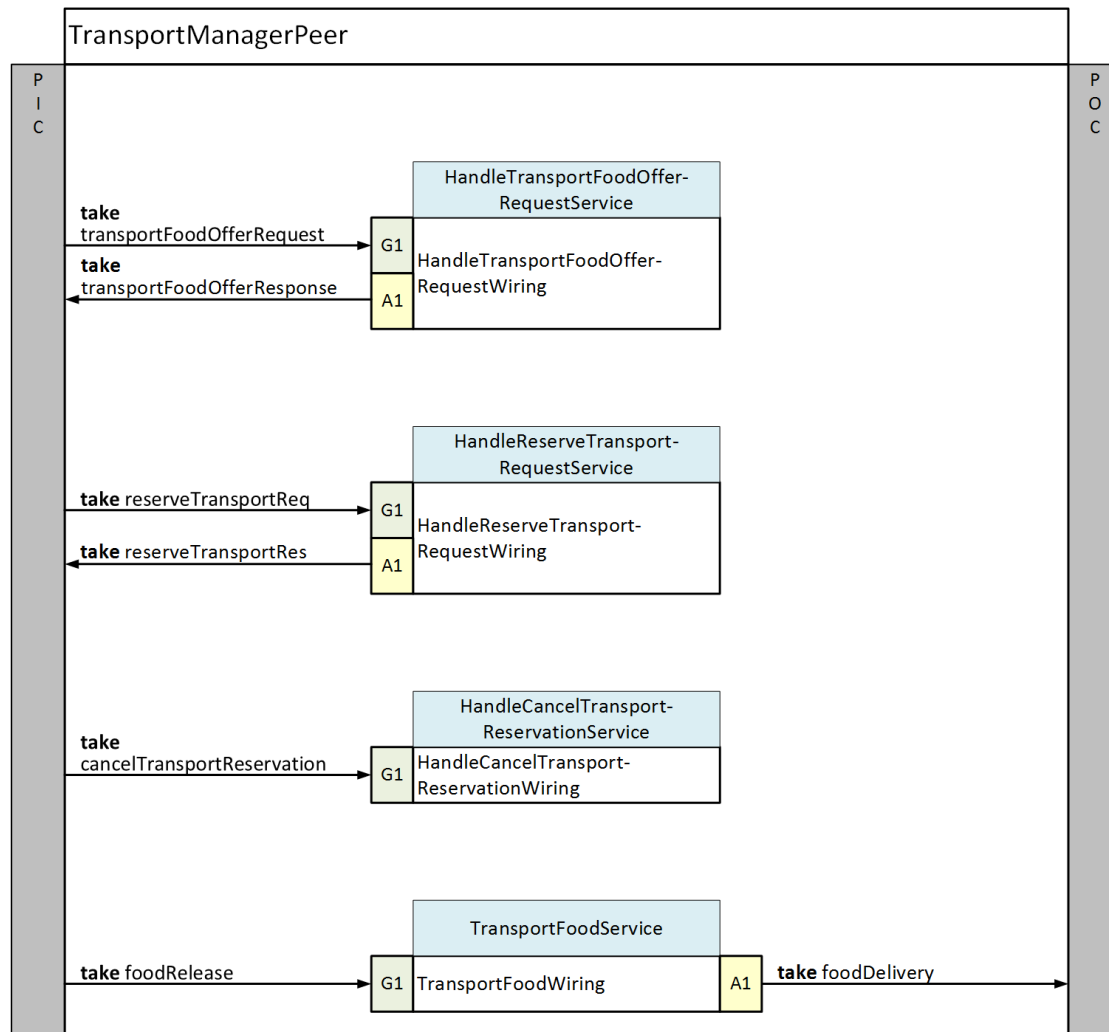


Figure 6.34: TransportManagerPeer

The HandleTransportFoodOfferRequestWiring receives a transportFoodOfferRequest (guard G1) and responds in a transportFoodOfferResponse (action A1) whether and when the transport of a food product is possible. Therefore, the service checks first if one of the transporters is available and then if it could perform the transport in the

given time.

When the `TransportManagerPeer` receives a `reserveTransportReq` the `HandleReserveTransportRequestWiring` is activated. Its service checks if there is still a transporter available, and only if so it assigns the transport to a transporter and responds with a `reserveTransportRes` (action A1).

With the `HandleCancelTransportReservationWiring` a previous reserved transport can be cancelled. The wiring takes a `cancelTransportReservation` (guard G1) and its service removes the transport from the transporter so that it is available for other assignments.

The `TransportFoodWiring` is responsible for the transport of a food. It takes a `foodRelease` (guard G1) before its service creates a `foodDelivery` whose `dest` property is set to the `TransportManagerPeer` of the receiving `FoodSharingPeer` and waits until the time for the delivery has elapsed. Then the `foodDelivery` is sent to the POC (action A1).

6.3.3 Simulation environment

In a productive environment, each food sharing participant would have a own runtime environment for the Peer Model where its instance of the `FoodSharingPeer` resides. For the simulation, it is adequate to use one runtime environment with all `FoodSharigPeer` instances (one for each food sharing participant).

The simulation environment provides several parameters to parametrise the simulation runs. The parameter `simulatedDays` is used to set the number of days of food sharing that are simulated with one simulation run. Furthermore, the parameter `simulationSpeedup` defines how much faster the simulation is than the reality. A `simulationSpeedup` of 86400 means that one day (= 86400 seconds) is simulated in one second.

The parameter `numberOfOfferingFoodSharers` is used to set the number of food sharing participants that create food offers, but no food searches and the parameter `FoodOffersPerFoodSharerAndDay` specifies the average number of food offers each food sharer creates per simulated day. Similar the parameter `numberOfSearchingFoodSharers` controls the number of food sharing participants that create food searches, but no food offers and the parameter `FoodSearchesPerFoodSharerAndDay` sets the average number of food searches each of them creates per simulated day. Furthermore the parameter `numberOfDualFoodSharers` is used to set the number of food sharers that create food offers and food searches. The point of time when a food offer respectively a food search is created is randomly chosen within the simulation time. The product name of a food offer is only one of the following three values: apple, banana and strawberry. The same applies for the search term of the food search. The pickup location of the food offer respectively the delivery location of the food search is set to the location of the corresponding food sharer. The location of a food sharing participant is generally random, but the parameter `maxDistanceFromBase` defines the maximal

distance between a food sharing household and the base station. With the parameter `foodOfferPickupDuration` the duration a food offer is available for pick up and with `foodSearchDeliveryDuration` the timespan a searching food sharer is available for delivery is set.

With the parameter `numOfConcurrentTransports` the number of transporters used for the simulation can be set. Furthermore, the parameter `transportSpeed` defines their speed (in meters per second).

The `FoodOfferManagerPeer` can be parametrised using the parameter `foodOffersUpdateInterval`. It sets the timespan the `FoodOfferManagerPeer` checks for updates of own food offers.

The main events of the simulation (e.g. food sharer has a new food offer, food offer has expired and food was picked up) are logged during the simulation. A graphical interface shows actual statistics (e.g. number of food offers, number of expired food offers and number of picked up food) during the simulation run.

6.3.4 Simulation results

The goal of the simulation is to prove that the S-FOSM is a valid solution for the coordination parts of the food sharing problem. This is the case when some of the food offers created by the food sharing participants are requested by searching food sharers. Furthermore the performance of S-FOSM is evaluated by the ratio of food offers that lead to food handovers.

Unfortunately, data about food sharing behaviour (e.g. when participants are at home for handover) are rare. Therefore, a fictitious example was used for the simulation runs. The fictitious example is a small community with 20 households. Near the village square in the centre of their village they have a base station for three transport robots (`numOfConcurrentTransports = 3`). Each of them can move autonomously with a speed of one meter per second (`transportSpeed = 1`). The houses of the villagers are not more than 500 meters away from the base station (`maxDistanceFromBase = 500`). Four of the 20 households use food sharing only to give away surplus food (`numberOfOfferingFoodSharers = 4`), four of them only want to get food (`numberOfOfferingFoodSharers = 4`). The remaining twelve households want to give away and get food (`numberOfDualFoodSharers = 12`). In average each household creates two food offers respectively two food searches per day (`FoodSearchesPerFoodSharerAndDay = 2` and `FoodOffersPerFoodSharerAndDay = 2`). Food offers can be requested up to two days after their creation (`foodOfferPickupDuration = 2`) and food sharing participants wait not longer than one day to get the food they want (`foodSearchDeliveryDuration = 1`). In a simulation run, a whole week of the village is simulated (`simulatedDays = 7`) accelerated by a factor of 5,000 (`simulationSpeedup`).

