Commonwealth Scientific and Industrial Research Organisation
Data61
PERCCOM Master Programme

Master's Thesis in
Pervasive Computing and Communications
for Sustainable Development

Niklas Kolbe

Reasoning over Knowledge-based Generation of Situations in Context Spaces to Reduce Food Waste

2016

Supervisors: *Prof. Arkady Zaslavsky* (CSIRO)
Dr. Sylvain Kubler (University of Luxembourg)

Examiners: *Prof. Eric Rondeau* (University of Lorraine)
Prof. Jari Porras (Lappeenranta University of Technology)
Prof. Karl Andersson (Luleå University of Technology)

This thesis is prepared as part of an European Erasmus Mundus programme PERCCOM - PERvasive Computing & COMmunications for sustainable development.

Co-funded by the
Erasmus+ Programme
of the European Union

This thesis has been accepted by partner institutions of the consortium (cf. UDL-DAJ, n°1524, 2012 PERCCOM agreement).
Successful defence of this thesis is obligatory for graduation with the following national diplomas:

- Master in Complex Systems Engineering (University of Lorraine)
- Master of Science in Technology (Lappeenranta University of Technology)
- Master in Pervasive Computing and Communications for Sustainable Development (Luleå University of Technology)

Abstract

Commonwealth Scientific and Industrial Research Organisation
Data61
PERCCOM Master Programme

Niklas Kolbe

Reasoning over Knowledge-based Generation of Situations in Context Spaces to Reduce Food Waste

Master's Thesis - 2016.

101 pages, 36 figures, 12 tables, 6 appendices.

Keywords: Situation Awareness, Context Space Theory, Situation Theory, O-MI/O-DF, Pervasive Computing, Ontologies.

With the ever-growing amount of connected sensors (IoT), making sense of sensed data becomes even more important. Pervasive computing is a key enabler for sustainable solutions, prominent examples are smart energy systems and decision support systems. A key feature of pervasive systems is situation awareness which allows a system to thoroughly understand its environment. It is based on external interpretation of data and thus relies on expert knowledge. Due to the distinct nature of situations in different domains and applications, the development of situation aware applications remains a complex process. This thesis is concerned with a general framework for situation awareness which simplifies the development of applications. It is based on the Situation Theory Ontology to provide a foundation for situation modelling which allows knowledge reuse. Concepts of the Situation Theory are mapped to the Context Space Theory which is used for situation reasoning. Situation Spaces in the Context Space are automatically generated with the defined knowledge. For the acquisition of sensor data, the IoT standards O-MI/O-DF are integrated into the framework. These allow a peer-to-peer data exchange between data publisher and the proposed framework and thus a platform independent subscription to sensed data. The framework is then applied for a use case to reduce food waste. The use case validates the applicability of the framework and furthermore serves as a showcase for a pervasive system contributing to the sustainability goals. Leading institutions, e.g. the United Nations, stress the need for a more resource efficient society and acknowledge the capability of ICT systems. The use case scenario is based on a smart neighbourhood in which the system recommends the most efficient use of food items through situation awareness to reduce food waste at consumption stage.

Acknowledgement

I would like to express my gratitude to everyone who supported the work of this Master's Thesis.

Firstly, thanks to my supervisors Prof. Arkady Zaslavsky and Dr. Sylvain Kubler for the discussions and feedback throughout the thesis work. Special thanks to Dr. Jérémy Robert from the University of Luxembourg for the feedback on the thesis and the support for the implementation regarding O-MI/O-DF. Thanks to Dr. Andrey Boytsov from TU Berlin for providing the ECSTRA source code and answering all my questions regarding the implementation.

Thanks to the PERCCOM consortium and the partners for organising this programme. Performing the studies jointly in France, Finland, Russia, Sweden and Australia was a unique, valuable and exciting experience.

I would like to thank my family for the continuous support, motivation and being there for me.

Last but not least, thanks to all my fellow PERCCOM colleagues; for all the jokes, gatherings, studies, discussions and trips that made these two years so incredible.

Melbourne, May 2016

Niklas Kolbe

Table of Content

List of Figures

List of Tables

List of Algorithms

Abbreviations and Symbols

API	Application Programming Interface
CSIRO	Commonwealth Scientific and Industrial Research Organisation
CST	Context Space Theory
CSTO	Context Space Theory Ontology
DL	Description Language
DST	Dempster-Shafer Theory
ECSTRA	Enhanced Context Spaces Theory-based Reasoning Architecture
FOL	First Order Logic
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JVM	Java Virtual Machine
MVC	Model-View-Controller
O-DF	Open Data Format
O-MI	Open Messaging Interface
OWL	Web Ontology Language
RDF	Resource Description Format
RDFS	RDF Schema
RFID	Radio-Frequency Identification
SAN	Semantic Actuator Network
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SQWRL	Semantic Query-Enhanced Web Rule Language
SSN	Semantic Sensor Network
STO	Situation Theory Ontology
SWRL	Semantic Web Rule Language
UML	Unified Modeling Language
W3C	World Wide Web Consortium
XML	Extensible Markup Language

1 Introduction

With the rise of the Internet of Things (IoT) everyday life becomes more and more supported by pervasive computing systems. The IoT paradigm aims to integrate computing technology gracefully everywhere and to make it available at any time in a non-intrusive way. According to Gartner the number of devices connected to the Internet of Things will rise to 6.4 billion in 2016, and is predicted to reach 20.8 billion in 2020 [1]. This trend is motivated by the huge expectations of the capabilities of IoT-based systems and thus enterprises, public sectors as well as private customers get involved. The huge amount of generated and accessible data enables the development of supportive computing systems in industry, environment and society, eventually creating a "better world for human beings" [2].

Research focuses on ways to collect, model, reason about and distribution of context, i.e. the data provided by the increasing amount of sensors. This addresses for example technologies regarding network and storage for the collection of data and semantic approaches like machine learning for interpretation. Open challenges exist in these fields and are addressed in research up to today [2].

Context awareness has become an established research field in computer science by being a core feature of ubiquitous and pervasive computing systems. With the evolvement of the IoT paradigm, the significance of the capabilities of context aware approaches have increased. Situation awareness, a higher level of abstraction of context, allows systems to understand their environment thoroughly and is thus enabling the development of systems that are more beneficial to human beings without their interaction.

Open challenges include the development of domain-independent approaches which ease the engineering effort to apply these to different domains. This is a difficult task because of the fundamental differences in the nature of domains, applications and corresponding problems, as well as the complexity of the systems and the diversity of available approaches in collection, analysis and interpretation of data.

1.1 Motivation

The United Nation have defined the Sustainable Development Goals in form of a UN Resolution for a global transformation by 2030. These include for example ending hunger, making cities sustainable and stopping climate change, as illustrated in Figure 1.1.

The related UN report on trends in sustainable consumption and production [3] states that "fundamental changes in the way societies produce and consume are indispensable for

achieving global sustainable development" and further agrees that "information and communication technologies (ICT) are bound to play an increasingly prominent role as a key enabler of renewed and sustainable growth". Particularly mentioned are the positive impacts ICT systems can provide on resource and energy efficiency, and on reductions in waste.

Figure 1.1 The UN Chart of Global Sustainability Goals[1]

According to the Ellen MacArthur Foundation and the World Economic Forum, the Internet of Things will enable the transition from today's linear to a future circular economy, unlocking its potentials [4]. The report states that systems based on the generated data of connected intelligent devices offer sustainable opportunities in all parts of society.

Our linear economy relies on finite resources and even increased efficiency will not prevent the natural stocks to run out eventually. The concept of circular economy proposes a fundamental change. The three key principles are firstly, preserving natural stocks and to improve the use of renewable resources, secondly, circulation of products, components as well as materials and thirdly, fostering the system effectiveness by minimising negative externalities. The Internet of Things is able to provide the overarching information about the location, condition and availability of assets in the economy. These have been identified as key factors to enable the extension of use cycles, increasing utilization, further looping and regeneration of the assets in the economy.

That the depletion of natural resources is an issue that needs to be addressed today is also acknowledged by the CSIRO report on global megatrends [5]. Climate change, population and economic growth are putting pressure and demand on natural mineral, energy, water and food resources. Resources need to be accessed in a more sustainable and efficient way to ensure future supply.

[1] Screenshot taken from https://sustainabledevelopment.un.org/sdgs - Accessed 09/05/2016

Pervasive systems have thus been developed which can be seen as ICT enabling technology to provide sustainable solutions, e.g. in form of smart energy systems (for lighting, heating/cooling, displays etc.), environmental impact monitoring, traffic flow scheduling, decision support systems and many more.

This thesis is motivated by the opportunities that IoT-based applications represent, as described above, and to improve its capabilities in order to contribute to the sustainability goals and solve related issues. Solving problems in the domain of situation aware systems helps to develop sustainable solutions provided by pervasive systems. Moreover, the scenario chosen for this thesis is explicitly selected to demonstrate how the developed ICT-/IoT-based solution can be applied to contribute to the sustainability goals by reducing food waste.

1.2 Scenario

The scenario described in this section aims to reduce food waste in smart homes. It will be considered throughout this thesis to illustrate and validate the benefits of the approach, in particular by providing a solution contributing to sustainability.

As previously mentioned, food security is one of the global challenges to be solved in our future society. Reducing food waste and food loss are major roles in achieving food security. It is estimated that on a global scale up to 50% of the food produced is wasted along the whole value chain, i.e. in production, handling and storage, processing and packaging, distribution and market as well as consumption [6]. The impact of the different stages in the value chain on the overall waste varies between different regions in the world [7]. However, estimates stated from different organisations, e.g. by the Food and Agriculture Organization of the United Nations (FAO) [8] and the Organisation for Economic Co-operation and Development (OECD) [9], indicate that production and consumption have the biggest impact on food loss.

The opportunities to improve food security at consumer side are shown by the estimate that 30-50% of the food quantity that reaches supermarkets is eventually thrown away by consumers at home [7]. The proposal of this work is based on a situation aware pervasive system in a smart home neighbourhood. The concept of the scenario is visualised in Figure 1.2.

Awareness about the available food items via smart fridges and shelves enables the system to propose the most efficient use of these products, e.g. by proposing corresponding recipes. With the consideration of the expiration of and the demand for food the system is able to assist users to consume the right products at the right time to avoid food waste. Moreover, by expanding the awareness over a neighbourhood, users can be encouraged to share expiring

food items if they are needed by other users nearby. It could be further imagined that the system automatically orders new food items from local stores to replace shared items.

Figure 1.2 Scenario in Smart Homes to Reduce Food Waste

Beyond the ethical problems of food waste, it also impacts the other two pillars of sustainability. According to FAO [8], food waste has a significant impact on the environment. For example, produced but not consumed food corresponds to 28% of agricultural land for crops. 250km^3 of water are wasted by growing these crops which corresponds to the world household water needs. Wasted food emits 3.3GT of greenhouse gases per year. The economic consequences to producers of food waste are estimated to be around 750 billion USD annually. By incorporating these factors when recommending the use of particular food items, awareness of the users about the implications of food waste can be risen and motivate behavioural change regarding the consumption of food.

1.3 Problem Definition

This thesis is concerned with the general applicability and reasoning capabilities of situation aware approaches from a holistic point of view. Interpretation of data is based on external knowledge. Common approaches either make use of explicitly specified knowledge or use learning techniques that require given training data as an input. This initial process is complex, time-consuming and error-prone.

On the one hand, specification-based techniques require experts of a certain domain who need to integrate their knowledge in a way which the application is able to process. Mistakes in the specification lead to inconsistency in the context model and will cause errors in the situation reasoning process. On the other hand, gathering the training data set for learning-based techniques may be difficult, the set may not cover all possible cases or training data may not be available at all. Errors or lacks in the training data will lead to errors or unknown cases in the final application [10].

While the research community has put a lot of effort to propose general approaches for context awareness, it is more difficult to provide a general way for a situation aware approach. This is caused by the higher level of abstraction, which depends to a greater extend on specific domain or application knowledge and requires a thorough foundation for knowledge integration.

Moreover, not solely knowledge about the situations but also knowledge about other parts of the system's environment needs to be specified. This includes for example the involved sensors and how their values are related to situational aspects. The identified challenge is to combine all these mentioned aspects in one approach to have a complete and easily applicable framework for the development of situation aware applications.

1.3.1 Research Question and Objective

Based on the presented context and outlined problems, the research question is phrased as follows.

Research Question

What are the required knowledge assets of a situation aware approach and how can they be integrated in a standardized, reusable way?

The identified knowledge assets are concerned with the specifications about situations and the system's environment that are absolutely necessary for a situation aware approach and are thus required independent of, but have to be specified for each, domain or application.

Correspondingly, the objective of the thesis is defined as follows.

Research Objective

To develop a general applicable framework which provides enhanced reasoning capabilities, a formal integration of all required knowledge assets and standardized embedment in the IoT environment to enable a more efficient development of situation aware applications and to avoid typical specification problems.

The goal is to overcome the intense and error-prone work of knowledge specification and create a formal way that encourages the reuse of existing knowledge assets. This would not only improve the initial development process of situation aware applications but eases the enhancement and maintenance of existing applications over time. This needs to be wrapped around an approach with general reasoning capabilities that meet the challenging requirements of domain-independent situation awareness.

1.3.2 Research Methodology

A literature review, focusing on the theoretical approach to situation awareness and investigating related work, will be conducted to answer the above presented research question. The results will be taken into account for the design process of the framework that is part of the research objective. Refined requirements for each mentioned characteristic of the framework will be defined based on the discussion of the literature study.

An iterative process of requirements analysis, framework design and system implementation ensures that the theoretical and practical issues are aligned with each other, furthermore allowing a refined framework design during the project work. The approach will be validated with a selected use case after the iterative process ended, i.e. when the practical implementation and the theoretical approach are sound. However, early prototypes for preliminary validation will be developed in order to detect and avoid major design issues.

1.4 Project Scope

This thesis is concerned with a general framework for situation awareness and closing the gaps to external dependencies. To provide a general approach, existing standards and solutions will be applied and adopted if they are feasible.

The proof-of-concept of the framework will be based on a simulated environment. It is not the objective of the project to solve for example physical challenges of sensing context.

1.5 Thesis Structure

This thesis is structured as follows.

Chapter 2 - Background and Related Work. Chapter 2 introduces background knowledge and presents the state of the art of situation aware approaches. The background knowledge is taken into account to formulate requirements for situation aware systems and comparing related approaches. The conclusion of this study forms the foundation for the proposed framework in this thesis.

Chapter 3 - Ontology-based Generation of Situation Spaces. Motivated by the investigation in chapter 2, chapter 3 proposes to add an ontological knowledge base to Context Space Theory (CST), realised with the Situation Theory Ontology (STO). This chapter maps the different concepts of the theories and shows how CST can be enhanced by STO.

Chapter 4 - Framework Design and System Implementation. Based on the selected approaches and the proposed enhancement of chapter 3, the design and implementation of a situation aware framework will be described in chapter 4. Whereas previous discussions focused on theoretical solutions, this chapter considers practical aspects.

Chapter 5 - Use Case: Food Sharing Neighbourhood. This chapter introduces a use case as a proof-of-concept of the proposed framework. It demonstrates in more detail how the system can be used to develop a situation aware application. The selected use case is a web application which gives recommendations about the best usage of food items in a smart neighbourhood.