Each simulation run creates 224 food offers and 224 food searches. In each of the ten simulation runs (see Table 6.1) between 184 and 207 food handovers have been taken

place. That means that between 82.14% and 92.41% of the food offers respectively of the food searches have led to a food handover. Therefore the model is indeed a valid solution for the food sharing problem. Furthermore, at least for this example it had delivered pretty good results.

simulation run	food handovers	ratio
1	207	92.41%
2	185	82.59%
3	193	86.16%
4	187	83.48%
5	193	86.16%
6	200	89.29%
7	198	88.39%
8	189	84.38%
9	186	83.04%
10	184	82.14%

Table 6.1: Simulation results

6.4 Summary

In this chapter the S-FOSM, a model to coordinate food sharing between households was developed. At first the Peer Model was used to model the S-FOSM. The S-FOSM was illustrated with the graphical representation of the Peer Model and its sequence was explained. Then the S-FOSM was adapted and translated to source code that could be executed with the Java runtime of the Peer Model. Finally, a simulation environment was created where several simulated households use the S-FOSM to share food among each other. Simulation runs have shown that the model works and the participants can give away surplus food or get food they need. Based on S-FOSM and the other ICTs suited to improve food sharing (see chapter 4) the next chapter (see chapter 7) will describe the vision of a smart food sharing solution.

S-FOSS: a vision of a Smart Food Sharing Solution

Chapter 2 has found out that households are responsible for a big share of the worldwide food waste. Furthermore chapter 3 has shown that food sharing solutions for this audience are harder to provide than others and that current food sharing solutions for households are too expensive and cumbersome to use for many people. This chapter will integrate the ICTs described in chapter 4 and chapter 5 and describe S-FOSS, a vision of a smart food sharing process that is convenient to use and fits into today's smart cities. Although the S-FOSS is intended for households, it can also be used by retailers, producers, charities and others. Please also note that S-FOSS is visionary and not all of its requirements (e.g. autonomous transport robots that are allowed to use the public streets) are already given.

The following section (see section 7.1) describes properties of the S-FOSS and explains that it improves current food sharing solutions. Then section 7.2 shows some examples how S-FOSS can be used by its users. Finally, section 7.3 sum up the chapter.

7.1 Properties of the S-FOSS

This section will describe important properties of food sharing platforms and how they are solved in the S-FOSS in contrast to current food sharing systems.

The following subsection (see subsection 7.1.1) is about product information. Then subsection 7.1.2, subsection 7.1.3 and subsection 7.1.4 describe the monitoring of food products, the coordination of food offers and food needs and the transport of food products in the smart food sharing solution. Finally, subsection 7.1.5 is about intermediate food storages.

7.1.1 Product information

Information about the food products are crucial for a food sharing solution. Only with adequate product information food sharers can search for edibles and get the foodstuff they need.

Current food sharing solutions

In current food sharing solutions food sharers that create a food offer are responsible to enter product information. They have to transcribe the text from the food products label to the food sharing system. This process is very expensive and the data is often incomplete or faulty. Furthermore, often several food products are summarised to one food offering.

Food sharers that need food, search themselves. If they see an interesting product with incomplete product information, they can try to get the missing data (e.g. write a message to the offerer) or take the food anyway.

Requirements for the smart food sharing solution

In smart food sharing, food needs are automatically matched to food offers. This process only works if there are machine readable, accurate and detailed information of each available food product. Hence all food products must have a machine readable label that provides the product information. Food sharers use a reader that quickly reads this information and forwards it to the smart food sharing solution.

S-FOSS

The asset and licence costs for standardised RFID solutions are very low. Furthermore, they enable process optimisations (e.g. inventory or checkout) through the whole supply chain with considerable savings. So, all food maker equip their products with standardised RFID tags that uniquely identify the product they are attached to. Standardised and public available web services provide detailed information (e.g. name, producer, weight, expiration date, ingredients and nutrition facts) about each food product and its way through the supply chain.

Each food sharer has a RFID reader that reads the tag of the product and provides the product information for the S-FOSS, before an edible is shared. Home-made products must be equipped with RFID tags by the food sharer and the product information must be entered to a web service.

7.1.2 Food monitoring

Before a food sharer creates a food offer she or he has to detect that she or he has a food product that she or he does not need anymore. Therefore a good monitoring and easy inventory of food products is important for a prosperous food sharing solution.

Current food sharing solutions

In current food sharing solutions, food sharers are responsible for monitoring their edibles. They must supervise their stock themselves and recognise when they have food that they do not need before it goes off. Then they must create a food offer and enter the product information by hand.

Requirements for the smart food sharing solution

In smart food sharing the food stock is monitored by the system. A food sharer can always get an actual list of all edibles he possesses. Furthermore, smart food sharing reminds the users to share food products before they go off.

When a food sharer wants to create a food offer she or he only needs to select the food product on the list and set handover options (when and where she or he can handover the food stuff). The product information is known by the smart food sharing solution.

S-FOSS

Current fridges have been replaced by smart fridges. Beside lots of other smart features they can read the RFID tags of food products and have an internet connection. Furthermore, all other shelves that store edibles (e.g. in the cellar) are equipped with RFID readers and internet connection as well. So the smart food sharing can provide each food sharer her or his current food inventory and remind her or him to share surplus food.

Intermediate storage (e.g. Public fridges) and transporters of the smart food sharing solution have RFID readers and internet connection too. So, the S-FOSS can always determine how much and what food products are available at the moment.

7.1.3 Coordination of food offers and food needs

People that use a food sharing solution want to give off surplus food or get edibles they need (or both). So, the main task of food sharing solution is to find for each person that searches food another participant that offers a suitable product and vice versa.

Current food sharing solutions

In most current food sharing solutions members that have surplus food products create food offers. Participants that want food have to browse these food offers to find a matching offer. Then they use text messages to request a food offer and arrange the hand over.

For people who need food this process is very cumbersome. At first they must look through lots of offers to find the right one and then wait until the offering partner answers if they can have the food or not.

Requirements for the smart food sharing solution

In the smart food sharing process people that want food enter their requirements (e.g. what products they want, when and where) and the smart food sharing solution finds immediately the best matching food offer for that search. Then the offering partner is informed that its offer will be picked up, before it is transported to the searching partner.

The matching of food offers and food needs is a coordination task. Therefore, for the smart food sharing solution a coordination framework should be used to solve this problem in an appropriate way.

S-FOSS

The S-FOSS uses the Peer Model to coordinate the food offers and food searches and find matches. As FOSM(see section 6.1) is a model that solves the food sharing problem, it can be used for the S-FOSS. Only some minor adaptations (e.g. support of intermediate stores and reservation of several food products in one transaction) must be performed.

7.1.4 Transport of food products

Food sharers that need food are seldom at the position where the food they want is offered. Hence, for a working food sharing solution, food must be transported from the offering member to the searching participant.

Current food sharing solutions

In current food sharing solutions members that want to exchange edibles have to arrange the transport of the product themselves. Once a searching participant has found a matching food offer she or he has to exchange text messages with the offering partner to find a date and time to hand over the food.

This process needs lots of text messages and time. Moreover, they could fail on arranging the handover and the searching partner must find another food offer.

Requirements for smart food sharing

In the smart food sharing solution members that create food offers always have to enter when and where they can hand over their products. Also participants that want food have to define where and when they can take over the food. In addition, the smart food sharing process supports external transporters that carry food between the members. Thus, also members that cannot hand over food directly can share food.

For participants that want food, the food sharing solution automatically finds the best food offer. Thus, it only considers food offers where the transport of the products is possible. From these food offers it selects that one that matches the food search best.

S-FOSS

Autonomous vehicles have reached a degree of dependability so that they are allowed to use the public roads. Since they are electrically driven they are ecologically friendly and cheap. Authorities have recognised that food sharing is a good way to save the environment and improve the quality of life of the residents. Therefore, they sponsor autonomous transport vehicles in order to make food sharing more comfortable.

Members that have surplus food just create food offers and participants that need food let the S-FOSS search for suitable edibles. When there is a match the transport robots pick up the food at the offering partners home and deliver it to the partner who needs it.

7.1.5 Intermediate storages

In some cases food sharers want to give off food, but do not have the time to wait until some other food sharer needs it (e.g. they want to leave for holiday or need the space for other food). This food should be brought to an intermediate storage until its needed by a food sharer in order to decrease food waste and make the food sharing more efficient.

Current food sharing solutions

Most current food sharing solutions do not support the concept of intermediate storages, only foodsharing.de operates so called Fair-Teiler. Food sharers that do not want to wait until their offer is requested can put it into this public fridge and others can take everything they need from it.

The main problem is that food sharers do not know the status (i.e. if there is some free space for food offers or what food is offered at the moment) of these fridges. Furthermore, they are often placed in stores with limited opening hours.

Requirements for smart food sharing

In the smart food sharing solution intermediate stores detect that the maximum pick up time of a food product will elapse soon. If the food product could be needed by another food sharer in future (e.g. the edible does not expire in the next days) and the intermediate store is not full, it requests the food.

Intermediate stores create a food offer for each food product they store at the moment, so that it can be requested by other food sharers. The intermediate stores can take and give food at any time (i.e. they have no limiting opening hours). Products at intermediate stores that are not requested and go bad must be detected and disposed.

S-FOSS

Intermediate stores are equipped with automatic loading systems that take food from transporters and store them or bring stored products to transporters at any time. They keep the overview of their products and dispose food that is gone bad automatically. The

automatic loading system is very space efficient and so the intermediate store runs at little operating costs. Like the transport robots the intermediate storages are sponsored by authorities.

7.2 The S-FOSS in action

This section will give some examples of how food sharers use the S-FOSS to give off surplus food or search for food they need. As up to now the smart food sharing solution is a vision, storyboards are used to explain how food sharers interact with the S-FOSS.

The GUI prototypes for the storyboards were created with Pencil, an open-source tool for GUI prototyping¹. The used GUI elements are included into Pencil, the pictures inside the GUI prototypes are partially self-made. The remaining ones are from pixabay², a website with free pictures and videos under the Creative Commons CC0 licence.

The next subsection (see subsection 7.2.1) describes the personas used in the storyboards. Then subsection 7.2.2 and subsection 7.2.3 will give examples how the S-FOSS can be used.

7.2.1 Personas

The following paragraphs describe the users that use the smart food sharing solutions in the storyboards. These are just imaginary persons, but they should give the reader a feeling who would use the smart food sharing solution and why.

Claudia Winkler

Claudia Winkler is 22 years old and studies medicine in Vienna. She lives there in a small flat in the eighteenth district. Her studies are really challenging and during the week Claudia spends most of her time in lectures and in the library.

Claudia pays a lot of attention to her diet and loves fresh vegetables and fruits. Her fridge always contains the ingredients that she needs to quickly prepare a delicious and healthy meal.

At the weekends, Claudia often visits her parents. They have a farm near Melk, where they produce organic grain that is sold to local bakeries. In a huge garden, they grow vegetables and fruits for personal use and to give it to family and friends. Claudia loves the countryside and often helps her parents with their farm and garden work. Furthermore, she is a volunteer at the red cross agency in Melk and most of her fiends live there.

From her parent's farm, Claudia knows how much hard work and resources are needed to produce high quality food products and so she does not want that her food goes off or

¹<https://pencil.evolus.vn/>

²<https://pixabay.com/>

she has to waste edibles. But due to her food preferences and her busy lifestyle there are always some fruits or vegetables she cannot eat before they go off.

Therefore, Claudia uses the S-FOSS. She does not need to waste food, she can share it before it gets bad. Claudia is always glad when her food gets to someone who needs it.

Michael Weber

Michael Weber is 45 years old and works as key account manager for a large international bank. He and his children Lena (10 years old) and Lukas (8 years old) live in a generous flat in the twentieth district of Vienna.

Michael is a single father and loves to spent time with his children. Unfortunately, the time with his children is very short during the week. Michael works very long and so they can only spend some hours after he picks up the children from after-school supervision and before they have to go to bed together. Therefore, Michael's weekends are reserved for his children.

Michael earns enough money to enable his children a comfortable life now, but he grew up in humble conditions. He wants to show his children that they can have a good life without spending a lot of money. He uses the S-FOSS because it confirms his attitude. He can get great food products for free and additionally save the environment.

7.2.2 Claudia goes rural for a long weekend

Claudia's best friend Daniela celebrates her 30th birthday in Melk on Wednesday eve, the day before a bank holiday. Claudia has no compulsory lectures on Friday so she will leave Vienna on Wednesday and stay until Sunday. She has planned this trip for a long time and added it to the calendar of her smartphone.

Reminder to create food offer

Claudia comes home from university on Wednesday afternoon. She must pack her things and leave the flat within one hour to reach her train to Melk. The S-FOSS knows her schedule and that some of the food in her smart fridge will go off during the weekend. Therefore, the S-FOSS app shows a notification on her smartphone (see figure 7.1) to remind her.

Claudia opens the app to see a list with all the products that expire until Sunday (see figure 7.2). By default, all products on the list are selected for sharing, but Claudia wants to eat the cheese when she comes back on Sunday and therefore she unselects it. By touching the add button Claudia could add additional products of her fridge, that do not expire during the weekend, for sharing.

Claudia pushes the share button and comes to the next step where she can define when and where she can hand over the food (see figure 7.3). The S-FOSS app has already