Chapter 6 - Discussion of Results. This chapter discusses the results regarding the theoretical analysis from chapter 3, the implementation from chapter 4 and the use case in chapter 5. The discussion includes a performance analysis for relevant features.

Chapter 7 - Conclusion and Future Work. The last chapter concludes the thesis and presents future challenges.

2 Background and Related Work

This chapter presents the background theory and related work of the thesis. First of all the fundamentals of context and situation awareness will be described. Afterwards theoretical techniques for situation identification will be explained. Based on this background knowledge existing approaches to situation awareness that apply these techniques will be evaluated regarding selected criteria. A conclusion will be drawn that will form the foundation for the proposed system of this thesis.

2.1 Fundamentals of Context and Situation Awareness

The fundamental feature of applications in pervasive computing and the Internet of Things is the concept of context awareness. Context is defined as “any information that can be used to characterize the situation of an entity” [11]. A system becomes context-aware if it offers information or services to the user which are related to the user's tasks by making use of context. Thus computing systems that are aware of their environment can provide services of higher value to humans [11]. Context usually originates from data produced by sensors. Examples for contextual information are current temperature, battery status or user’s emotion.

A context model, also referred to as context representation, describes the relevant aspects of the context, i.e. accessible assets from sensors, applications and users, regarding a certain task or application in a formal and general way [12]. There are a lot of different possibilities to model context. Context modelling techniques can be classified into key-value, mark-up schemes, graphical, object, logic and ontology based modelling [2]. Each of the approaches differ in complexity, accuracy and applicability of context representation.

Context reasoning, or context inference, means to "deduce new knowledge, based on the available context data" [13]. Context models form the basis for application independent context reasoning and the context reasoning capabilities depend on the used context model technique. Context reasoning can be divided into three steps. During the first phase, preprocessing, the data is cleaned of inaccurate values and missing values are handled. Afterwards data from multiple sensors are combined during the sensor data fusion phase. The last step is context inference in which new high-level context information is inferred from the low-level context data [2, 14].

Context prediction describes the tasks of inferring future context information by observing the progression of a context time series [15]. In other words, past and present context information is linked to future context [16]. Predicted knowledge enables pro-activity of applications. For example, applications could have more time to prepare and present services [17].

Situations represent a higher level of abstraction than context, as depicted in Figure 2.1. A situation can be defined as an "external semantic interpretation of sensor data" [18] by linking contexts to a descriptive name. Meaning needs be assigned based on the correlations between the relevant collected contexts to identify a situation. These correlations represented in a logical expression form the logical specification of a situation [10]. Situations, thus, can be seen as "logically aggregated pieces of context" [19].

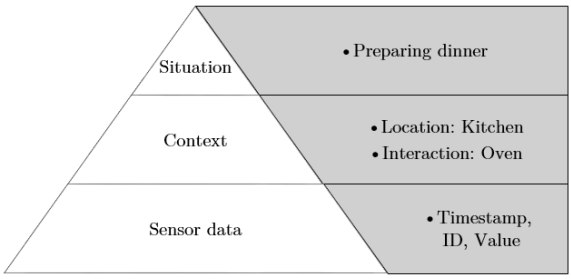

Figure 2.1 Levels of Abstractions in Pervasive Computing, based on [10] and [20]

Situation aware applications are triggered by the descriptive name of identified occurring situations. Situation awareness is desirable because it provides a simple representation of a complex set of sensor data to applications, hiding the complexity and related issues about noise, inference and uncertainty of sensor data [10]. This abstraction is useful for effective development of applications that understand and react to their environment [21].

2.2 Situation Identification Techniques

Situation identification in pervasive computing, also referred to as situation determination, situation recognition or situation inference, deals with the following three issues: The logical representation to define the logical specification of situations, how it is formed to allow specifications by an expert or machine learning and, lastly, situation reasoning, i.e. inferring situations from imperfect sensor data [10].

This section gives an overview of common and relevant techniques that can be applied to solve the above mentioned issues. The discussion is focused on a high-level view of the techniques which is sufficient to evaluate their eligibility for situation awareness approaches later on. The discussion is based on the review by Ye et al. [10].

2.2.1 Specification-based Techniques

As mentioned previously, situation aware applications rely on external knowledge to interpret the sensor data. In specification-based approaches this expert knowledge is first represented in logic rules. Reasoning engines applied on these rules then infer the situations, based on the sensor data [10].

Formal Logic. A popular way to represent the knowledge about situations is to use logical predicates. Logic based models provide a strong formalisation to represent the logical specification of situations. The reasoning is then applied in a rule-based way, whereas rules are statements that define the relation between facts [22]. The underlying concept of approaches representing situations with formal logic is that "knowledge about situations can be modularized or discretized" [23]. Reasoning capabilities of this approach include the verification of integrity and consistency of the situation specification and systems can be extended to reason about more situations later on [10].

Fuzzy Logic. This technique, originally presented in [24], is used in the field of situation identification to model imprecise knowledge so that vague information can be expressed. Fuzziness handles uncertainty not by using a formal representation with probability but rather by focusing on the natural ambiguity of an event itself [19]. In fuzzy logic sensor data is linked to linguistic variables by membership functions. For example, a set or range of numerical values can be mapped to a certain term or fuzzy variable. The rule-based reasoning then infers a membership degree between 0 and 1 for each fuzzy set, since the conditions for the sets may overlap [25].

The eventual result of the reasoning process thus will provide a degree of belief of occurring situations [10]. It is argued in [26] that this approach would be rather inappropriate for situation awareness because the rule-based reasoning is very dependent on the domain and problem. Furthermore equal beliefs for contradictory situations could be calculated on which it would not be possible to react properly for the system.

Ontologies. The term ontology originates from philosophy and is defined as an "explicit specification of a conceptualization" [27]. Ontologies are applied in various research domains and are used in pervasive computing as a formal representation for sensor data, context, as well as situations. For situation identification, ontologies can be seen as a way to capture domain knowledge with a well-structured terminology which is readable to humans and machines [10, 28]. Ontological modelling includes the concepts of classes, instances, attributes and relations [29].

Three kinds of ontologies can be differentiated. Generic ontologies, also referred to as upper ontologies, describe general concepts. Domain ontologies specify concepts of a certain domain and application ontologies represent application specific knowledge [30]. Ontologies are a popular technique for situation awareness because of the rich semantics and expressiveness. Additionally, ontological reasoners can check automatically the consistency and infer new knowledge based on a given ontology [10].

Dempster-Shafer Theory. This mathematical theory of evidence, presented in [31], allows to calculate the likelihood of events, e.g. a situation, with information from different

evidence sources. Mass functions specify the distribution of belief across the frame of discernment, the set of possible hypotheses. The combination rule merges evidence from different sources [32].

Dempster-Shafer Theory allows to assign beliefs to sets or intervals, enabling reasoning even if the beliefs are only partially known. This makes the technique very powerful in terms of handling uncertainty and belief distribution. However, it requires a lot of expert knowledge to create an evidential network - i.e. which context information can be inferred from which sensor data and which situation can be inferred from which contexts - and domain experts need to define the degree of belief for all evidences [10, 33].

2.2.2 Learning-based Techniques

In today's pervasive computing and IoT environments a huge amount of sensor data is generated which may contain noise. Handling the noise on a specification-based way is impractical, instead machine learning techniques are used to identify situations based on the sensor data. Learning-based techniques rely on a large set of training data to achieve proper results [10].

Bayesian Techniques. Bayesian classification frameworks are based on Bayes' theorem. Bayes' theorem is used to update the probability of a hypothesis - i.e. a situation occurring - if a new evidence is given. A prior probability is assigned to both an evidence to support a hypothesis and to the hypothesis itself. With the posterior probability of the supportive evidence conditioned on the hypothesis the theorem updates the probability of the hypothesis [33, 34].

Naïve Bayes assumes that all features characterising an evidence are statistically independent. With this premise the posterior probability can be calculated with reduced complexity by multiplying the probability for each feature of the evidence conditioned on the hypothesis [34]. This technique relies on a-priori knowledge about the probabilities of the hypothesis, if the probability for a feature of an evidence is missing in the training data the probability will be zero if it occurs later [10, 22].

Bayesian networks are used in case dependencies between features characterising an evidence exist. A Bayesian network is a directed acyclic graph, whereas nodes represent random variables and edges represent casual influence [33]. Each root node is associated with a prior probability. In a qualitative Bayesian network each non-root node is associated with a conditional probability distribution, in a quantitative Bayesian network with a conditional probability table, which indicate the influence of each parent of the node. The relationships are usually defined by domain experts. The process of inference and belief update is similar to Naïve Bayes [10, 33]. Bayesian networks have the same downside as Naïve Bayes in terms of unavailability of prior probability [10].

Markov Models. This technique is a generative probabilistic model based on Markov chains. Markov chains are sequences of random variables, describing conditional probabilities for transitions of the state of the system.

In Hidden Markov Models each state is composed of a hidden and an observable state [35]. A hidden variable at a time t depends only on the previous hidden variable at $t-1$, whereas an observable variable at a time t depends only on the hidden variable at time t. Based on this the model can be specified with three probability distributions, prior probability for initial states, state transition probability and the probability of a hidden state inferring an observable state [36]. For a HMM, observations need to be specified as training data. Problems with default HMM include that the probability of an event declines exponentially over time intervals and that hierarchical relations cannot be modelled. Thus, this approach was mainly applied for activity recognition approaches, whereas situations usually require a more complex specification of structural aspects [10].

Neural Networks. In a neural network artificial neurons are linked together according to a specific architecture. A neural classifier is based on an input and output layer. The mapping between these two is done by a hidden layer, a composition of activation functions which learn through training data [10].

The accuracy of neural networks depends strongly on the training data set. Neural networks are as well usually applied for activity recognition. If the mapping is composed of a lot of features and linked neurons the computations become complex [10].

2.3 State of the Art in Situation Aware Approaches

The situation identification techniques explained in the previous section have often been applied for situation aware applications. However, the developed solutions are usually dependent on the domain or the application. This section aims to give a survey about existing approaches to situation awareness that aim to be domain- and application-independent. Before, related surveys will be discussed.

A lot of efforts in the research community have been done to develop context and situation aware approaches which led to a lot of different kinds of solutions. Related surveys can be found which aim to give an overview and to evaluate existing approaches and projects. These are discussed in the following in order to select relevant criteria for a comparison of related situation awareness approaches later in this chapter.

In [10] the authors provide a comprehensive review about the state of the art of situation identification techniques and mention situation awareness approaches. Even though the focus lies more on the techniques themselves and that there is no specified framework to evaluate the approaches, the discussion is very helpful because the underlying techniques

(as presented in 2.2) applied by an approach give an indication about its capabilities. Findings of this survey include aspects about different levels of abstraction, uncertainty, temporality, complexity, knowledge incorporation and derivation, engineering effort and effect of sensors.

Paper [37] provides a survey about situation awareness approaches that are based on ontologies. It makes a clear distinction between situation and context related concepts, having distinct criteria for both to compare the different approaches. However, it solely focuses on ontologies and does not include a comparison with other techniques. Nonetheless, interesting facts can be concluded. Firstly, ontologies designed for context awareness are not sufficient for situation awareness because these do not support a complete formal representation of situations. Secondly, required aspects in ontologies for situation awareness are identified: space and time, roles, situation types and situations should be represented as objects.

Survey [2] gives an extensive overview over existing research efforts in the field, however, from a context awareness point of view. The focus lies in investigating the capabilities of the context reasoning approaches without evaluating the domain independence and general applicability. A similar effort, though less extensive, is done in [38]. Nonetheless, some of the selected criteria can be adopted for the comparison of situation aware approaches, e.g. fundamental reasoning technique, knowledge management, sensor integration and real-time processing. They also show that there are more aspects that go beyond the scope of this thesis, like security and privacy.

The discussion in [39] is based on classifications of context modelling techniques. It evaluates the classes of context models in terms of reasoning capability aspects, but does not include situational aspects. Considered criteria include distributed composition, partial validation, richness and quality of information, uncertainty and formality. These are relevant for situation models, too. The conclusion is that only object oriented and ontology based models are appropriate to model all the aforementioned aspects and presents ontologies as the most promising approach. However, the survey neglects that different techniques could be combined.

Context modelling techniques are evaluated in a similar way in survey [20]. This survey goes further by investigating reasoning techniques with differentiation between low-level and high-level context, i.e. situations. The requirements chosen in this survey are heterogeneity and mobility, relationships and dependencies, timeliness, uncertainty reasoning, usability and performance. Modelling approaches are categorized into object-role, spatial and ontological. The conclusion of this paper is that only a hybrid context model is capable to

satisfy all requirements. Each type of model has its strengths, e.g. object-role in usability, spatial in mobility and ontological in relationships, and its weaknesses. The paper illustrates the benefits of a hybrid model by proposing a multilayer framework. Not only the selected requirements but also the conclusions are helpful for the further discussion throughout this chapter.

Another comparison of ontologies for context awareness has been done in [30], the criteria considered focus only on context related aspects like location, person, time etc. and do not include higher abstraction concepts. However, interesting points like discovery, interaction design and essential infrastructure are taken into account and discussed for existing ontologies. The survey states that ontologies are a key concept to "simplify the creation, composition, and analysis of pervasive computing systems" and that after all it is only a part of the whole system since the ontology does not define how the information will be processed.

In conclusion, surveys often focus on a selected technique or do not consider situation aware aspects sufficiently. Despite that, some criteria can be adopted for situation aware aspects and conclusions of single techniques integrated in the following overall comparison of situation aware approaches.

2.3.1 Requirements

This section aims to discuss the high-level requirements of situation aware systems nowadays in order to evaluate existing approaches to situation awareness. The selected criteria are merging requirements regarding modelling, development, reasoning, external dependencies and functional requirements.

The criteria were selected by taking into account the key components of a situation aware system that have been identified by Endsley et al. [40] in 1995, and by considering the previous discussions of surveys. Figure 2.2 shows the model of situation awareness by Endsley which served as a source of inspiration to identify modern requirements and components for situation awareness.

Situation. The first category defined is concerned with the abstraction of contextual information. It is mainly concerned with the comprehension of current situation feature. Based on the survey this aspect is important because context aware approaches may not support the higher level abstraction of situational aspects. These modelling requirements can be seen as preconceptions which will form the basis for comprehension of situations.

Knowledge. The integration of environmental specifications and semantic aspects that are required for the system to be deployed are covered by this category. Closest related to the preconceptions, but also related to abilities and experiences mentioned in the situation

awareness model, it further includes the practical engineering process to specify and reuse relevant knowledge in an automated way.

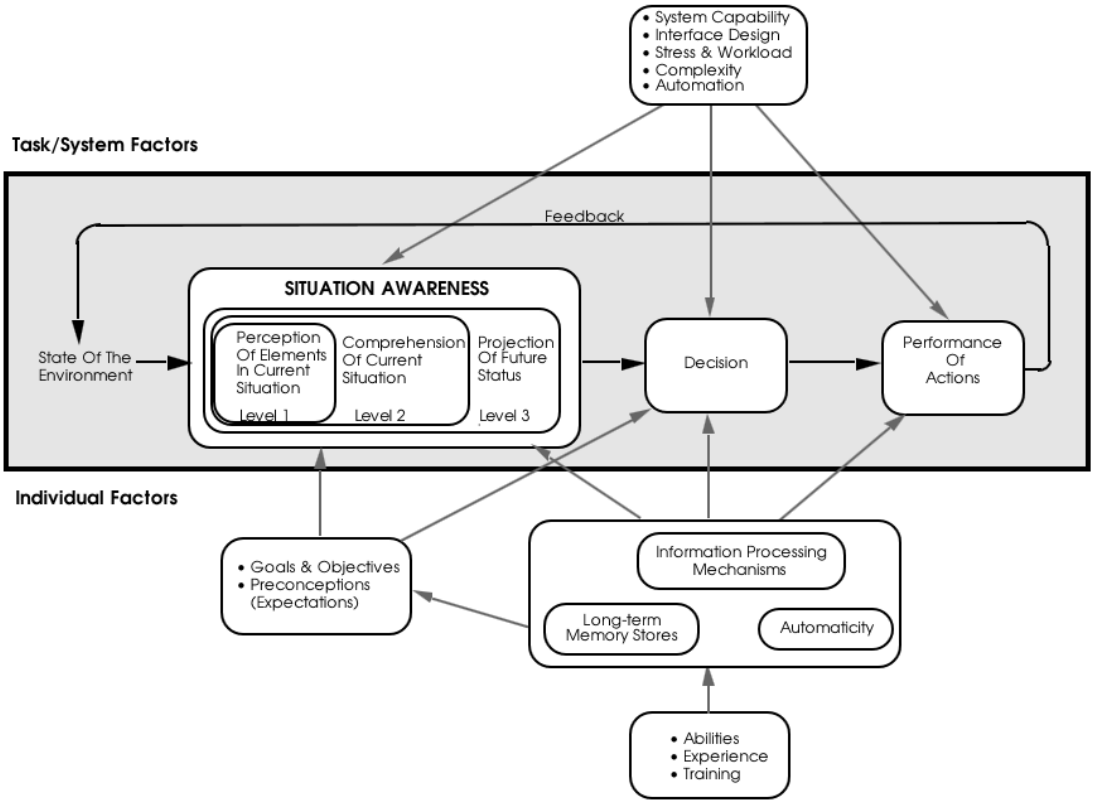

Figure 2.2 Model of Situation Awareness [41]

Reasoning. This category addresses the inference capabilities of the approach, i.e. the processing and used mechanisms to comprehend current and predict future situation occurrences. As it was already shown in section 2.2, the techniques have to cope with various issues like uncertainty, temporality aspects, etc.

Application. Lastly, this group of requirements addresses the challenges of implementation. This refers to sensor access to perceive the current state and actuators to react to reasoning results. This correspond to state of the environment and decision in the situation aware model. Considering the global IoT trend and the huge amount of available sensor data, the performance and applicability of situation aware systems became another very important aspect.

The deduced criteria for a general situation aware approach are summarized in Table 2.1. Each group of requirement is decomposed into three criteria.

This concludes the analysis of criteria for approaches to situation awareness. It will be referred to these requirements to evaluate related approaches and the proposal of this thesis.

Criteria		Description
Situation	*Space and Time*	Representation of space and time related aspects of context and situations.
	Roles of Objects and Situation Types	Support of general roles for objects to understand the relation of them in situations and types of situation to enable general modelling.
	Relations	Considering relations between objects and situations.
Knowledge	*Situational Knowledge*	Capability of integrating situational knowledge for further reasoning.
	System Knowledge	Formal specification of application dependent setup and its relation to situational semantics.
	Knowledge Reuse / Sharing	Capability of reusing and sharing already defined knowledge.
Reasoning	*Overall Capability and Universal Applicability*	Degree of reasoning capabilities of the approach in terms of situation inference, considering the degree of domain independence and general applicability.
	Uncertainty and Temporality	Considering uncertainty in sensor data and situation specifications; as well as considering evolution over time.
	Prediction	Capability of predicting situations.
Application	*Sensor Data Acquisition*	Considering the integration of sensed information, especially regarding IoT-related aspects.
	Actuators	Considering the formal integration of actuators to react to reasoning results.
	Performance and Applicability	The performance and applicability for application development of the implemented approach.

Table 2.1 Evaluation Framework for Situation Awareness Approaches

2.3.2 Discussion

In this section existing approaches to situation awareness will be introduced and discussed. The comparison according to the criteria discussed in the previous section can be found in the following section.

Context Broker Architecture (CoBrA). The Context Broker Architecture was developed to provide a general solution to support context-aware systems [42]. It is based on an upper ontology, which is called Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) [43], and rule-based reasoning. The context broker in this architecture is a single-agent who manages a model of context by inferring knowledge from SOUPA-based ontologies, accessing sensor data from various sources and reasoning about the context. How these components interact is visualized in Figure 2.3.

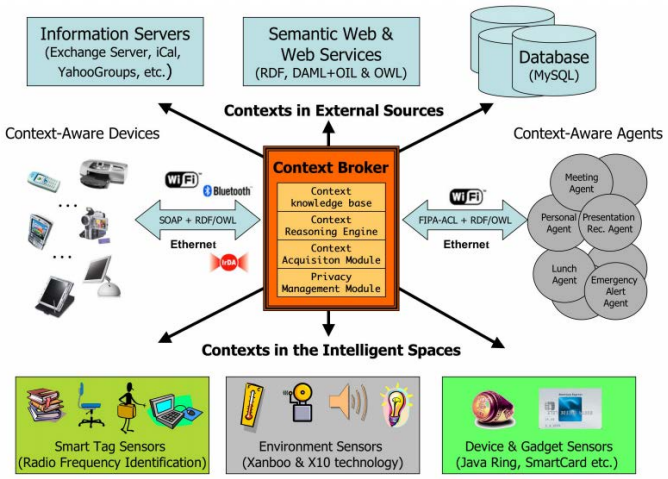

Figure 2.3 Context Broker Architecture (CoBrA) [42]

The SOUPA ontology forms the core of the architecture and thus provides a knowledge base about the semantic context of the situation aware system. The ontology is given in Figure 2.4. Objects are defined as instances of classes like *Person*, *Agent* or *Event*.

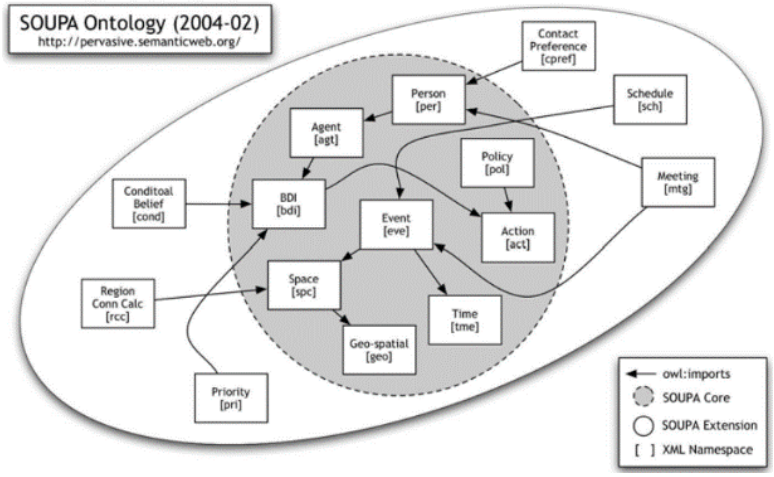

Figure 2.4 Standard Ontology for Ubiquitous and Pervasive Applications [43]

This approach was designed for context, not situation awareness. Relations and attributes are not formally defined and situations itself are not represented as objects, so there is no differentiation of situation types. Relations on a higher abstraction level are thus not supported. It demonstrates that context aware approaches cannot be adopted for situation awareness. It does consider the integration of sensed data from various sources but it does not consider uncertainty in sensor data and modelling of the system's environment.

Context Space Theory (CST). The Context Space Theory was developed to provide a general context model with a rich theoretical foundation. Approaches to context awareness are limited to the underlying, general theory of the used techniques. With a model that contains context as a central concept these limitations vanish [22]. It was also designed to "enable context awareness in [a] clear and insightful way" [44].

Context Attributes, which are measurable properties usually provided by sensors, form the dimensions of the *Application Space*. Real-life situations are represented as subspaces of the application space, named *Situation Spaces*. A *Context State* describes a point that moves through the application space depending on the current values of a corresponding set of context attribute values over time. If the context state lies in the subspace of a situation, this situation is occurring [44]. This concept is depicted in Figure 2.5.

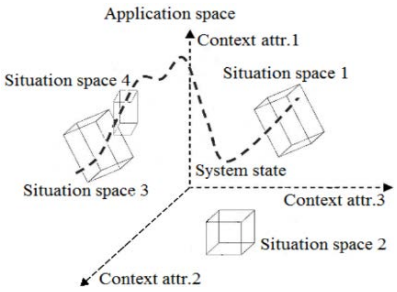

Figure 2.5 Context Space Theory [45]

Based on this representation, techniques for reasoning under sensor uncertainty and unreliability have been developed and are considered as a part of the Context Space Theory [46]. Specification- and learning-based techniques like Bayesian reasoning or Dempster-Shafer can be applied to determine which situation is occurring. Also algebraic operations and logic-based reasoning are used to reason about situations [22]. CST also supports methods for context prediction [47].

Situation Awareness Assistant (SAWA). SAWA is based on the Core SAW Ontology. The Core SAW Ontology was developed to represent "a theory of the world" [48] to achieve situation awareness (SAW). It was designed to represent objects, relations and any reasonable evolution of them in an economical way. The UML diagram of the ontology is shown in Figure 2.6. A situation is represented as a set of *Entities* with *Attributes*, *Goals* and *Relations*. Attributes and Relations are attached to a class of *PropertyValues* to allow evolving values over time. *EventNotices* represent changes of the real world perceived by sensor data that affects the property values.

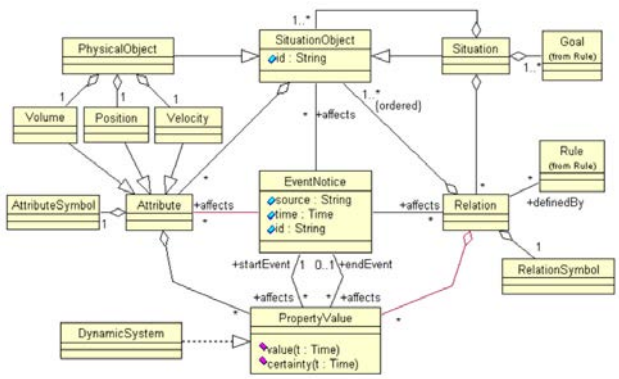

Figure 2.6 Core SAW Ontology [48]

SAWA adds a consistency check and SWRL rules to knowledge representation in form of the ontology to form a pre-runtime knowledge management component. The runtime system is comprised of an Event Management Component (EMC), a Situation Management Component (SMC), a Relation Monitoring Agent (RMA) and a Triple Store (TDB) [49]. The relation between the components is shown in Figure 2.7.

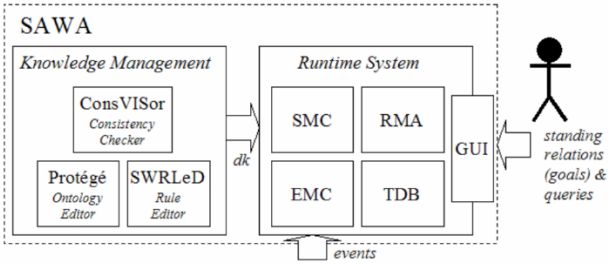

Figure 2.7 SAWA Architecture [49]

By focusing on relations of objects and considering their evolution over time, this approach has its strength in considering temporality. Also knowledge management regarding semantic specification is given. A formal integration of sensor data and system knowledge is missing. The reasoning is limited to logical rules and it does not consider uncertainty.

Hierarchical Situation Modelling. The Hierarchical Situation Modelling approach proposes an OWL DL based top-level Situation Ontology and reasoning based on the First-Order Logic (FOL) [50]. The ontology is divided into a context and a situation layer and it differentiates between atomic and composite situations. However, as also stated in [37], there is no clear distinction between these two layers and basic concepts like objects and relations are not properly modelled. In Figure 2.8 the fuzziness between the layers as well the lack of basic modelling concepts for context and situations can be observed.

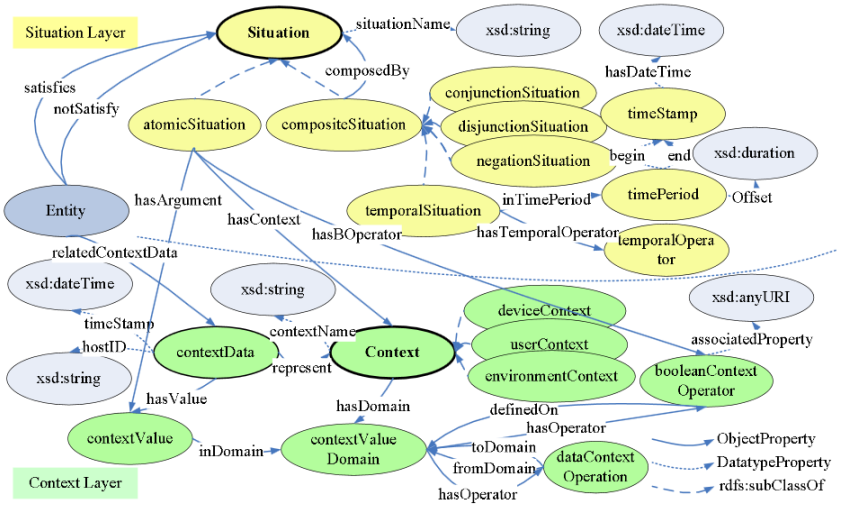

Figure 2.8 Situation Ontology [50]

The OWL axioms are then transformed into FOL rules with transformation rules for each situation type. Whereas this approach does not provide a promising knowledge base or in general integration of other relevant system specifications, it achieves to improve the performance compared to a plain ontological approach.

Situation Lattice. This approach applies the lattice theory for situation inference [18]. Lattice theory is an algebra "concerned with the properties of a single undefined binary relation $\leqq$" [51]. The characteristics of situations described are dependencies and generalisation. A lattice is a partially unordered set. A situation lattice is defined as $L = (S, \leq)$, which represents the generalisation of a set of situations. Dependencies of situations are recognized based upward down on the ordered set. A situation lattice can look like Figure 2.9. It is based on a uniquely true situation at the top and a uniquely false situation at the bottom.