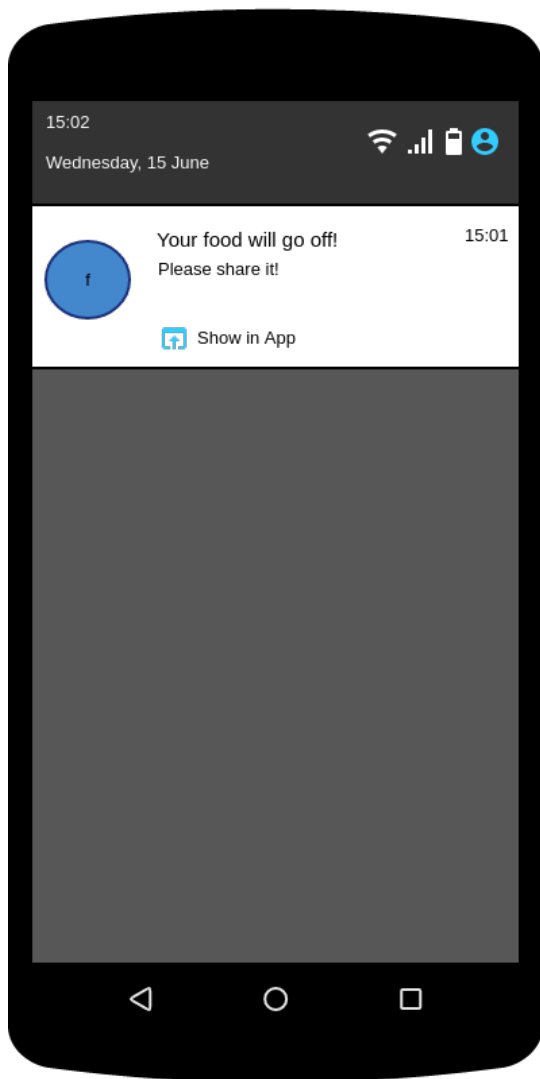


Figure 7.1: Notification to share food

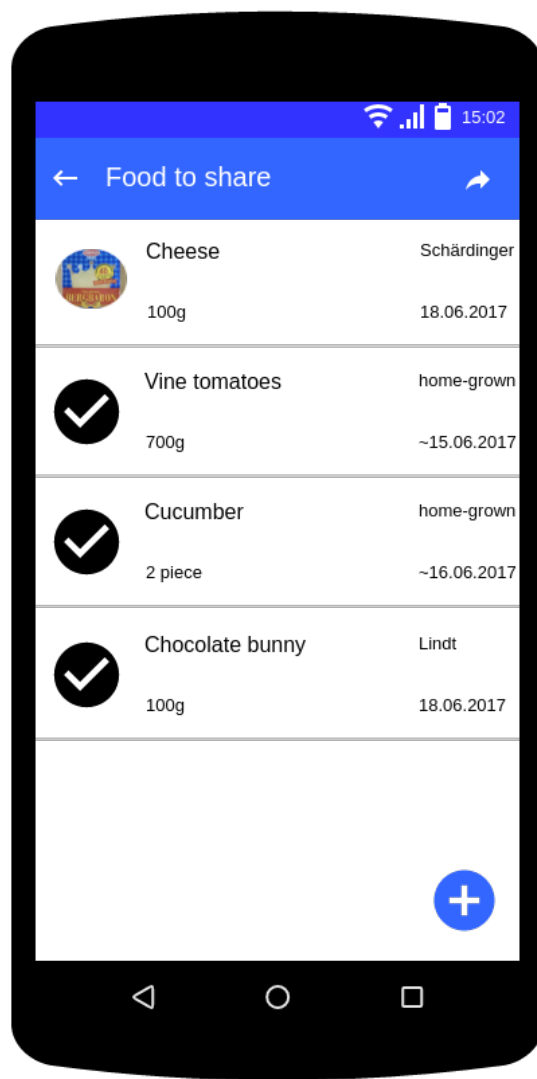


Figure 7.2: Select products to share

filled in her address as location and the time until she must leave as time. Hence Claudia had nothing to change.



Figure 7.3: Set Handover options

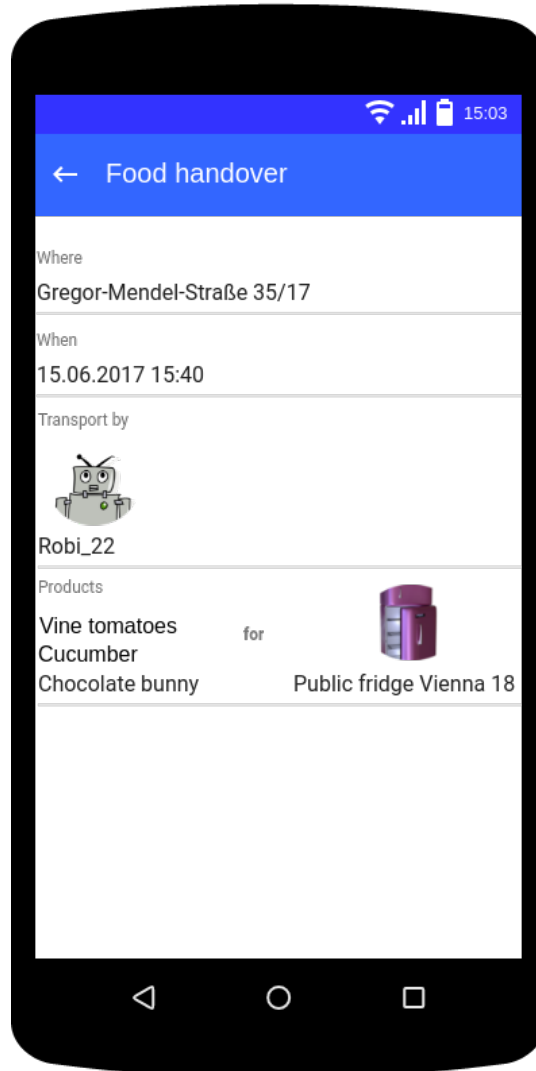


Figure 7.4: Food handover

Claudia pushes the share button again and in a final step she gets explained that food offers are created and that she will be informed when someone wants to have her food. Claudia lays aside her smartphone and starts to pack her suitcase.

Public fridge requests food

As the handover times are very short a public fridge in Claudia's neighbourhood decides to take the food until a food sharer needs it. Then the S-FOSS app notifies Claudia that

her offers had been requested. She opens the app (see figure 7.4) where she can see that a transport robot will come to her place and pick up the food in about 40 min. Claudia continues packing her stuff.

Food handover to transport robot

Nearly 40 minutes later, Claudia has just finished her packing, the S-FOSS app sends the next notification: The transport robot is here. Claudia takes the products from her fridge and walks to the robot outside of her flat. The robot recognizes the RFID tags of the food products and opens its lid as soon as Claudia is near the robot. Claudia puts the food inside. The robot closes its lid again and transports the goods to the public fridge. Claudia takes her suitcase, leaves her flat and reaches the train station in time.

7.2.3 Michael cooks with his children

It is a warm and sunny Sunday in May and Michael and his children had a late breakfast. They pack up there swimming things and go to a public sunbathing area near the New Danube. Lena, Lukas and Michael play with their volleyball, refresh themselves in the water and the time flies.

Search for food

It is already four o'clock when they take their first rest and lay down in the shade. Lena and Lukas started to play a card game when it comes in Michael's mind that he has to prepare dinner for him and the children in the evening. He can remember that after the breakfast there was not much food left at home, but he is not sure what. So, Michael takes his smartphone and opens the S-FOSS app to see his food inventory (see figure 7.5). As he suspected, there is not much in it, but he has an idea. With some canned tuna and cream cheese he could make the tuna pasta that his children really like. Furthermore, a tomato salad with mozzarella would be nice.

Michael begins a new food search and fills in the food he needs (see figure 7.6). Then he indicates when and where he could take over the food. The S-FOSS searches for these products and tells him that they will be delivered at 6 pm (see figure 7.7). Michael joins his children's card game for a final match before they leave the park.

Take food from the transport robot

Arrived at home Michael starts to boil the noodles. Sharp at 6 pm he gets a notification (see figure 7.8) by the S-FOSS app: his food is here. He opens the door where the transport robot already waits for him. Michael pushes the button to open the transport robots lid and together they unload the food. The robot closes its lid and drives away and Lena and Lukas help Michael to prepare a delicious dinner.