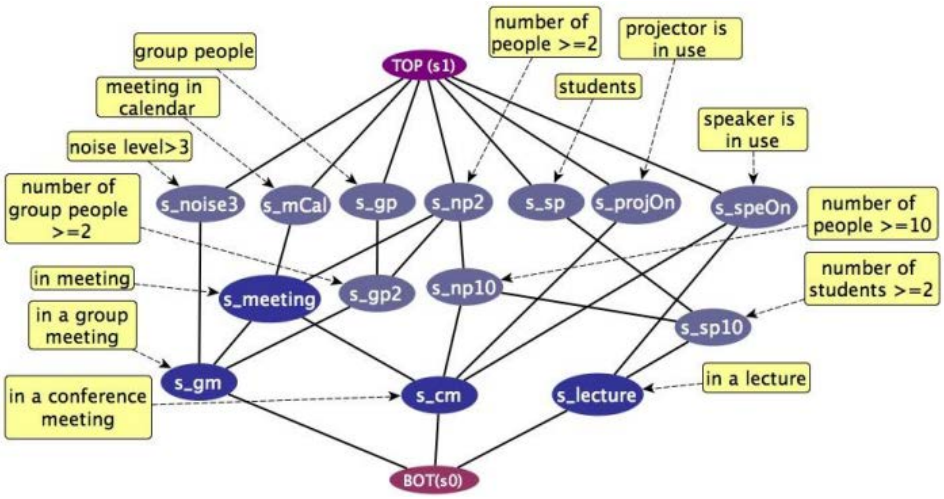

Figure 2.9 Situation Lattice [18]

A situation lattice can be checked on consistency and integrity. Situations are identified through forward chaining, starting from the acquired context. Further an approach to resolve uncertainties (incomplete, imprecise, conflicting, incorrect and out-of-date sensor data) is proposed.

This approach offers a strong foundation for situation modelling and validation. Furthermore it can handle uncertainty well. However, modelling of situation lattices can become a complex task and big lattices will affect the performance significantly. The flexible acquisition of sensor data is also acknowledged as an open challenge.

Situational Context Ontology. The approach of the Situational Context Ontology aims to enhance the ontological specification with fuzzy logic to model vagueness in knowledge about situation types and to overcome imprecise sensor readings [19]. The developed ontology is depicted in Figure 2.10. It can be observed that the ontology does not have

situations as a central concept. Instead, *context* information is attached to *persons* which itself are involved in *situations*.

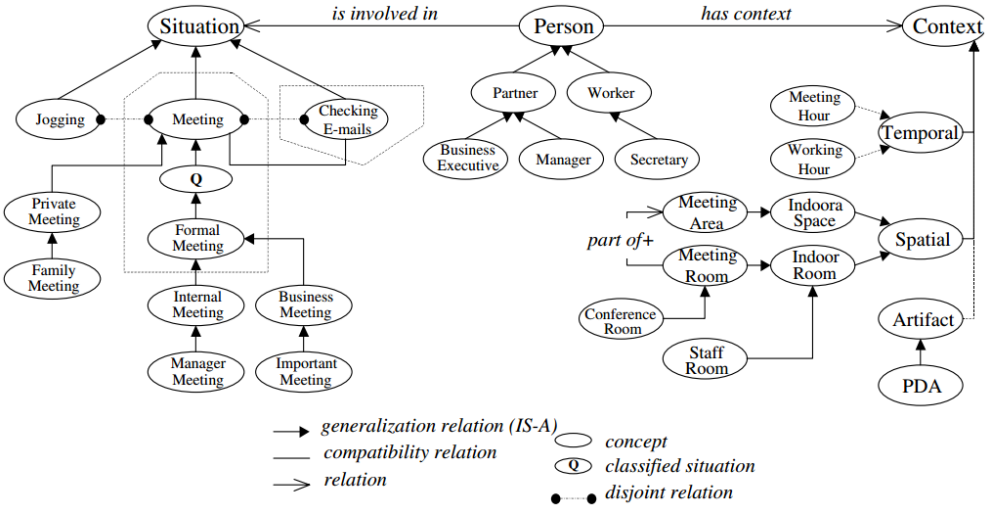

Figure 2.10 Situational Context Ontology [19]

Fuzzy logic is applied by measuring the similarity of situational context, calculating a degree of involvement with approximate reasoning and comparing it to fuzzy boundaries. Furthermore it considers the historical context of the user. Thus the advantages of this approach lie in handling uncertainty and temporality. However, the situation model is not very thorough and the embedded environment of the system is not considered.

Activity Recognition in a Home Setting. This approach applies both HMM and Conditional Random Fields (CRF) to identify activities in a home setting and conducts experiments for comparison [36]. The experiments show that both techniques can be applied for activity recognition. However, even though activity recognition is a high-level inference, it does not consider rich temporal and other structural aspects of situations, which distinguishes it from situation recognition.

Thus, this approach lacks a thorough foundation for situation modelling. Furthermore the reasoning results strongly depend on the selected observation representation which implies the difficulty to find a general model that can be reapplied. However, being based on learning techniques, this approach is flexible in adjusting itself through the learning data, which enables individual accurate recognition results for different setups.

Activity Recognition Based on Acceleration. This approach employs neural network techniques to learn human activities from acceleration data [52]. Even though this approach is limited to recognise activities based on acceleration it is included in the discussion because the model for neural classification is always dependent on specific input data.

The design proposal appear promising in general application for acceleration-based activity recognition, however, performance issues may appear depending on the amount of included features. This approach brings along the advantages of learning-based techniques but also the disadvantages of operating on an abstraction level of activities instead situations.

Situation Theory Ontology (STO). The Situation Theory Ontology is based on an interpretation of Barwise' situation semantics, referred to as Situation Theory [53]. Information about situations is represented with *infons*, which represent a *relation* over *n objects*. Each infon is furthermore assigned to a *polarity* to define if the objects stand in the relation or not (true or false). Infons can be seen as particular facts of a real-life situation. Real-life situations can support different infons and infons can entail each other, as shown in Figure 2.11. It represents how an agent uses logical inference to understand facts about situations. The objects part of a relation can be broken down into different types, e.g. attributes, individuals or situations [54].

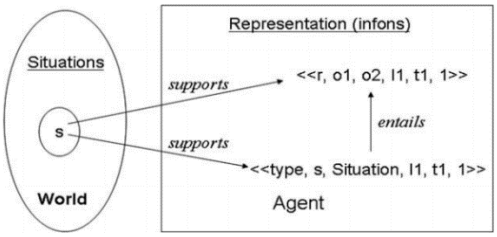

Figure 2.11 Situation Theory [54]

Figure 2.12 shows a simplified view of how this concept is formalized as an ontology. Most classes represent the corresponding concept of the Situation Theory. *Situation*, *ElementaryInfon, Relation, Polarity* etc. are linked via object properties. Individual situations, facts (infons) and further related aspects are modelled as instances of these classes. Some classes extend the Situation Theory concepts, e.g. *dimensionality* describes the dimension of an attribute.

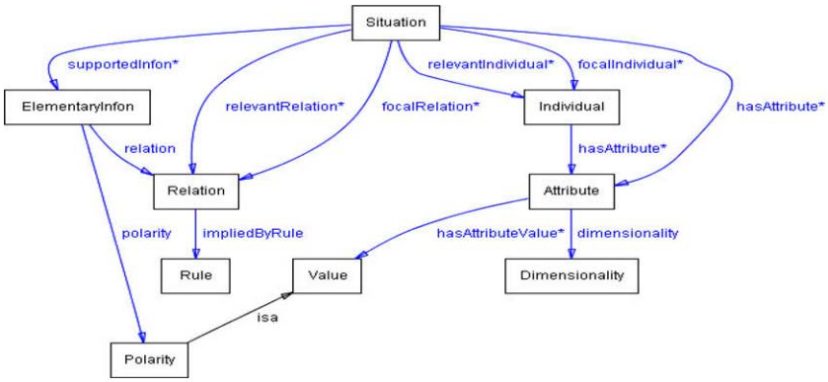

Figure 2.12 Situation Theory Ontology [54]

The situation inference is then based on an OWL reasoner and further defined logical rules. By formulating queries based on SPARQL it can be checked if a certain situation is occurring.

This approach has its strength in its strong theoretical foundation for situation representation and its weakness in the reasoning capabilities. It's limited to logical inference; it does not consider uncertainty or temporality. Contributors of domain and application specific approaches, for example in [55], argue that a general modelling approach implies overhead and has negative impact on performance.

Dempster-Shafer Theory for Situation Inference. The authors in [32] propose an approach for situation inference with DST based on a directed acyclic graph, shown in Figure 2.13. The inference process takes into account the belief from sensors, quality information, belief of evidence and evidence fusion to determine if a situation is occurring. The process starts from sensor data at the bottom which is abstracted to context. This information is used for situation inference with the aforementioned considerations.

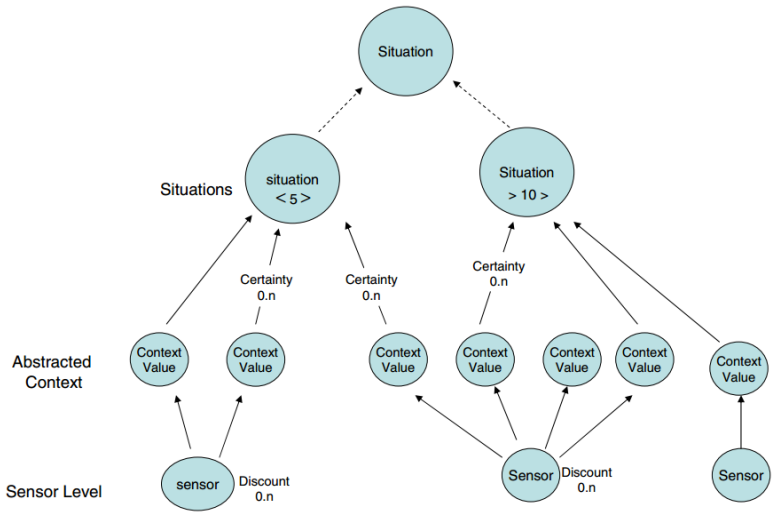

Figure 2.13 Situation Inference Diagram [32]

The challenge of DST-based systems is the specification of the evidential network - in this case the directed acyclic graph. Reuse of these graph specification is not discussed.

BeAware! The situation awareness framework BeAware! is based on an extension of the Core SAW Ontology, which was described earlier in this section. Relation types were added to the ontology to support a better representation of time and space related aspects, providing a solution for the lack of situation type support [56]. The overall architecture of the system is depicted below in Figure 2.14. Domain specific ontologies are integrated via *mappers*. The situation assessor uses the mapped input and rule-based engines based on the Core SAW Ontology for situation inference.

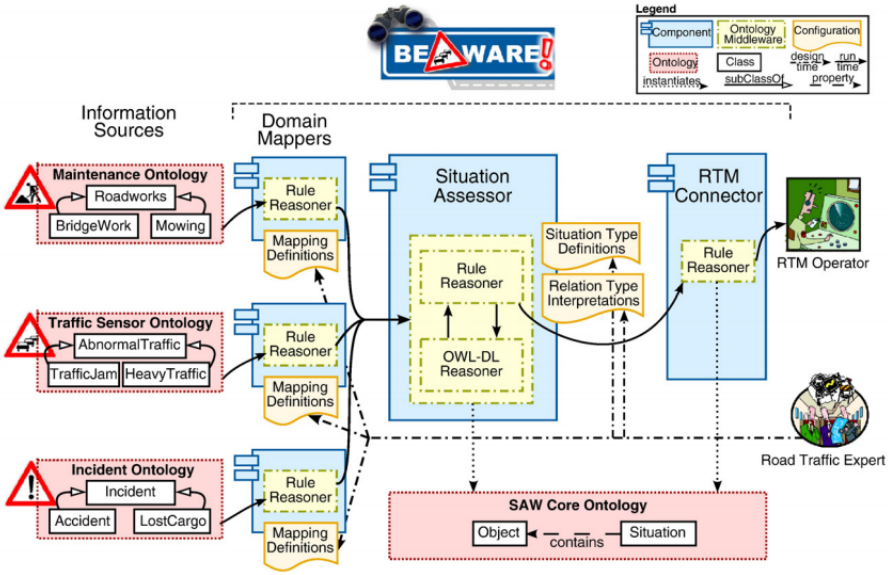

Figure 2.14 BeAware! Architecture applied to Road Traffic Management [56]

The use of primitive relations as a foundation for ontology-based situation awareness is justified in [57]. It is argued that situation awareness is based on deriving relations between objects. Domain dependent and complex situational relations implicitly use domain independent and simple primitive relations. By incorporating these primitive relations into a framework, domain-independent algorithms can be implemented to a certain extent, situational relations can be described by using existing primitive relations and it still allows the integration of exceptional cases. The extension of the *Relation* object in the ontology is given in Figure 2.15. It is decomposed into primitive, spatial, and temporal relations.

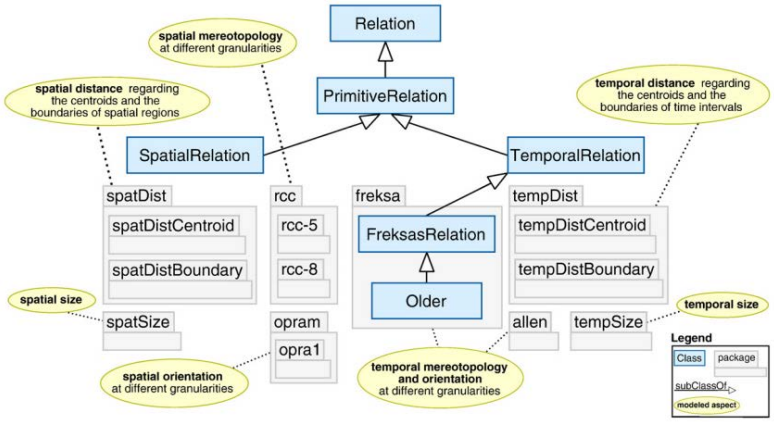

Figure 2.15 Extended Relations of the Core SAW Ontology [56]

The reasoning capabilities of this approach exploit the definition of primitive relation types to derive relations. As for ontological reasoning, the specifications of disjoint, equivalent and subsumed relation types are used to map the given configuration of objects to specified relations. As in the SAWA approach, temporality is considered by observing the evolution of relations over time [56].

The strength of this approach include the knowledge foundation and general applicability of the reasoning technique. It lacks the formal integration of sensors as well as handling their uncertainty and the system setup specification.

Ambient Intelligence for Situation Awareness. This approach proposes an ontology-based situation awareness architecture decomposed into perception, comprehension and projection [58], which is illustrated below in Figure 2.16. It is applied for face detection and is built upon three levels: *Perception* of the environment, *comprehension* of the perceived state and *projection* for evaluation.

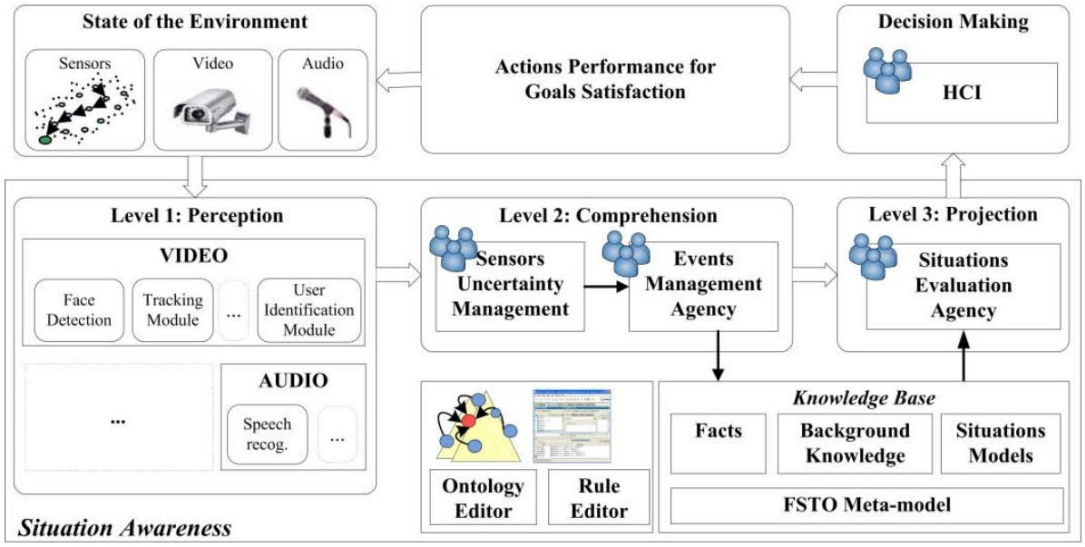

Figure 2.16 Ambient Intelligence for Situation Awareness Architectural Model [58]

The reasoning tasks are distributed among different agents and the reasoning process is driven by the ontological knowledge base, the Fuzzy Situation Theory Ontology (FSTO). FSTO is an extension of STO with incorporated fuzzy modelling by extending the *polarity* of *infons* with more values than simply true and false, e.g. *quite true* [59].