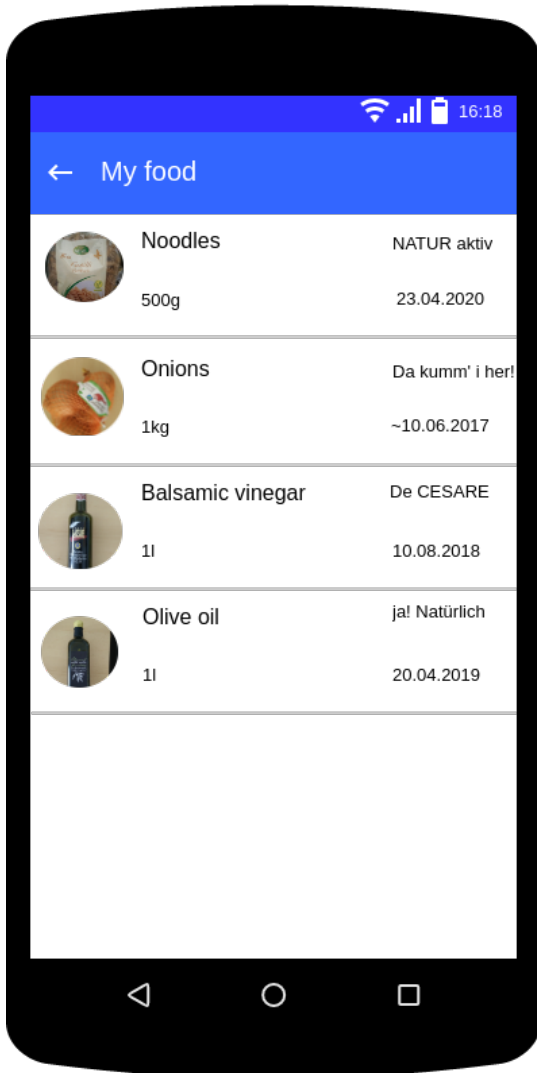


Figure 7.5: Show food inventory



Figure 7.6: Enter food needs

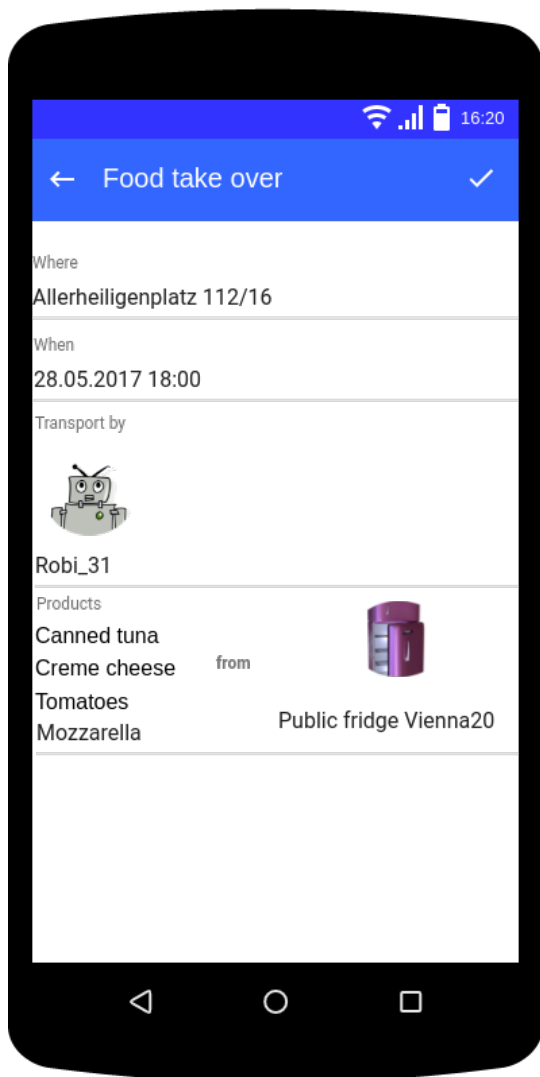


Figure 7.7: Food take over

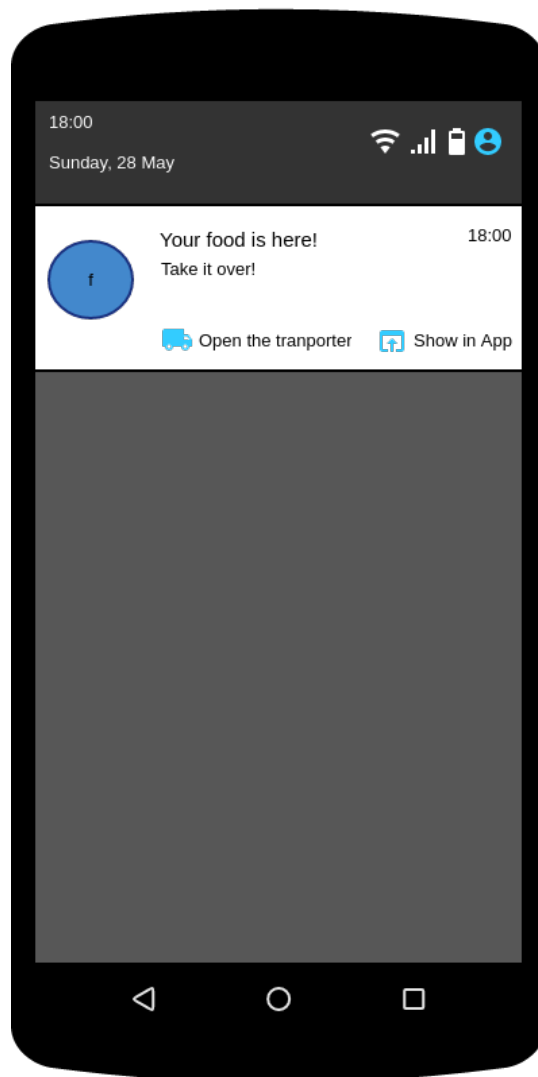
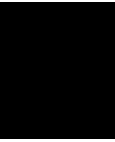


Figure 7.8: Reminder to take over food

7.3 Summary

This chapter has described S-FOSS, a vision of a smart food sharing process. At first properties of current food sharing solutions were described, as they should be for a smart food sharing solution and how the ICTs analysed in chapter 4 and chapter 5 can be used to establish that. Afterwards the chapter has described how users can use S-FOSS. Therefore, personas were created and storyboards were used to describe their interactions with S-FOSS.



Conclusion

The conclusion sums up the thesis and points out its contribution (see section 8.1). Additionally, it gives ideas for possible future works in the area of food sharing or with the Peer Model (see section 8.2).

8.1 Summary

This thesis has dealt with several aspects of food sharing in the modern human society. The following paragraphs will provide a summary and point out the contribution of the work.

The work started with a survey about the global food situation and analysed the magnitude, the negative impacts and the countermeasures for food losses and food waste as well as for food insecurity and undernourishment. Furthermore, it showed that food sharing has the potential to considerably improve both of these problems. Hence, with this global food situation survey the first goal of the work was reached.

Then research approaches and literature about food sharing as well as deployed food sharing initiatives have been reviewed. It has come out that there are various different forms of food sharing that are used by food sharing initiatives all over the world. This work has concentrated on food sharing solutions that want to reduce food waste and improve food security. It found out that food sharing initiatives that redistribute surplus food from producers and retailers to charities already perform well, but food sharing between households is inefficient. This is mainly because of too simple models of current food sharing initiatives. So, the second goal of the thesis was reached.

In the following the work analysed ICTs and how they can be used to improve food sharing solutions in order to make them more convenient for the users. A particular focus was to enhance the support for coordination (e.g. the process to find a suitable food offer) of the food sharing people. Therefore, different coordination models have

been compared, the Peer Model was selected as most suitable to handle the food sharing process and an overview of the Peer Model was given. This accomplished the third goal of the thesis.

The Peer Model was used to create S-FOSM (a Smart FOod Sharing Model). This was the first design of a food sharing process with a formal model. At first the S-FOSM was modelled with the graphical representation of the Peer Model. Then S-FOSM was translated to source code that can be executed with the Java runtime of the Peer Model. Furthermore, in a simulation, fictive households used the S-FOSM to show that they can share food with it. So, the fourth goal of the thesis was reached.

Finally, the previous described ICTs were taken and integrated into S-FOSS (a vision of a Smart FOod Sharing Solution). At first the thesis described the properties of the S-FOSS. Then prototypes of the GUI were created and fictive personas interacted in storyboards with the S-FOSS to show how its processes work. Thereby the fifth goal of the thesis was achieved.

8.2 Future work

This work has dealt intensively with food sharing and found some ideas for further works on this topic (see subsection 8.2.1). Furthermore, for creating the S-FOSM (see chapter 6) the work made use of the Peer Model and got ideas for future work on it (see subsection 8.2.2).

8.2.1 Food sharing

The topic of food sharing is very broad and there are lots of possibilities for future work. The following paragraphs are about topics that arose during the work on this thesis, but were not handled since they were out of scope.

Data about food losses and food insecurity are on the one hand very imprecise and on the other hand not very detailed (e.g. what food is wasted by whom). Food sharing and other countermeasures against these problems need better data in order to be improved. Furthermore, the collection of this data must be repeated in regular intervals to find out if the implemented methods are working or not.

The S-FOSM presented in chapter 6 is just one way to solve the coordination problem inherent to food sharing between households. The performance of this model can be compared to other models that provide solutions for the food sharing problem either in simulations (but therefore better data about food losses and food insecurity would be useful) or in real operation.

Chapter 7 describes S-FOSS, a real-life implementation could be done in a future work. As some of the prerequisites are not given (e.g. not all food products are equipped with RFID tags), for the implementation some alternatives must be found (e.g. scanning the bar code with the smart phone).

This thesis has mostly dealt with technical aspects of food sharing between households, but there is more about it. For example the lawgivers in many countries did not yet have reacted to the current trend in food sharing appropriately and so it is unclear under with conditions food sharing is allowed and when it is forbidden. Additionally, the legal situation differs from country to country. Hence, a detailed survey about the legal aspects of food sharing would be useful.