This approach proposes a holistic framework to situation awareness. Considering uncertainty and incompleteness of perceivable information, based on situation theory, provides a thorough foundation for situation modelling. Whereas the architectural model fosters reuse due to its modular structure, inference is bound to a specific issue.

Wavellite. The Wavellite framework aims for situation awareness in environmental monitoring [60]. However, it is concerned with the general situation awareness issues like representation of situational knowledge and the interpretation of heterogeneous sensor data. The project is available open source[2].

[2] https://www2.uef.fi/en/envi/projects/wavellite - Accessed 18/04/2016

The framework is based on an ontology which combines upper ontologies such as the Semantic Sensor Network (SSN) ontology, RDF Data Cube Vocabulary, Situation Theory Ontology and more to model situational knowledge, sensor setup and to store sensor observations. The framework is structured into four layers, measurement, observation, derivation and situation, to achieve situational abstraction from sensor data observations. Furthermore the framework allows implementation of specification-based (i.e. inference rules) as well as learning-based techniques (neural networks) to interpret sensor data [61].

The strength of this approach lies in it holistic view by modelling the situations as well as the system environment and supporting different engines for situation inference. Limitations identified are, however, performance and usability and its focus on environmental monitoring [60]. Prediction of context and actuators are not considered. The framework does not aim to provide a domain independent way for situation inference, but allows the implementation of application engines.

2.3.3 Comparison

The previous section presented numerous approaches to situation awareness. Table 2.2 shows a comparison of these. The assessment of fully (+), partially (~) or not sufficient (-) met criteria, identified in 2.3.1, are based on the individual and overall discussion. The distinction of a criteria being not met sufficiently or being not considered (n.c.) depends on the scope of the approach. If a criteria is not considered and an evaluation cannot be inferred with the given sources it is marked with a question mark (?).

The table lists the presented approaches ordered by year of publication and shows the number of citations and used techniques. Each approach was then marked for each criteria to provide an overview of the strength and weaknesses of the approaches. Usually the approaches do not consider or do not support all identified aspects.

2.3.4 Conclusion

Based on the discussion and assessment which were presented throughout this chapter the following three conclusions can be made:

1. Ontologies are the most promising approach to develop a situation model and to handle knowledge management.

This is given due to the nature of ontologies by the rich expressiveness compared for example to first order logic. Due to its popularity in the semantic web domain, ontologies have been proven as a feasible tool for knowledge reuse. The assessment shows that approaches based on ontologies usually get better results for situation modelling and knowledge reuse.

Year	Cit.[3]	Approach	Techniques	Situation			Knowledge			Reasoning			Application		
				Space and Time	Roles and Situation Types	Relations	Situational Knowledge	System Knowledge	Knowledge Reuse / Sharing	Capability and Universality	Uncertainty and Temporality	Prediction	Sensor Data Acquisition	Actuators	Performance / Applicability
2003	195	Context Broker Architecture [42]	Ontology, Formal Logic, Policies	+	~	-	+	-	+	~	-	-	+	n.c.	+
2004	114	Context Space Theory [62]	Algebra, Formal Logic, Bayesian, Dempster-Shafer, Markov Model	~	~	+	~	n.c.	-	+	+	+	+	+	+
2005	76	Situation Awareness Assistant [49]	Ontology, Rules	+	~	+	+	-	+	~	~	n.c.	n.c.	n.c.	+
2006	84	Hierarchical Situation Modelling [50]	Ontology, Formal Logic	+	-	-	~	-	+	-	-	n.c.	n.c.	n.c.	+
2007	29	Situation Lattice [18]	Formal Logic, Algebra, Bayesian Network	+	+	+	+	~	-	+	+	n.c.	~	n.c.	~
2007	67	Situational Context Ontology [19]	Ontology, Fuzzy Logic	+	~	~	+	n.c.	+	~	+	n.c.	n.c.	n.c.	+
2008	486	Activity Recognition in a Home Setting [36]	Hidden Markov Model, Conditional Random Fields	~	~	~	~	~	-	+	+	n.c.	+	n.c.	+
2008	172	Activity Recognition Based on Acceleration [52]	Neural Network	~	~	~	~	~	-	~	+	n.c.	?	n.c.	~
2009	179	Situation Theory Ontology [54]	Ontology, Rules	+	+	+	+	-	+	-	n.c.	n.c.	n.c.	n.c.	~
2009	36	DST for Situation Inference [32]	Dempster-Shafer, Fuzzy Logic	+	-	+	-	-	-	+	+	n.c.	n.c.	-	~
2010	67	BeAware! [56]	Ontology, Relations	+	+	+	+	-	+	+	~	+	?	?	+
2011	6	Ambient Intelligence for Situation Awareness [58]	Ontology, Fuzzy Logic	+	+	+	+	n.c.	+	~	+	n.c.	+	+	?
2015	-	Wavellite [60]	Ontology, Rules, Neural Networks (MLP)	+	+	+	+	+	+	~	-	n.c.	+	n.c.	~

Table 2.2 Comparison of Situation Awareness Approaches

2. Only a hybrid approach, combining different situation identification techniques, can integrate highly capable reasoning.

Solely one technique is not capable of providing general reasoning, considering uncertainty and temporality. Each technique is appropriate to handle different aspects but only a combination can provide a holistic solution. It can be seen that some approaches are more suitable for using different techniques than others.

[3] Number of citations; Source: Google Scholar (https://scholar.google.com) - Accessed 27/01/2016

3. The gaps between sensing, reasoning and actuating is often not considered in situation aware approaches.

The formal integration of sensor data and actuators, for example, is usually considered as a separate issue. It is often not discussed how these can be integrated in the overall framework.

This thesis aims to provide an appropriate solution to all identified criteria by combining existing techniques which solve the different issues. Based on the survey of this section, the approach will be based on the *Situation Theory Ontology* as a situation model and knowledge base and integrating it to the *Context Space Theory* as a general and flexible approach for reasoning, both having a strong theoretical foundation. The performance issue for a general modelling technique like STO is not a major concern. The knowledge specified in STO will be used to initialise the context space and thus does not affect runtime reasoning. Furthermore a solution for a formal integration for sensors and actuators will be discussed and added to the overall framework.

2.4 Summary

This chapter introduced the basic concepts of situation awareness and fundamental techniques that are applied in this domain to identify situations. Based on an investigation of situation aware systems, required criteria that a general, holistic framework to situation awareness has to consider have been identified. Related approaches to situation were introduced and their strength and weaknesses discussed.

Based on the evaluation of these approaches three conclusion were formulated, which all motivate the approach of this thesis, i.e. using the Situation Theory Ontology as a situation model and knowledge base and combining it with the Context Space Theory as a general reasoning approach. Furthermore the integration of sensors and actuators will be discussed and added to the framework to meet all the identified requirements.

3 Ontology-based Generation of Situation Spaces in CST

This chapter presents the integration of an ontology as a knowledge base for situation aware systems based on the Context Space Theory, as it was motivated in the previous chapter. Firstly, an upper ontology will be designed based on the Situation Theory Ontology (STO) and the Semantic Sensor Network (SSN) ontology. Afterwards a formal approach to extract the knowledge from the ontology and to integrate it into CST will be proposed.

The approach described in this chapter is based on the following high-level concepts:

1. Designing an upper ontology to model situations and the system's environments.
2. Defining algorithms to extract the knowledge and generate situation spaces in CST.

Only a theoretical point of view is taken. The relevant concepts are investigated more thoroughly and a solution for modelling and situation generation is proposed. The implementation of the proposal and also other implementation relevant aspects of a situation aware system, as discussed in chapter 2, will be considered in chapter 4.

3.1 Ontologies

Ontologies have been introduced in section 2.3.2. As this chapter focus intensively on ontologies, related concepts and terminology are briefly explained in more detail.

Ontologies can be represented in various different ways, the most prominent being the Resource Description Format (RDF) [63] in combination with RDF Schema (RDFS) [64], and the Web Ontology Language (OWL) [65]. All three are standards of the World Wide Web Consortium (W3C).

RDF is a model of metadata to provide a structure for the description of resources. These resources or identified with a Uniform Resource Identifier (URI) [66]. The resource description is based on RDF statements which consist of triples, formed respectively by a subject, predicate and object. The most important property defined by RDF is `rdf:type`, meaning that the subject is of a certain type, specified through the object resource [63]. RDF does not define general relationships, e.g. to make statements between groups of resources or to define type of relationships. This is achieved with RDFS, which provides a vocabulary for RDF data. This includes the specification of classes, `rdfs:class`, and defining properties such as `rdfs:subClassOf`, `rdfs:subPropertyOf`. It also allows to state which classes are allowed as a domain (`rdfs:domain`) and range (`rdfs:range`) of a property [64]. With RDFS basic ontologies can be developed.

OWL adds richer semantics to the RDF statements than RDFS by adding more vocabulary to describe properties and classes. In OWL general statements about set of individuals are referred to assertions. OWL supports complex classes and property expressions which represent sets of individuals. These support for example intersection (`ObjectIntersectionOf`) and union (`ObjectUnionOf`) of instances of certain classes, or they can restrict properties, e.g. by existential (`ObjectSomeValuesFrom`) or universal quantification (`ObjectAllValuesFrom`), an individual restriction (`ObjectHasValue`) or with cardinality restrictions (e.g. `ObjectMinCardinality`). Furthermore an OWL ontology defines a set of axioms that can refer to classes, properties or individuals and make statements about what is true in the ontology. These are based on class expressions, property expressions and assertions. Besides the `owl:subClassOf` axiom, which corresponds to `rdfs:subClassOf`, classes can for example be defined as equivalent (`equivalentClasses`) or disjoint (`disjointClasses`). Correspondingly, these can also be defined for properties, furthermore properties can be defined as reflexive, inverse or transitive, as examples [65].

The main advantage of these complex expressions is that it enables inference based on first order logic (FOL) and validation of the facts in the ontology, also referred to as *ABox* (assertions), or compliant with the given axioms in the ontology, which are referred to as *TBox* (terminology). Inference in ontologies means to automatically generate new relationships [67].

These tasks are performed by so-called reasoners. The richer the expressivity of the ontology language, the bigger are the inference capabilities of the reasoner, but also the more complex the inference process becomes. OWL provides three sublanguages with different expressiveness. OWL Lite only supports a classification hierarchy and simple constraints. OWL DL supports all capabilities with some restrictions to retain computational completeness. OWL Full offers maximum expressiveness, but does not have computational guarantees, i.e. no guarantee that all inferred statements are valid [65].

The approach described in the following and the system proposed throughout the thesis is based on OWL2 DL. Rich semantic expressiveness has priority over simpler language with less complexity to provide a sophisticated foundation for situation modelling. Still, computational completeness is necessary, which is guaranteed by OWL DL.

3.2 Ontology Design for Context Space Theory

This section proposes an upper ontology for CST which is capable of modelling all relevant knowledge assets. Thus, the required assets for a CST-based system are analysed first. Subsequently the upper ontology will be developed based on the Situation Theory Ontology and other standards.

3.2.1 Analysing Required Knowledge Specifications

In the following the required knowledge assets that need to be stored in the ontology are analysed by investigating situation representation in the Context Space Theory closer, as primarily defined in [22]. The basic approach of the Context Space Theory has already been introduced in section 2.3.2.

Formally a *Situation Space* S_j (1) is defined as a set of acceptable regions for each corresponding context attribute. An *Acceptable Region* A_n^j (2) of a context attribute is a set of elements that satisfy a certain predicate $P(V)$.

$$S_j = (A_1^j, A_2^j, \dots, A_n^j) \tag{1}$$

$$A_i^j = \{V | P(V)\} \tag{2}$$

The *Relevance Function* $w_S(a_i)$ (3) assigns a weight w_i to each context attribute a_i of a situation space which signifies the relative importance of a particular context attribute to infer a situation.

$$w_S(a_i) = w_i; w_i \in [0,1] \text{ and } \sum_{i=1}^{n} w_i = 1 \tag{3}$$

The *Contribution Function* $\eta_i^S(x_i)$ (4) assigns a contribution degree c for each value x_i in an acceptable region for a situation space. In other words, some values of an acceptable region may infer stronger that a situation is occurring than others.

$$\eta_i^S(x_i) = c; \; c \in [0,1] \tag{4}$$

The Context Space Theory is not restricted to a single reasoning technique. Instead the *Inference Function* γ (5) is based on a placeholder function. The reasoning technique is expected to calculate a *Confidence Value* μ which is compared to a *Confidence Threshold* ε_i of the situation space. A situation occurs if the confidence is higher than the threshold.

$$\gamma = (\mu(x_i, S_i, \Sigma_i) \geq \varepsilon_i)$$

$$\mu(x_i, S_i, \Sigma_i) \in [0,1], \varepsilon_i \in [0,1], \; x_i\text{: context state}, \; S_i\text{: situation space}, \\ \Sigma_i\text{: set of functions specifying relations between attributes and situations} \tag{5}$$

Furthermore CST allows the definition of complex situation expressions by defining *Logical Operators* (6) to combine different situation spaces.

$$\begin{array}{c} and \; (\wedge) \\ or \; (\vee) \\ not \; (\neg) \end{array} \tag{6}$$

For the scope of this thesis it is assumed that the calculation of the confidence measure μ is solely based on the concepts discussed earlier, i.e. only taking the relevance and contribution function into account. Thus the confidence value that a situation is occurring with a context state x is calculated with (7).

$$\mu_s(x) = \sum_{i=1}^{N} w_i * \eta_i^S(x_i) \tag{7}$$

As a primary conclusion, the ontology needs to be able to capture all these mentioned aspects of situation specification and inference of CST. The aspects required for each situation are summarized in Table 3.1. Moreover it shows how the corresponding values need to be specified in the ontology.

ID	Requirement	Representation
$\mathbf{R_{Sit}1}$	Name of situation spaces	String
$\mathbf{R_{Sit}2}$	Name of context attributes	String
$\mathbf{R_{Sit}3}$	Situation space compositions	Relation
$\mathbf{R_{Sit}4}$	Acceptable regions for all context attributes	Double/String
$\mathbf{R_{Sit}5}$	Relevance function - Weight for each context attribute	Double ∈ [0,1]
$\mathbf{R_{Sit}6}$	Contribution function - Degrees within acceptable regions	Double ∈ [0,1]
$\mathbf{R_{Sit}7}$	Confidence Threshold for a Situation Space	Double ∈ [0,1]
$\mathbf{R_{Sit}8}$	Logical operators between Situation Spaces	Rules

Table 3.1 Requirements regarding the Situation Representation

These requirements would be sufficient to generate all specified situation spaces from an ontology. However, the aim is to consider situation aware applications as a whole and to integrate all relevant assets.

The Context Space Theory does not provide a framework to handle individuals and objects involved in situations. The situation spaces are modelled in a general way and each involved individual maintains a different context state in the application space. For example, a situation called *Running* is defined in a general way and with the context state of each involved person it can be inferred if this particular individual is running. Situations are not exclusively related to persons or individuals, but can be related to objects, too. A room, fridge or window, for example. The term *individual* in the following refers to persons as well as objects which are involved in a situation.

A situation aware application does not work on the general situation definitions but rather with actual individuals involved in these situations. By integrating the knowledge about involved individuals in the ontology, it is possible to formally define the relation between individuals and situations and it provides a framework, a base, to request the situation occurrences of different individuals. Thus, the related requirements identified are as shown in Table 3.2.

ID	Requirement	Representation
$\mathbf{R_{Ind}1}$	Individual name	String
$\mathbf{R_{Ind}2}$	Involvement in situation	Relation

Table 3.2 Requirements regarding Involved Individuals

Typically, in a situation aware approach, information about the environment is retrieved from sensors. In CST, the data provided by sensors has to be linked to context attributes in order to be able to update the context state. However, this is not sufficient. Sensors also need to be linked to the individuals of which the context attribute is observed. E.g. a sensor that captures the heart rate of a person needs to be linked to the context attribute 'heart rate' and also to the person that is observed. Another sensor might measure the same context attribute of a different individual.

From this follows that information about the sensor setup has to be modelled, in order to clearly identify a sensor and to link it to situational aspects. The derived requirements are listed in Table 3.3.

ID	Requirement	Representation
$\mathbf{R_{Sen}1}$	Sensor ID	String
$\mathbf{R_{Sen}2}$	Related context attributes	Relation
$\mathbf{R_{Sen}3}$	Related individuals	Relation
$\mathbf{R_{Sen}4}$	Sensor deployment setup	String
$\mathbf{R_{Sen}5}$	Actuator deployment setup	String
$\mathbf{R_{Sen}6}$	Actuating property	Relation

Table 3.3 Requirements regarding Sensors and Actuators

The ontology described in the following sections is aligned to the aforementioned requirements and it will be shown which parts of the ontology meet which requirement.

3.2.2 Modelling Situation Spaces with STO

As motivated in the second chapter of this thesis, the selected approach is based on the Situation Theory Ontology[4]. This section describes how the concepts of STO can be linked to the Context Space Theory. The basic concepts of STO have already been described in section 2.3.2.

To understand STO more thoroughly, first the Situation Theory is considered in a formal way, as primarily defined in [53]. A situation s is defined by specifying the infons σ, i.e. informational items about the real world, they support (8). An infon is denoted as shown in (9), where R describes a relation among the objects $a_1, \ldots, a_n$ that is either true (polarity $i = 1$) or false (polarity $i = 0$). Objects can be specified as parameters which represents objects of a certain type, indicated as $\dot{a_n}$.

[4] http://www.vistology.com/ont/2008/STO/STO.owl - Accessed 27/01/2016

$$s \vDash \sigma \tag{8}$$

$$\sigma = \ll R, a_1, \ldots, a_n, i \gg \tag{9}$$

Figure 3.1 shows a simplified view of STO, focusing on the key concepts. The classes shown correspond to the presented concepts of the Situation Theory. Objects part of a relation can be individuals, attributes or other situations. Figure 3.2 highlights the CST concepts, which were already explained, so it can be easier referred to the corresponding concepts in the following.

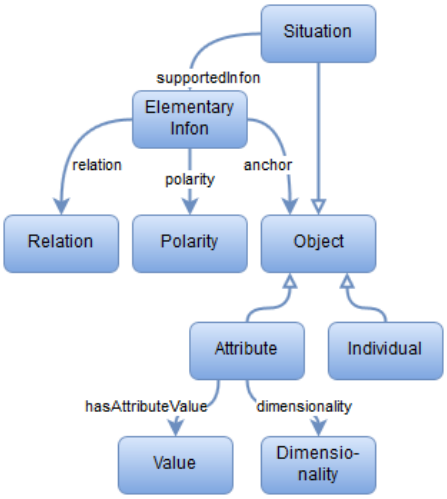

Figure 3.1 Simplified View of STO

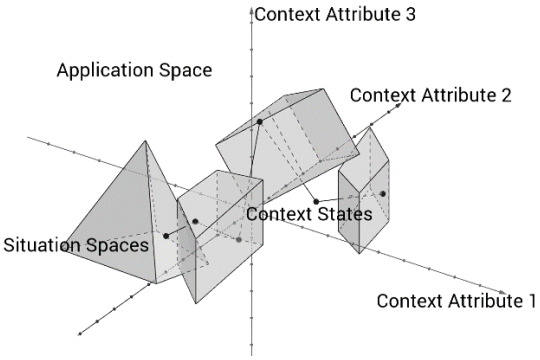

Figure 3.2 Context Space Theory

Table 3.4 below presents the relation of concepts of the Situation Theory Ontology and the Context Space Theory. I.e., the knowledge specified in STO can be used for the corresponding part of the situation generation in CST. Furthermore it shows which requirement is covered by which concept.

STO Concept	**CST Concept**	**Associated Req.**
Situation subClassOf Object	Name of situation space	$R_{Sit}1$
Attribute subClassOf Object	Name of context attribute	$R_{Sit}2$
Individual subClassOf Object	-	$R_{Ind}1$
Situation supportdInfon ElementaryInfon	Composition of situation space	$R_{Sit}3$, $R_{Ind}2$
Attribute hasAttributeValue Value	Acceptable region per attribute	$R_{Sit}4$
Attribute dimensionality Dimensionality	Unit of acceptable region values	-

Table 3.4 Mapping of STO and CST Concepts

It can be perceived that the basic definitions for situation spaces in CST can be extracted from knowledge given in the Situation Theory Ontology. Furthermore, STO considers individuals that are involved in situations which covers the requirements regarding involved

individuals in the previous section. However, more CST-specific definitions, like the relevance and the contribution function, need to be added to the ontology. This will be described in section 3.2.4.

3.2.3 Modelling Dependencies with SSN and SAN

To incorporate specifications about sensors and the setup of a situation aware system, the Semantic Sensor Network (SSN) ontology[5] will be used. The ontology not only allows to describe the specification of sensors, but also the way of sensing and the observations made by sensors [68]. SSN is the result of an expert-based review initiated by the W3C to define a standard for a sensor ontology[6]. Since the observations of sensors are maintained by the context state of CST, mainly the sensor perspective of SSN is of interest for this thesis.

SSN does not incorporate the modelling of actuators, which has been identified as an asset of a situation aware system. Thus, the Semantic Actuator Network (SAN) ontology[7] is also added to the upper ontologies. SAN was developed based on SSN, following the same design patterns correspondingly for actuators.

A simplified view of SSN and SAN with the major classes important for this work is depicted in Figure 3.3 below.

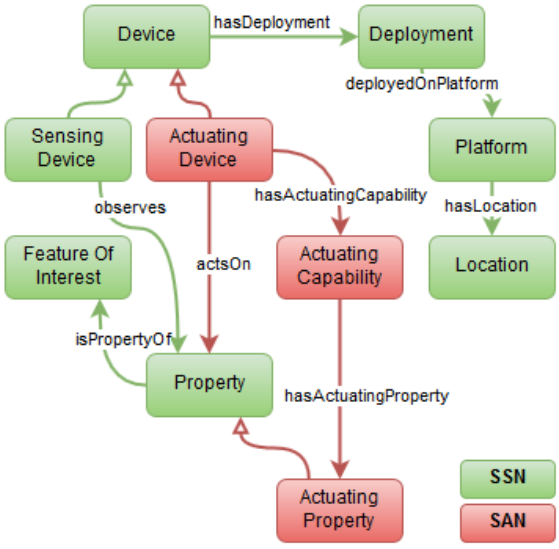

Figure 3.3 Simplified View of SSN and SAN

[5] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn.owl - Accessed 10/05/2016

[6] https://www.w3.org/2005/Incubator/ssn/wiki/Review_of_Sensor_and_Observations_Ontologies - Accessed on 18/04/2016

[7] https://www.irit.fr/recherches/MELODI/ontologies/SAN.owl - Accessed 10/05/2016

Considering that CST does not incorporate the sensor setup natively, most of the concepts of SSN cannot be directly mapped to CST concepts. The only exception is the class *Property*, which corresponds to context attributes in CST. However, it is shown that a sensor setup can be described with the given ontology. Thus, some of the earlier defined requirements are met, as shown in Table 3.5.

SSN Concept	**CST Concept**	**Associated Req.**
SensingDevice	-	$R_{Sen}1$
SensingDevice observes Property	Name of context attribute	$R_{Sen}2$
Deployment, Platform, Location	-	$R_{Sen}4$, $R_{Sen}5$
ActuatingProperty subClassOf Property	-	$R_{Sen}6$

Table 3.5 Mapping of SSN and CST Concepts

3.2.4 Contribution: Upper Ontology for a CST-based System

The selected upper ontologies proved feasible to meet some of the aforementioned requirements. However, these two concepts need to be combined in order to establish relevant links between corresponding assets and to provide one knowledge base. Furthermore, specific assets required for CST have to be added to the ontology. These aspects are discussed in this section. The result will be referred to as *CSTO*, Context Space Theory Ontology, throughout this thesis. The contribution of this section is to:

- Map SSN/SAN with STO.
- Add properties and classes for CST concepts.
- Form an upper ontology which conforms to all identified requirements (see 3.2.1).

The final upper ontology is depicted in a simplified view in Figure 3.4. The properties and classes in orange indicate that they are newly added. First of all, SSN/SAN and STO have to be mapped. This is achieved by creating two additional owl:`subClassOf` axioms. The class `sto:Attribute` and the class `ssn:Property` can both be referred to context attributes in CST. By specifying `sto:Attribute` as a subclass of `ssn:Property`, sensors can *observe* the context attributes of situation definitions in STO. Correspondingly actuators have actuating attributes.

Furthermore, the class `ssn:FeatureOfInterest` represents the object to which a `ssn:Property` belongs to. This is mapped to the class `sto:Individual`, hence this class represents objects that are in a relation with `sto:Attributes`. The property `csto:observesPropertyOf` is added among the `ssn:SensingDevice` and `ssn:FeatureOfInterest` to serve as a shortcut of the connection between these two classes. With these added elements to the model, a sensor can observe the attributes of individuals that are involved in situations.

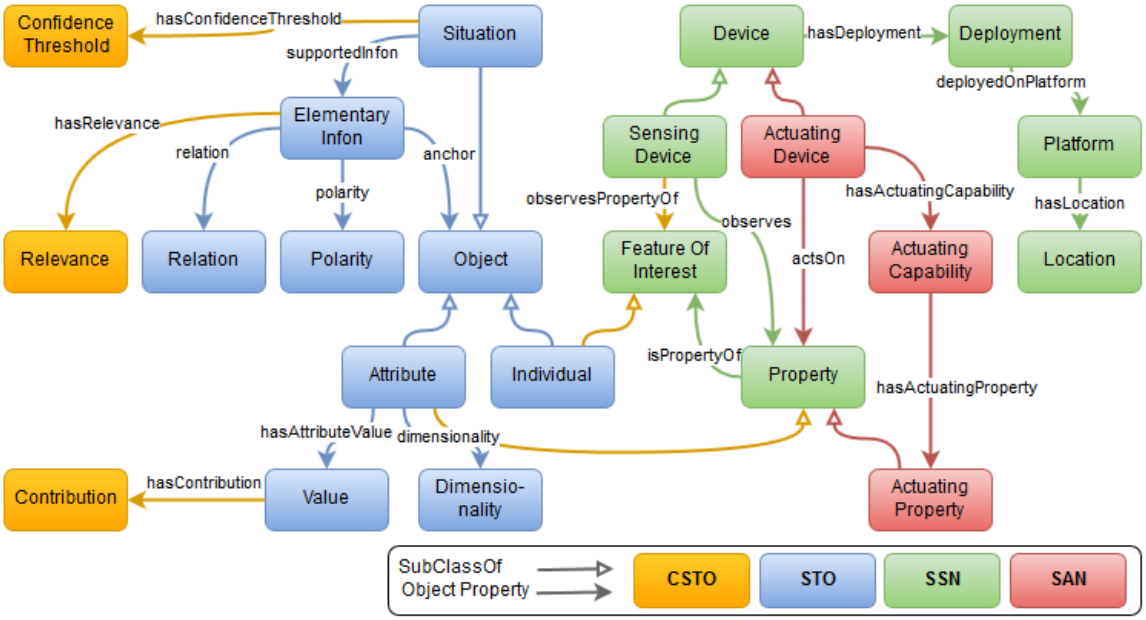

Figure 3.4 Upper Ontology for Context Space Theory (CSTO)

Three classes and properties are added to represent necessary information about CST-specific concepts. To comply with the STO way of ontological design, values that represent `csto:Contribution`, `csto:Relevance` and `csto:ConfidenceThreshold` are wrapped in corresponding classes. The confidence threshold is simply attached to situations by the property `csto:hasConfidenceThreshold`.

The relevance values in CST are simply attached to a context attribute of a situation. However, in STO individuals are considered in situation representations. Since situations are composed of infons, the property `csto:hasRelevance` needs to be attached to the `sto:ElementaryInfon` class. This attachment allows context attributes of same situations for different individuals to have a different relevance value.

Finally, the property `csto:hasContribution` defines a contribution value for values in acceptable regions. Since it is impractical to assign for each single value a contribution value, acceptable regions can be specified in intervals and the contribution value applies to all values of the interval.

The last remaining challenge is the representation of complex situation expressions. Situation Semantics [53] define an infon logic which includes the *conjunction* (10) and the *disjunction* (11).

$$s \vDash \sigma \wedge \tau \text{ iff } s \vDash \sigma \text{ and } s \vDash \tau \tag{10}$$

$$s \vDash \sigma \vee \tau \text{ iff } s \vDash \sigma \text{ or } s \vDash \tau \text{ or both} \tag{11}$$

These operations correspond to the logical operators in CST, however, they have not been explicitly considered during the development of STO. The creators of STO suggest a

rule language support to create complex situation expressions, but only mention conjunction [54].

The representation of the situation expressions with these operators in OWL is not trivial because OWL does not support complex object property expressions. Definitions that corresponds to class axioms like (12), for example, are not legal statements within OWL.

```
situationX owl:subClassOf (
      sto:supportedInfon owl:value (infon1 OR infon2)          (12)
)
```

Indeed, an integration of a logical rules engine is the best approach for this level of abstraction. The feasibility and even the generation based on these rules has been described and proven in [69]. Since the rules are based on defined situations, this approach can be adopted for CSTO. Thus the integration of complex situation expressions will not be further discussed.

Table 3.6 below summarizes the added assets and show which requirements they cover.