8.2.2 Peer Model

With the use of the Peer Model, the complex coordination problem food sharing between households, was to some degree easy to model and implement. But further improvements would make it even more comfortable to deal with coordination problems using the Peer Model. These ideas appeared during using the Peer Model:

A modeller to model coordination problems with the graphical notation of the peer model is still under development [Schon]. One can use a general-purpose modeller (e.g. in this thesis Visio was used), but it is quite cumbersome to find the right shapes for the Peer Model and assemble them to a model. The right tool would make the modelling more comfortable and prevent people from generating models that are not conform with the graphical notation of the Peer Model.

There exist different runtime environments of the Peer Model for various platforms (e.g. Java, Android and .NET). All of them support different features, but unfortunately none of them all features, that the formal definition of the Peer Model provides. For this work it was quite expensive to transform the created model into a model that only uses features supported by the Java runtime, the runtime with the most features. So, all the runtimes for the Peer Model (or at least one) should provide all features of the formal model.

For this work a lot of time was invested to implement the source code for the Java runtime from the graphical model. That is not difficult, but time-consuming. Hence, it should be done automatically by a code generator. There exists already a tool that can generate code for the embedded C runtime from Peer Model DSL, but it should also take graphical models and generate source code for all Peer Model runtimes.

List of Figures

2.1	Per capita food losses for different regions (from [GCS ⁺ 11])	7
2.2	Change of the magnitude of undernourishment in developing countries over the time (from [FIW15])	10
3.1	Types of food sharing (from [Dav16])	15
3.2	Average number of food sharing initiatives per city for different regions (from [DW17])	16
3.3	Prototype of a UrbanFoodSpot (from [Pla16])	22
3.4	Create a food basket with the Foodsharing website	24
3.5	Details page of the Fair-Teiler Amtshaus Wien 7 (Photos and names of Foodsharing users have been made irrecognisable)	24
3.6	Create listing with the OLIO Android app	26
4.1	Samsung Family Hub	33
4.2	Autonomous transport robot from Starship Technologies	35
5.1	Graphical representation of a peer (without wirings and sub-peers)	43
5.2	Graphical representation of a link (without source and target container)	44
5.3	Graphical representation of a wiring (guard and action links are not shown)	46
6.1	The FoodSharingPeer (from a high level point of view)	50
6.2	SendMyFoodOfferToManagerWiring	52
6.3	MyFoodOfferExpiredWiring	52
6.4	GetMatchingFoodOffersWiring	53
6.5	NoMatchingFoodOffersWiring	53
6.6	RequestTransportForFoodOfferWiring	53
6.7	UpdateStatusWiring	54
6.8	SelectFoodWiring	54
6.9	NoFeasibleTransportWiring	55
6.10	ReserveFoodAndTransportWiring	55
6.11	ReservationTimeoutWiring	55
6.12	ReserveFoodWiring	56
6.13	CancelFoodReservationWiring	56
6.14	PrepareFoodReleaseWiring	56
		89

6.15	ReleaseFoodWiring	56
6.16	FoodNotPickedUpWiring	57
6.17	ReserveTransportWiring	57
6.18	GetTransportReservationWiring	57
6.19	CancelTransportReservationWiring	58
6.20	PrepareForFoodReceivingWiring	58
6.21	FoodReceivingWiring	58
6.22	FoodNotReceivedWiring	58
6.23	GetMatchingFoodOffersWiring (adapted S-FOSM)	60
6.24	ReserveFoodAndTransportWiring (adapted reservation process)	61
6.25	CommitReservationsWiring (adapted reservation process)	61
6.26	ReservationTimeoutWiring (adapted reservation process)	61
6.27	CleanUpFoodReservationWiring (adapted reservation process)	61
6.28	CleanUpTransportReservationWiring (adapted reservation process)	62
6.29	ReserveFoodWiring (adapted reservation process)	62
6.30	CancelFoodReservationWiring (adapted reservation process)	62
6.31	PrepareFoodReleaseWiring (adapted reservation process)	62
6.32	PrepareForFoodReceivingWiring (adapted reservation process)	62
6.33	FoodOfferManagerPeer	66
6.34	TransportManagerPeer	67
7.1	Notification to share food	78
7.2	Select products to share	78
7.3	Set Handover options	79
7.4	Food handover	79
7.5	Show food inventory	81
7.6	Enter food needs	81
7.7	Food take over	82
7.8	Reminder to take over food	82

List of Tables

5.1	Comparison of popular coordination models	40
6.1	Simulation results	70

List of Algorithms

6.1	Java code of the FoodSharingPeer	63
6.2	Java code of the GetMatchingFoodOffersWiring	63

Acronyms

- AGT** automated guided transport. 33
- AGV** automated guided vehicles. 33
- ALE** Application Level Event. 31
- ASRW** Availability-Surplus-Recoverability-Waste. 17
- dest** destination. 41, 42, 44
- DoR** Degree of Recoverability. 17, 18
- EC** entry collection. 44–46
- EPC** Electronic Product Code. 30, 31
- FAO** Food and Agriculture Organisation of the United Nations. 5, 9
- fid** flow identifier. 41, 43–45
- FSC** food supply chain. 5, 6, 8, 17, 19
- FWTX** flexible wiring transactions. 46, 47, 59, 60
- GDP** gross domestic product. 8, 10
- GID** General ID. 31
- GTIN** Global Trade Item Number. 30
- GUI** graphical user interface. 3, 76, 86
- HF** high frequency. 30
- ICT** information and communication technology. xiii, 1–3, 14, 15, 17, 21, 22, 27, 29, 30, 32, 34–36, 70, 71, 83, 85, 86

LF low frequency. 30

MDG Millennium Development Goal. 9

ONS Object Naming Service. 31

PIC peer input container. 41, 42

PM-DSL Peer Model Domain Specific Language. 40

POC peer output container. 42

RFID radio-frequency identification. xiii, 29–32, 35, 36, 72, 73, 80, 86

S-FOSM a Smart FOod Sharing Model. xiv, 49–70, 86, 90

S-FOSS a vision of a Smart FOod Sharing Solution. xiv, 71–83, 86

SSCC Serialized Shipping Container Code. 30

ttl time-to-live. 41, 42, 44, 45

tts time-to-start. 41, 44, 45

UAV unmanned aerial vehicle. 34

UHF super high frequency. 30

UHF ultra high frequency. 30

USDA United States Department of Agriculture. 9, 12

WFP World Food Program. 11

WFS World Food Summit. 8, 9

WTX wiring transaction. 45, 46

XVSM extensible virtual shared memory. 42, 43

Bibliography

- [Arb04] Farhad Arbab. Reo: A channel-based coordination model for component composition. *Mathematical Structures in Comp. Sci.*, 14(3):329–366, June 2004.
- [Bar10] Christopher B. Barrett. Measuring food insecurity. *Science*, 327(5967):825–828, 2010.
- [BCC⁺10] Manuela Bucci, Caterina Calefato, Sergio Colombetti, Monica Milani, and Roberto Montanari. Fridge Fridge on the Wall: what Can I Cook for Us All? An HMI Study for an Intelligent Fridge. In *Proceedings of the International Conference on Advanced Visual Interfaces, AVI '10*, page 415. ACM, 2010.
- [BCH16] Graeme Bampton, Dalene Campbell, and Werner Heyns. Autonomous transport - the future is now. *Civil Engineering : Magazine of the South African Institution of Civil Engineering*, 24:9–16, September 2016.
- [BEK93] Omran A. Bukhres, Ahmed K. Elmagarmid, and eva Kühn. Implementation of the Flex Transaction Model. *IEEE Data Eng. Bull.*, 16(2):28–32, June 1993.
- [BKHM⁺16] G. Bernhofer, M. Kalleitner-Huber, G. Mraz, C. Pladerer, M. Pohl, E. Weißenböck, R. Wauschek, and S. Gemballa. UrbanFoodSpots – Kühlstationen mit integriertem Informationssystem zur Lebensmittelweitergabe im urbanen Raum. Technical report, Österreichisches Ökologie-Institut, 2016.
- [BVW⁺13] Robert E. Black, Cesar G. Victora, Susan P. Walker, Zulfiqar A. Bhutta, Parul Christian, Mercedes de Onis, Majid Ezzati, Sally Grantham-McGregor, Joanne Katz, Reynaldo Martorell, Ricardo Uauy, and the Maternal and Child Nutrition Study Group. Maternal and child undernutrition and overweight in low-income and middle-income countries. *The Lancet*, 382:427–451, June 2013.
- [CJRGS16] Alisha Coleman-Jensen, Matthew P. Rabbitt, Christian A. Gregory, and Anita Singh. Household Food Security in the United States in 2015. Economic Research Report 215, United States Department of Agriculture, September 2016.