CSTO Concept	**Associated Req.**
Attribute subClassOf Property	$R_{Sen}2$
Individual subClassOf FeatureOfInterest	$R_{Sen}3$
ElementaryInfon hasRelevance Relevance	$R_{Sit}5$
Value hasContribution Contribution	$R_{Sit}6$
Situation hasConfidenceThreshold ConfidenceThreshold	$R_{Sit}7$
Logical rules	$R_{Sit}8$

Table 3.6 CSTO Concepts and Covered Requirements

This concludes the development of an upper ontology for a CST-based system. The proposed ontology CSTO fulfils all identified requirements. However, for one requirement, complex situation expressions based on logic operators, an external rule engine outside the scope of plain OWL needs to be integrated.

3.3 Generating Situation Spaces

In this section algorithms are proposed that extract and use information modelled in the ontology (presented in the previous section) to initialise the context space by generating situation spaces.

The specification of situations can be done on two levels of abstraction.

- Firstly, based on individual situations, referred to as situation objects.
- Secondly, based on general situation type definitions.

A Situation type is referred to as the logical structure of a situation which is not bound to specific objects but rather to meta-level concepts.

3.3.1 Generation based on Situation Objects

Specific situations in STO are modelled solely with instances of classes, i.e. situation objects, and corresponding situation spaces can be generated in a straight-forward manner. The OWL individuals of type `sto:Situation` need to be asserted with at least one object property `sto:supportedInfon` with a corresponding `sto:ElementaryInfon` instance. Correspondingly, `sto:Attribute`, `sto:Value` and other relevant classes have to be explicitly instantiated in the ontology. Other values, e.g. the `csto:Relevance` or `csto:Contribution`, are optional and can be assigned with default values.

The generation of situation spaces can be split into the following steps:

1. Acquiring relevant knowledge from given CSTO-based ontologies.
2. Store information for the creation of situation spaces in a cache.
3. Create situation spaces from cache, resolving dependencies of subspaces.

Steps 1 and 2 are a straightforward process. The algorithm iterates over all situation individuals defined in the CSTO-based ontology and reads the attached information via the infons etc. and stores the values in cache elements. The tools to obtain the information defined in the ontology programmatically will be discussed in chapter 4.

A simplified algorithm for step 3 is presented as pseudo code below (Algorithm 3.1 and Algorithm 3.2). The approach is based on iterations over the given individuals in the ontology and solves the dependencies of subspaces with recursion. It initializes the application space of CST and generates situation spaces within it and thus prepares the application for reasoning tasks.

```
Function: GenerateSituationSpacesBasedOnObjects()
applicationSpace = new ApplicationSpace();
for situation : situationsFromCache do
   situationSpace = ProcessSituationObject(applicationSpace, situation);
   applicationSpace.addSituationSpace(situationSpace);
endfor
```

Algorithm 3.1 Creation and Population of Application Space

Algorithm 3.1 simply shows the iteration through the cached situations and calling the `ProcessSituationObject()` function described in Algorithm 3.2.

```
Function: ProcessSituationObject(applicationSpace, situation)
situationSpace = new SituationSpace(situation.name);
for infon : situation.getInfons() do
   for anchor : infon.getAnchors() do
      switch anchor.getType() do
         case attribute:
            // Anchored attributes: Add context attribute and acceptable region
            axis = new Axis(attribute.name)
            for acceptableRegion : attribute.getValues() do
               if infon.polarity == 1 do
                  axis.addRegion(acceptableRegion.value,
                                                 acceptableRegion.contribution);
               else do
                  axis.addAsymmetricRegion(acceptableRegion.value,
                                                 acceptableRegion.contribution);
               endif
            endfor
            situationSpace.addAxis(axis, infon.getRelevance());
            break;
         case situation:
            // Anchored situation: Process and add subspace
            SubSpace = ProcessSituationObject(applicationSpace, anchor);
            situationSpace.addAxisSubSpace(SubSpace);
            break;
         case individual:
            //not part of situation spaces in CST
            break;
      endswitch
   endfor
endfor
return situationSpace;
```

Algorithm 3.2 Generation of Situation Spaces

Algorithm 3.2 processes the cached information about a situation and creates the situation spaces of CST. Attributes of anchors of supported infons are used to create an axis in the situation space, attribute values are used to create acceptable regions on the axis and anchored situations are processed first and then added as subspace. Anchored individuals are not considered because they are not part of a situation space definition in CST.

The algorithms above are simplified and only demonstrate the outline of the process of situation space generation. Further steps are for example differentiating between axis types, validating the information and assigning default values if possible.

3.3.2 Generation based on Situation Types

Situation Theory defines so called types which represent a "scheme of individuation" [53], i.e. a meta-level description of objects. The most important types are *SIT*, *INF* and *IND*, which describe the types for situation, infon and individual. A situation type $\dot{s}$ in Situation Theory is formally denoted as shown in (13).

$$[\ \dot{s} \mid \dot{s} \vDash \sigma\ ] \tag{13}$$

This concept was adopted for STO by introducing a meta-level of classes. The types are modelled as subclasses of *TYP* which is a subclass of `owl:Class.` Class definitions about situations, infons, individuals etc. are not modelled as an `owl:Class`, but of the corresponding meta class. This implies that instances of the *TYP* classes are classes, too. STO further introduces a type *ATTR* for attributes [54].

Thus, situation types in the ontology are defined as classes. A situation type class defines for example which infon types it supports by asserting OWL class axioms. A possible axiom to define a situation type can look like (14).

```
SituationTypeX owl:equivalentClass (
    sto:Situation
    and (sto:supportedInfon owl:value InfonTypeX)
    and (sto:supportedInfon owl:value InfonTypeY)
)
```
(14)

As it can be seen, types in STO are specified with the `owl:eqivalentClass` axiom. The introduction of meta-classes in STO allows to specify `owl:ObjectHasValue` axioms to restrict to a group of objects. For example, in `sto:supportedInfon value` *InfonTypeX*, *InfonTypeX* represents all infon objects that are a subclass of this infon type class definition.

A type definition for an infon has the following structure (15).

```
InfonTypeX owl:equivalentClass (
    sto:ElementaryInfon
    and (sto:relation owl:value relationX)
    and (sto:anchor1 owl:value individualX)
    and (sto:anchor2 owl:value attributeX)
    and (sto:polarity owl:value polarityX)
)
```
(15)

Situation types not only simplify the situation modelling process, they are also necessary to ensure reusability of the ontology. Situation specifications may be based on application dependent individuals, for example, and an explicit definition in a situation specification would not allow to adopt the same definition in another system. By referring to general

types, application dependent individuals can be specified in a separate manner, completing the general situation specification.

Algorithms to generate situation spaces are conceptually the same as for situation objects, explained in section 3.3.1. Only the retrieval of the information from the ontology differs.

3.3.3 Populating Ontology with Situation Objects based on Types

The modelling based on meta-classes may be counterintuitive, popular ontology editors for example are not supporting it, which influences the modelling process. Furthermore it might be of interest to create individual situation spaces for each involved individual. This would be an extensive manual process and also impacts reusability because of the dependency to individuals.

Thus, another way of representing situation types is introduced to programmatically populate the ontology with situation objects with all possible situations. Again, the definition of class axioms in the ontology are used to represent these general types of situations. This approach has the advantage of keeping the scope of situation modelling within OWL, compared to modelling these types for example with rules. Rules are semantically a good extension of OWL, however, syntactically separated which discourages reuse.

To differentiate from the situation type definition presented in the previous chapter, this type is specified with the `owl:subClassOf` axiom. The structure for situation type axioms (16) and infon type axioms (17) are shown below.

```
SituationTypeX owl:subClassOf (
    sto:Situation
    and (sto:supportedInfon owl:some InfonTypeX)
    and (sto:supportedInfon owl:some InfonTypeY)
    and (sto:relevantIndividual owl:some IndividualX)
)
```
(16)

Types in this case are not type of specific meta-level classes. Thus, the `owl:ObjectSomeValueFrom` axioms is used to specify the instantiated situation objects can have `sto:supportedInfon` object properties to all instantiated infon objects of that infon type class definition.

```
InfonTypeX owl:subClassOf (
    sto:ElementaryInfon
    and (sto:relation owl:value relationIndividualX)
    and (sto:anchor1 owl:some ObjectX)
    and (sto:anchor2 owl:some ObjectY)
    and (sto:polarity owl:value polarityX)
)
```
(17)

The object property `sto:relevantIndividual` from the situation type definition is defined by STO and captures the individuals that participate in a situation. The `owl:Object-SomeValueFrom` axiom will be used to create situation objects for all individuals of the specified type.

The algorithm for the ontology population is based on iteration through subclasses of `sto:Situation` and check if class axioms were correctly defined. In that case supported infons will be created for each combination of specified instances of anchored classes. This implies to create an infon instance for each element in the Cartesian product of the sets of instances of the anchored classes. A situation instance is created for each existing instance that has an `owl:subClassOf` axiom of the class referred by the `sto:releventIndividual` object property.

The selected functions presented as pseudo code below are meant to present an outline of the strategy to create instances in the ontology, resolving the dependencies to sets of instances of particular subclasses.

```
Function: PopulateSituations(ontology)
// Go through all situation subclasses and parse their axioms.
for situationType : ontology.getSituationSubclasses() do
   for axiom : situationType.getAxioms() do
      switch axiom.getType() do
         case: supportedInfon someValuesFrom:
            supportedInfonReferences.add(axiom.getReferredClass());
            break;
         case relevantIndividual someValuesFrom:
            relevantIndividualReferences.add(axiom.getReferredClass());
            break;
         case hasConfidenceThreshold value:
            confidenceThresholdInstance = axiom.getReferredInstance()
            break;
      endswitch
   endfor
   // Create a situation instance for each relevant individual.
   for relevantIndividualInstance : relevantIndividualReference.getInstances();
      situationInstance = CreateSituationInstance(situationType,
                                                  supportedInfonReferences,
                                                  relevantIndividualInstance,
                                                  confidenceThresholdInstance);
   endfor
endfor
```

Algorithm 3.3 Populating the Ontology, Demonstrating Parsing of Class Axioms

Algorithm 3.3 demonstrates the process of parsing class axioms, in this case for a class of type `sto:Situation`. For each situation type specified as described previously the referred classes and instances will be read and once all information is gathered, a new instance is created.

```
Function: CreateSituationInstance(situationType,
                                  supportedInfonReferences,
                                  relevantIndividualInstance,
                                  confidenceThresholdInstance)
situationInstance = new OWLIndividual();
situationInstance.addClassAssertion(subClassOf situationType);
situationInstance.addObjectPropertyAssertion("relevantIndividual",
                                              relevantIndividualInstance);
situationInstance.addObjectPropertyAssertion("hasConfidenceThreshold",
                                             confidenceThresholdInstance);
for supportedInfonReference : supportedInfonReferences do
   infonInstances = CreateInfonInstances(supportedInfonReference);
   for infonInstance : infonInstances
      situationInstance.addObjectPropertyAssertion("supportedInfon", infonInstance);
   endfor
endfor
ontology.addIndividual(situationInstance);
```

Algorithm 3.4 Creation of OWL Situation Instances

```
Function: CreateInfonInstances(infonClass)
AnchorCombinationSet = cartesianProduct(infonClass.getAssertedAnchoredInstances());
for anchorCombination : AnchorCombinationSet do
   infonInstance = new OWLIndividual();
   infonInstance.addClassAssertion(subClassOf infonClass);
   infonInstance.addObjectPropertyAssertion("relation",
                                            infonClass.getAssertedRelationInstance());
   infonInstance.addObjectPropertyAssertion("polarity",
                                            infonClass.getAssertedPolarityInstance());
   for anchorInstance : anchorCombination do
      infonInstance.addObjectPropertyAssertion("anchorX", anchorInstance);
   endfor
   ontology.addIndividual(infonInstance);
   infonInstances.add(infonInstance);
endfor
return infonInstances;
```

Algorithm 3.5 Creation of OWL Infon Instances

Algorithm 3.4 demonstrates how a particular instance of a situation is created. A new OWL individual has to be created and asserted with the appropriate axioms. Parsing the axioms from the class definition is neglected, since this was already shown in Algorithm 3.3.

Finally, Algorithm 3.5 shows the steps to create instances for the infons for each possible combination (Cartesian product) of the anchored instances. Again, an OWL individual is created, asserted with corresponding axioms and added to the ontology.

3.4 Summary

This chapter introduced an upper ontology that is capable to model all relevant knowledge assets required for a CST-based situation aware system. Furthermore, algorithms that allow the automated situation space generation in the context space, based on the ontology specifications, were formulated. For the ease of situation specification and its reuse, a concept to represent situation types based on class axioms and algorithms for the automated creation of instances in the ontology were proposed.

In the following chapter these contributions will be integrated into a holistic framework for situation awareness. Tools to develop the building blocks from this chapter will be discussed, integrated in an architecture and implemented as a library.

4 Framework Design and System Implementation

This chapter aims to describe the situation aware system that meets the requirements identified in chapter 2. This includes the implementation of the theoretical concepts described in the previous chapter and furthermore addresses other relevant issues like sensor data acquisition.

4.1 Tools and Libraries

To implement a system that incorporates ontological knowledge, CST-based reasoning and access to sensor data different tools and libraries are required. Available options and selection of the appropriate tools will be discussed in this section.

4.1.1 Ontology Management

The development and programmatic use of ontologies is supported by various tools and libraries.

Ontology editors help to develop and maintain ontologies by providing an interface that visualises the concepts and generates corresponding files. A lot of graphical editors have been developed, for example Protégé[8], OWLGrEd[9] or Fluent Editor[10]. The selected editor for this thesis work is Protégé which has become the most widely used software for ontology development. It supports a flexible plug-in architecture which led to the development of various extensions and integration of other relevant tools within the editor [70].

Libraries have been developed to work programmatically with ontologies, e.g. extracting the defined statements, adding new facts or manipulating the terminology. The most prominent APIs are the Apache Jena Ontology API[11] and the OWL API[12]. Both APIs are open source Java frameworks and have an active community for support.

The Jena API is rooted in RDF but as shown, RDF can be used to develop ontologies. Furthermore, OWL can be serialized using RDF. The Jena API exploits this to wrap OWL level constructs over the RDF serialization. However, this limits the reasoning capabilities. According to the documentation the Jena Ontology API only supports OWL1 fully and provides limited support for OWL2.

[8] http://protege.stanford.edu/ - Accessed 03/05/2016

[9] http://owlgred.lumii.lv/ - Accessed 03/05/2016

[10] http://www.cognitum.eu/Semantics/FluentEditor/ - Accessed 03/05/2016

[11] https://jena.apache.org/documentation/ontology/ - Accessed 03/05/2016

[12] http://owlapi.sourceforge.net/ - Accessed 03/05/2016

In contrast, the OWL API is OWL centric. It is closely aligned with the OWL2 specification and supports like Jena parsing ontologies in different formats, manipulation and use of reasoning engines [71]. In fact, Protégé 4 and higher are realised with the OWL API. Working with the OWL API focuses more on axioms and expressions rather than RDF triples. As the approach described in chapter 3 is based on OWL2 concepts and the algorithms access the TBox, the OWL API is chosen for the implementation.