- [CKS09] Stefan Craß, eva Kühn, and Gernot Salzer. Algebraic Foundation of a Data Model for an Extensible Space-Based Collaboration Protocol. In *Proceedings of the 2009 International Database Engineering & Applications Symposium*, pages 301–306. ACM, 2009.
- [CRC12] Alessandra Cozzolino, Silvia Rossi, and Alessio Conforti. Agile and lean principles in the humanitarian supply chain: The case of the United Nations World Food Programme. *Journal of Humanitarian Logistics and Supply Chain Management*, 2:16–33, 2012.
- [Csu14] Maximilian Alexander Csuk. Developing an Interactive, Visual Monitoring Software for the Peer Model Approach. Master’s thesis, Vienna University of Technology, 2014.
- [CV16] Aaron Ciaghi and Adolfo Villafiorita. Beyond food sharing: Supporting food waste reduction with ICTs. In *Smart Cities Conference (ISC2), 2016 IEEE International*, pages 1–6. IEEE, 2016.
- [Dav16] Anna R. Davies. Sharecity typologies of food sharing. Working paper 1, Trinity college Dublin, Ireland, 2016.
- [DEM⁺17] Anna R. Davies, Ferne Edwards, Brigida Marovelli, Oona Morrow, Monika Rut, and Marion Weymes. Creative construction: crafting, negotiating and performing urban food sharing landscapes. *Area*, 2017.
- [dLN15] Willem de Lange and Anton Nahman. Costs of food waste in South Africa: Incorporating inedible food waste. *Waste Management*, 40:167–172, March 2015.
- [Dob12] Daniel M. Dobkin. *The RF in RFID: UHF RFID in Practice*. Newnes, Kidlington UK, 2012.
- [DW17] A. Davies and M. Weymes. The SHARECITY100 Database. SHARECITY Briefing Note 1, Trinity college Dublin, Ireland, 2017.
- [ELLR90] Ahmed Elmagarmid, Yungho Leu, Witold Litwin, and Marek Rusinkiewicz. A Multidatabase Transaction Model for InterBase. In *Proceedings of the 16th International Conference on Very Large Data Bases*, pages 507–581. Morgan Kaufmann Publishers Inc, 1990.
- [FAO81] FAO. *Food loss prevention in perishable crops*. Number 43 in FAO agriculture organisation services bulletin. FAO, Rome, 1981.
- [FAO01] FAO. *The State of Food Insecurity in the World 2001*. FAO, Rome, 2001.
- [FAO03] FAO. *Trade reforms and food security: Conceptualizing the linkages*. FAO, Rome, 2003.

- [FAO06] FAO. Food security. Policy brief, FAO’s Agriculture and Development Economics Division (ESA), June 2006.
- [FAO14] FAO. *Mitigation of food wastage. Societal costs and benefits*. FAO, Rome, 2014.
- [Fin15] Klaus Finkenzeller. *RFID-Handbuch: Grundlagen und praktische Anwendungen von Transpondern, kontaktlosen Chipkarten und NFC*. Carl Hanser Verlag GmbH Co KG, Munich, 2015.
- [FIW15] FAO, IFAD, and WFP. *The State of Food Insecurity in the World 2015. Meeting the 2015 international hunger targets: taking stock of uneven progress*. FAO, Rome, 2015.
- [Flä16] Heike Flämig. Autonomous vehicles and autonomous driving in freight transport. In *Autonomous Driving*, pages 365–385. Springer, Berlin, 2016.
- [Flö05] Christian Flörkemeier. EPC-Technologie — vom Auto-ID Center zu EPC-global. In *Das Internet der Dinge: Ubiquitous Computing und RFID in der Praxis: Visionen, Technologien, Anwendungen, Handlungsanleitungen*, pages 87–100. Springer, Berlin, 2005.
- [FM89] Anna T.C. Feistner and William C. McGrew. Food-sharing in primates: a critical review. *Perspectives in primate biology*, 3:21–36, 1989.
- [FS16] A. D. Floarea and V. Sgârciu. Smart refrigerator: A next generation refrigerator connected to the IoT. In *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages 1–6. IEEE, June 2016.
- [FWCF14] Jeremy Farr-Wharton, Jaz Hee-Jeong Choi, and Marcus Foth. Food talks back: exploring the role of mobile applications in reducing domestic food wastage. In *Proceedings of the 26th Australian Computer-Human Interaction Conference on Designing Futures: the Future of Design*, pages 352–361. ACM, 2014.
- [FWFC13] Jeremy Farr-Wharton, Marcus Foth, and Jaz Hee-jeong Choi. Eatchafood: challenging technology design to slice food waste production. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 559–562. ACM, 2013.
- [GB06] Bill Glover and Himanshu Bhatt. *RFID Essentials*. O’Reilly Media, Inc., Sebastopol USA, 2006.
- [GBC+10] H. Charles J. Godfray, John R. Beddington, Ian R. Crute, Lawrence Haddad, David Lawrence, James F. Muir, Jules Pretty, Sherman Robinson, Sandy M. Thomas, and Camilla Toulmin. Food security: the challenge of feeding 9 billion people. *science*, 327:812–818, 2010.

- [GCS⁺11] Jenny Gustavsson, Christel Cederberg, Ulf Sonesson, Robert Van Otterdijk, and Alexandre Meybeck. *Global food losses and food waste – Extent, causes and prevention*. FAO, Rome, 2011.
- [GFC13] Eva Ganglbauer, Geraldine Fitzpatrick, and Rob Comber. Negotiating Food Waste: Using a Practice Lens to Inform Design. *ACM Trans. Comput.-Hum. Interact.*, 20:11:1–11:25, May 2013.
- [GFSG14] Eva Ganglbauer, Geraldine Fitzpatrick, Özge Subasi, and Florian Güldenpfennig. Think Globally, Act Locally: A Case Study of a Free Food Sharing Community and Social Networking. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, CSCW '14*, pages 911–921. ACM, 2014.
- [Gib12] Mark Gibson. Food security—a commentary: What is it and why is it so complicated? *Foods*, 1(1):18–27, December 2012.
- [GLU12] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, June 2012.
- [GMP14] Paola Garrone, Marco Melacini, and Alessandro Perego. Opening the black box of food waste reduction. *Food policy*, 46:129–139, 2014.
- [Ham15] Thomas Hamböck. Towards a Toolchain for Asynchronous Embedded Programming based on the Peer-Model. Master’s thesis, Vienna University of Technology, 2015.
- [HBS73] Carl Hewitt, Peter Bishop, and Richard Steiger. A Universal Modular ACTOR Formalism for Artificial Intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence, IJCAI’73*, pages 235–245, San Francisco, 1973. Morgan Kaufmann Publishers Inc.
- [HBSA16] A. Hachani, I. Barouni, Z. Ben Said, and L. Amamou. Rfid based smart fridge. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–4. IEEE, November 2016.
- [HGDC09] Kevin D. Hall, Juen Guo, Michael Dore, and Carson C. Chow. The Progressive Increase of Food Waste in America and Its Environmental Impact. *PLOS ONE*, 4(11):1–6, November 2009.
- [HS13] Sue Horton and Richard H. Steckel. Malnutrition global economic losses attributable to malnutrition 1900-2000 and projections to 2050. In *How Much have Global Problems Cost the World?* Cambridge University Press, Cambridge, 2013.
- [JVS11] Adrian V. Jaeggi and Carel P. Van Schaik. The evolution of food sharing in primates. *Behavioral Ecology and Sociobiology*, 65:2125–2140, June 2011.

- [KCH14] eva Kühn, Stefan Craß, and Thomas Hamböck. Approaching coordination in distributed embedded applications with the peer model DSL. In *40th EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA)*, pages 64–68. IEEE, 2014.
- [KCJ⁺13] eva Kühn, Stefan Craß, Gerson Joskowicz, Alexander Marek, and Thomas Scheller. Peer-Based Programming Model for Coordination Patterns. In *15th International Conference on Coordination Models and Languages (COORDINATION), held as part of the 8th International Federated Conference on Distributed Computing Techniques (DisCoTec)*, pages 121–135. Springer, 2013.
- [KCJN14] eva Kühn, Stefan Craß, Gerson Joskowicz, and Martin Novak. Flexible Modeling of Policy-Driven Upstream Notification Strategies. In *29th Symposium On Applied Computing (SAC)*. ACM, 2014.
- [KCS15] eva Kühn, Stefan Craß, and Gerald Schermann. Extending a Peer-Based Coordination Model with Composable Design Patterns. In *23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 53–61. IEEE, 2015.
- [Küh16] eva Kühn. Reusable coordination components: Reliable development of cooperative information systems. *International Journal of Cooperative Information Systems*, 25:1740001, 2016.
- [Küh17] eva Kühn. Flexible Transactional Coordination in the Peer Model. In *7th IPM International Conference on Fundamentals of Software Engineering (FSEN)*. Springer, 2017.
- [LADK06] Tuan Le-Anh and M.B.M. De Koster. A review of design and control of automated guided vehicle systems. *European Journal of Operational Research*, 171:1–23, May 2006.
- [Lan05] Jeremy Land. The history of RFID. *IEEE Potentials*, 4:8–11, 2005.
- [LdFM08] Jan Lundqvist, Charlotte de Fraiture, and David Molden. Saving Water: From Field to Fork – Curbing Losses and Wastage in the Food Chain. SIWI policy brief, 2008.
- [LJL09] Suhuai Luo, Jesse Jin, and Jiaming Li. A smart fridge with an ability to enhance health and enable better nutrition. *Int. J. Multimedia Ubiquitous Eng*, 4(2):66–80, 2009.
- [LU15] Katharina Lehmann-USchner. Die langfristigen Folgen von Mangel- und Unterernährung in Entwicklungsländern. Research Report DIW Roundup: Politik im Fokus, No. 69, DIW Berlin, 2015.

- [LW95] Douglas C. Long and Donald F. Wood. The logistics of famine relief. *Journal of Business Logistics*, 16(1):213–229, 1995.
- [MFI⁺16] Piergiuseppe Morone, Pasquale Marcello Falcone, Enrica Imbert, Marcello Morone, and Andrea Morone. New consumers behaviours in the sharing economy: An experimental analysis on food waste reduction. Working Paper 2016/11, Economics Department, Universitat Jaume I, Castellón (Spain), 2016.
- [Mic04] Gurven Michael. To give and to give not: The behavioral ecology of human food transfers. *Behavioral and brain sciences*, 27:543–583, August 2004.
- [MK05] Olaf Müller and Michael Krawinkel. Malnutrition and health in developing countries. *Canadian Medical Association Journal*, 173(3):279–286, 2005.
- [Mur16] Benjamin Murphy. Assessing the Sustainability of ICT Enabled Urban Food Sharing in Dublin. Master’s thesis, University of Dublin, 2016.
- [OGS10] Steven Were Omamo, Ugo Gentilini, and Susanna Sandström, editors. *Revolution: From Food Aid to Food Assistance*. WFP, Rome, 2010.
- [PA09] Per Pinstrup-Andersen. Food security: definition and measurement. *Food security*, 1:5–7, 2009.
- [PBM10] Julian Parfitt, Mark Barthel, and Sarah Macnaughton. Food waste within food supply chains: quantification and potential for change to 2050. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 365:3065–3081, 2010.
- [Pet62] Carl Adam Petri. *Kommunikation mit Automaten*. PhD thesis, Technische Hochschule Darmstadt, 1962.
- [Pla16] Christian Pladerer. UrbanFoodSpots: Kühlstationen mit integriertem Informationssystem zur Lebensmittelweitergabe im urbanen Raum. Conference talk slides, 2016. Recy&DepoTech 2016.
- [QJ09] Tom Quested and Hannah Johnson. Household Food and Drink Waste in the UK. Report, WRAP, 2009.
- [Rau14] Dominik Rauch. PeerSpace.NET: Implementing and Evaluating the Peer Model with Focus on API Usability. Master’s thesis, Vienna University of Technology, 2014.
- [RIfEuL08] Max Rubner-Institut and Bundesforschungsinstitut für Ernährung und Lebensmittel. Ergebnisbericht, Teil 2. In *Nationale Verzehrs Studie II*. Max Rubner-Institut and Bundesforschungsinstitut für Ernährung und Lebensmittel, 2008.

- [RKB16] Stacey Rosen, Thome Karen, and Meade Birgit. International food security assessment, 2016-2026. Technical report, U.S. Department of Agriculture, 2016.
- [Rot07] M. Rothensee. A high-fidelity simulation of the smart fridge enabling product-based services. In *2007 3rd IET International Conference on Intelligent Environments*, pages 529–532. IEEE, September 2007.
- [Rou12] José Rouillard. The Pervasive Fridge. A smart computer system against uneaten food loss. In *Seventh International Conference on Systems (ICONS2012)*, pages pp. 135–140. HAL CCSD, February 2012.
- [Sch14] Gerald Schermann. Extending the Peer Model with Composable Design Patterns. Master’s thesis, Vienna University of Technology, 2014.
- [Sch17] Jörg Schoba. Mobile Peer Model A mobile peer-to-peer communication and coordination framework - with focus on scalability and security. Master’s thesis, Vienna University of Technology, 2017.
- [Schon] Matthias Schwayer. Towards a Visual Design and Development Environment for the Peer Model. Master’s thesis, Vienna University of Technology, in preparation.
- [SJQM16] Åsa Stenmarck, Carl Jensen, Tom Quested, and Graham Moates. Estimates of European food waste levels. Report, FUSIONS, March 2016.
- [TCM⁺12] Anja Thieme, Rob Comber, Julia Miebach, Jack Weeden, Nicole Kraemer, Shaun Lawson, and Patrick Olivier. We’ve bin watching you: designing for reflection and social persuasion to promote sustainable lifestyles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2337–2346. ACM, 2012.
- [Thu11] Valentin Thurn. Taste the waste. DVD, 2011.
- [Til17] Peter Tillian. Mobile Peer Model A mobile peer-to-peer communication and coordination framework - with focus on mobile constraints and persistence. Master’s thesis, Vienna University of Technology, 2017.
- [TT10] Gerrit Tamm and Christoph Tribowski. *RFID*. Springer, Berlin, 2010.
- [VZA⁺14] D. G Victor, D. Zhou, E.H.M. Ahmed, P.K. Dadhich, J.G.J Olivier, H-H. Rogner, K. Sheikho, and M. Yamaguchi. Introductory chapter. In *Climate Change 2014: Mitigation of Climate Change*, Cambridge, 2014. Cambridge University Press.
- [Wan06] Roy Want. An introduction to RFID technology. *IEEE Pervasive Computing*, 5:25–33, 2006.

- [WFP] WFP. Contributions to wfp in 2016. <http://www.wfp.org/funding/year/2016>, last visited 24.03.2017.
- [Wru13] Thomas Wruß. Guidelines to Support Design and Development of Trust in Mobile Community Applications. Master's thesis, Vienna University of Technology, 2013.
- [XYL⁺13] Lei Xie, Yafeng Yin, Xiang Lu, Bo Sheng, and Sanglu Lu. iFridge: An Intelligent Fridge for Food Management Based on RFID Technology. In *Proceedings of the 2013 ACM Conference on Pervasive and Ubiquitous Computing Adjunct Publication*, UbiComp '13 Adjunct, pages 291–294. ACM, 2013.