In order to take full advantage of the ontological specification an OWL reasoner needs to be applied. Reasoners for OWL DL that implement the corresponding interface of the OWL API are for example FaCT++[13], HermiT[14] or Pellet[15]. Reasoning engines have different capabilities, e.g. FaCT++ only provides partial support for OWL2 DL. An extensive investigation of reasoners was conducted in [72]. It shows that Pellet is the most capable reasoning engine by supporting all identified characteristics, e.g. soundness, completeness, incremental classification, rule support, justifications and ABox reasoning. HermiT does not provide justifications, incremental classification and performs worse for very large ontologies. Thus, Pellet is selected as a reasoning engine for the implementation.

Another helpful tool to work with ontologies are query languages. Query languages for ontologies are used to request a set of statements that correspond to a pattern, e.g. a triple pattern for RDF. This is comparable, for example, to the well-known Structured Query Language (SQL) which is used to manage data in relational database management systems. Examples for query languages are the SPARQL Protocol and RDF Query Language (SPARQL) for RDF and the Semantic Query-Enhanced Web Rule Language (SQWRL).

SPARQL is recognized as a standard by the W3C [73]. A query consists of triple patterns and also allows conjunction and disjunction. Since it is designed for RDF it does not have an understanding of OWL semantics, however, it can be used for basic RDF queries on OWL ontologies. In order to support OWL DL, SPARQL-DL [74] was developed as an extension to SPARQL. SPARQL-DL is designed to allow combination of ABox and TBox queries and to keep compatibility with OWL DL reasoners. The open source SPARQL-DL API[16] is based on the OWL API and completely supports OWL2 DL.

SQWRL is based on SWRL, the Semantic Web Rule Language, and built around OWL, not RDF. SWRL allows to specify logical rules about OWL individuals, allowing variables in these rules, in order to make further inferences about relationships in an ontology. In SQWRL these rules are used as patterns and all individuals for which the specified rule is

[13] https://code.google.com/archive/p/factplusplus/ - Accessed 03/05/2016

[14] http://www.hermit-reasoner.com/ - Accessed 03/05/2016

[15] https://github.com/Complexible/pellet - Accessed 03/05/2016

[16] http://www.derivo.de/ressourcen/sparql-dl-api.html - Accessed 03/05/2016

true will be part of the result set. In contrast to SPARQL, SWRL is not recognized as a standard but only as a member submission by W3C [75]. There are indications that SQWRL and SWRL are a less active effort in the community than SPARQL and SPARQL-DL. SQWRLQueryAPI[17], the SQWRL library based on the OWL API, for example, has not overcome the beta status yet. In conclusion, the SPARQL-DL API is selected to query the ontology in the implementation.

4.1.2 CST Implementation

Two undertakings to implement the Context Space Theory have been done, the Extensible Context Oriented Reasoning Architecture (ECORA) [76] and the Enhanced Context Spaces Theory-based Reasoning Architecture (ECSTRA) [77]. ECORA was developed to provide a general framework for context awareness, capable of handling uncertainty and a flexible architecture. ECSTRA is a successor of ECORA and has "extended support for multi-agent reasoning, enhanced support for sharing and reusing reasoning information and also enables integration of context prediction and proactive adaptation components" [77]. ECSTRA is implemented as a Java library with a flexible architecture and thus is chosen for the implementation as a CST-based reasoning engine.

In order to integrate ECSTRA in the overall system it is necessary to understand its architecture, which is depicted below in Figure 4.1.

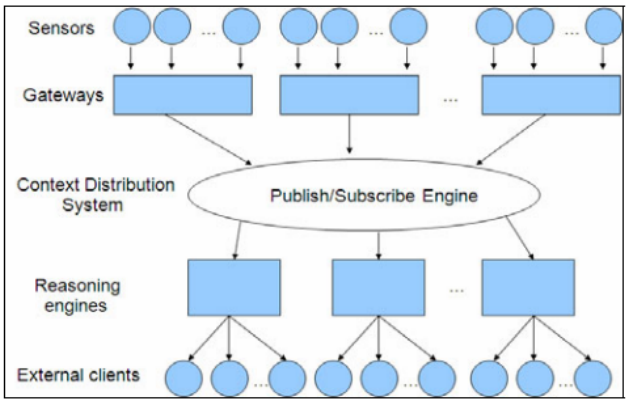

Figure 4.1 ECSTRA [77]

ECSTRA is usually based on a publish/subscribe engine to which sensor data is published via gateways. Reasoning engines consist of one or more reasoning agents which have a context collector, responsible for subscription to the context information and the calculation of context states, and further maintains the application space with defined situation spaces. With the calculated context state the reasoning results will be calculated and distributed to external clients.

[17] https://github.com/protegeproject/swrlapi/wiki/SQWRLQueryAPI - Accessed 03/05/2016

The architecture is very flexible in the way that it provides a generic platform proxy for the context distribution system. This enables various ways for the implementation of how context collectors receive context information. Implemented platforms in ECSTRA include for example the Avis Event Router[18], an open source implementation of the Elvin publish/subscribe protocol[19], and the Java Agent Development Framework (JADE)[20] platform in which case the context information is sent and received by JADE agents. In the case of a simple non-distributed application it is also possible to pass the context information directly to the context collector without an intermediate platform.

4.1.3 Sensor Data

The system needs to have the access to sensed data. Sensed data can be provided by various types of sources, even human-computer interaction (HCI) can be considered as sensed input. However, the current challenge is to realise situation aware systems in the context of the Internet of Things, i.e. integrating sensed values from smart assets. Since the goal of this thesis is to provide a general applicable framework to situation awareness, it is particularly important for the system to support standards, in this case standards for publishing and reading data from smart assets.

The H2020 bIoTope project[21], for example, is concerned with the development of an ecosystem for connected smart objects. One of the objectives is to provide necessary standardised APIs for the interoperability between smart objects of different platforms, e.g. OpenIoT[22] or FI-WARE[23]. Two Open API standards for the IoT to enable the publication, consumption and composition of the information from these various sources are mentioned, the Open Messaging Interface (O-MI) [78] and the Open Data Format (O-DF) [79].

O-MI defines a set of possible interactions between entities, developed based on the lifecycle consideration of smart objects [80]. O-MI messages can be exchanged on top of well-known protocols like HTTP, SOAP or SMTP. The communication protocol for O-MI supports the operations *read* for information retrieval (including subscription), *write* to send information, and *cancel* to unsubscribe. It is based on the observer design pattern, i.e. O-MI nodes are capable of observing events of other nodes in a publish/subscribe as well as a peer-to-peer manner, which is of high importance for smart objects in IoT [78].

[18] http://avis.sourceforge.net/ - Accessed 04/05/2016

[19] http://www.elvin.org/specs/index.html - Accessed 04/05/2016

[20] http://jade.tilab.com/ - Accessed 04/05/2016

[21] http://biotope-project.eu/ - Accessed 04/05/2016

[22] http://openiot.eu/ - Accessed 30/05/2016

[23] https://www.fiware.org/ - Accessed 04/05/2016

The payload format of the messages is not restricted by the O-MI standard. O-DF was proposed to provide a generic structure for IoT payload information, which can be used for O-MI messages [80]. Its specification is as well based on the lifecycle consideration of smart objects and aims to represent information about these in a standardized way. The O-DF structure is defined with XML Schema and defines a hierarchy of *Object* elements whereas each *Object* can consist of various *InfoItem* elements and other *Objects* [79]. The O-DF element hierarchy is illustrated below in Figure 4.2.

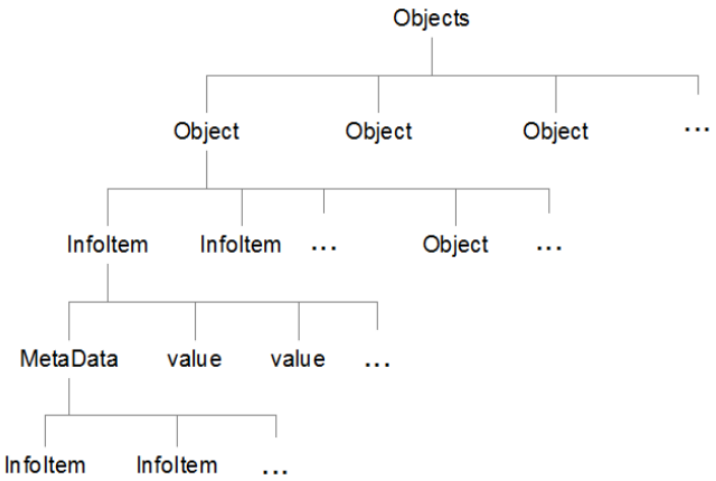

Figure 4.2 O-DF Element Hierarchy [79]

In the interest of positioning the proposed framework of this thesis in the context of IoT the access of sensed information will be realised based on O-MI and O-DF. These are defined as generic standards for the communication between smart assets and hence are suitable to be integrated in the overall architecture for the situation aware system.

4.2 Framework Architecture

In this section the architecture of the framework for situation awareness will be described from a conceptual point of view, based on the tools, libraries and standards selected in the previous section.

The proposed framework architecture which is built around the enhanced CST concepts presented in chapter 3 is shown in Figure 4.3 below. It is composed of the following key components.

Knowledge Base. The knowledge base consists of CSTO-based application ontologies. Multiple ontologies with different situation specifications and the application setup can be provided for the system.

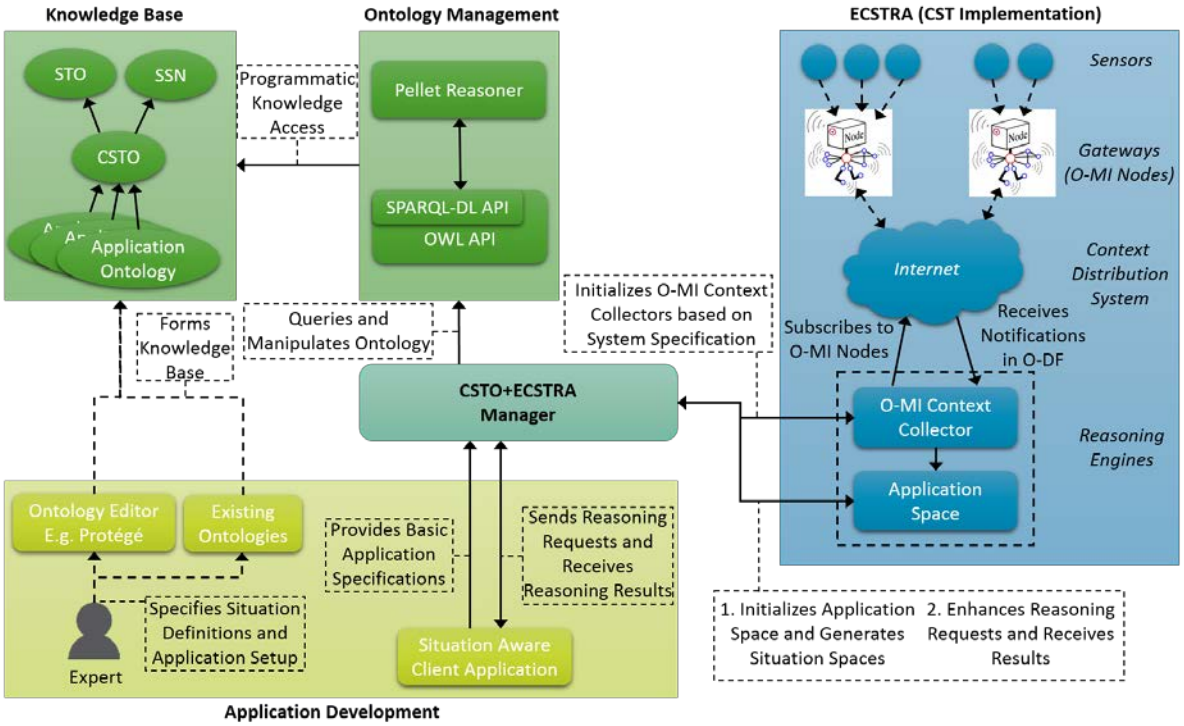

Figure 4.3 High-level Architecture of the Proposed Framework

Ontology Management. The ontology management component is responsible to allow programmatic access and manipulation for the knowledge base. It is composed by the previously selected tools for ontology management, i.e. OWL API, SPARQL-DL API and the Pellet reasoner.

ECSTRA. The ECSTRA architecture is adjusted for the O-MI/O-DF integration. Instead of subscribing to a central publish/subscribe engine, the context collectors subscribe directly to the O-MI nodes and receive the notifications in O-DF.

CSTO+ECSTRA Manager. This component binds all parts together. It acts as a proxy for the application towards the reasoning engine and the knowledge base. The tasks of this component are twofold.

Firstly, it is responsible for the initialisation of the system. This includes the automated ontology population based on situation types and the creation of an application space by generating the situation spaces based on the situation specifications as it was presented in section 3.3. Furthermore it initialises the O-MI context collectors based on the given specifications by the application. The context collectors are created and subscribe to the corresponding O-MI nodes.

Secondly, it serves as a proxy for an application during runtime. The component is responsible to enhance the reasoning requests, i.e. adding the formal support of involved individuals in a situation, as identified as missing in CST (see section 3.2.1). Thus it will

access the knowledge base during runtime to identify the relations between situations, individuals and sensors and coordinates the context collectors to update the context state in the application space. Eventually the client application will be informed about the reasoning results.

Application Development. This part of the architecture shows the necessary steps in order to develop a situation aware application based on this approach. The process is further illustrated in Figure 4.4.

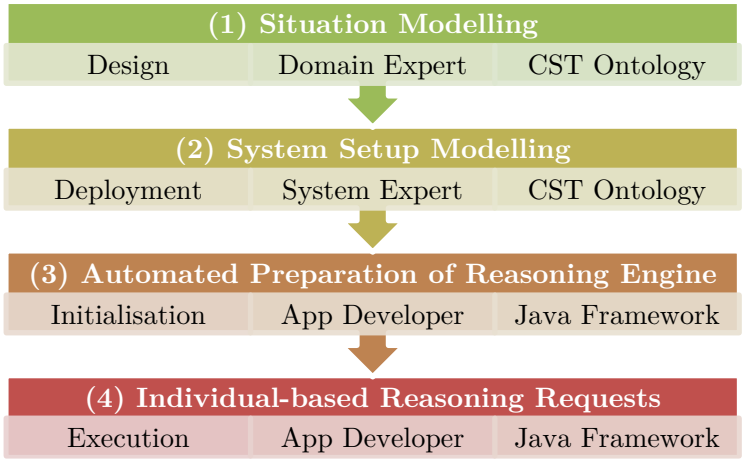

Figure 4.4 Process of Application Development

Initially a domain expert has to develop the knowledge base by specifying situation definitions (or reusing existing ones) (1) and the application setup (2) with CSTO-based ontologies. In order to correctly work with the CSTO+ECSTRA Manager, necessary information, e.g. ontology locations and O-MI node access, have to be specified. After starting the initialisation process (3), the application can send the reasoning requests and act upon reasoning results (4).

4.3 System Implementation

The system implementation of the proposed framework architecture incorporates the implementation of the ontology and the CSTO+ECSTRA manager. The objective is to provide a Java library that can be imported to develop client applications.

4.3.1 Ontology Development

The Protégé ontology editor was used to develop CSTO. It is necessary to import the relevant upper ontologies and to map the concepts as discussed in section 3.2. A screenshot of the ontology in Protégé 4 is shown below.

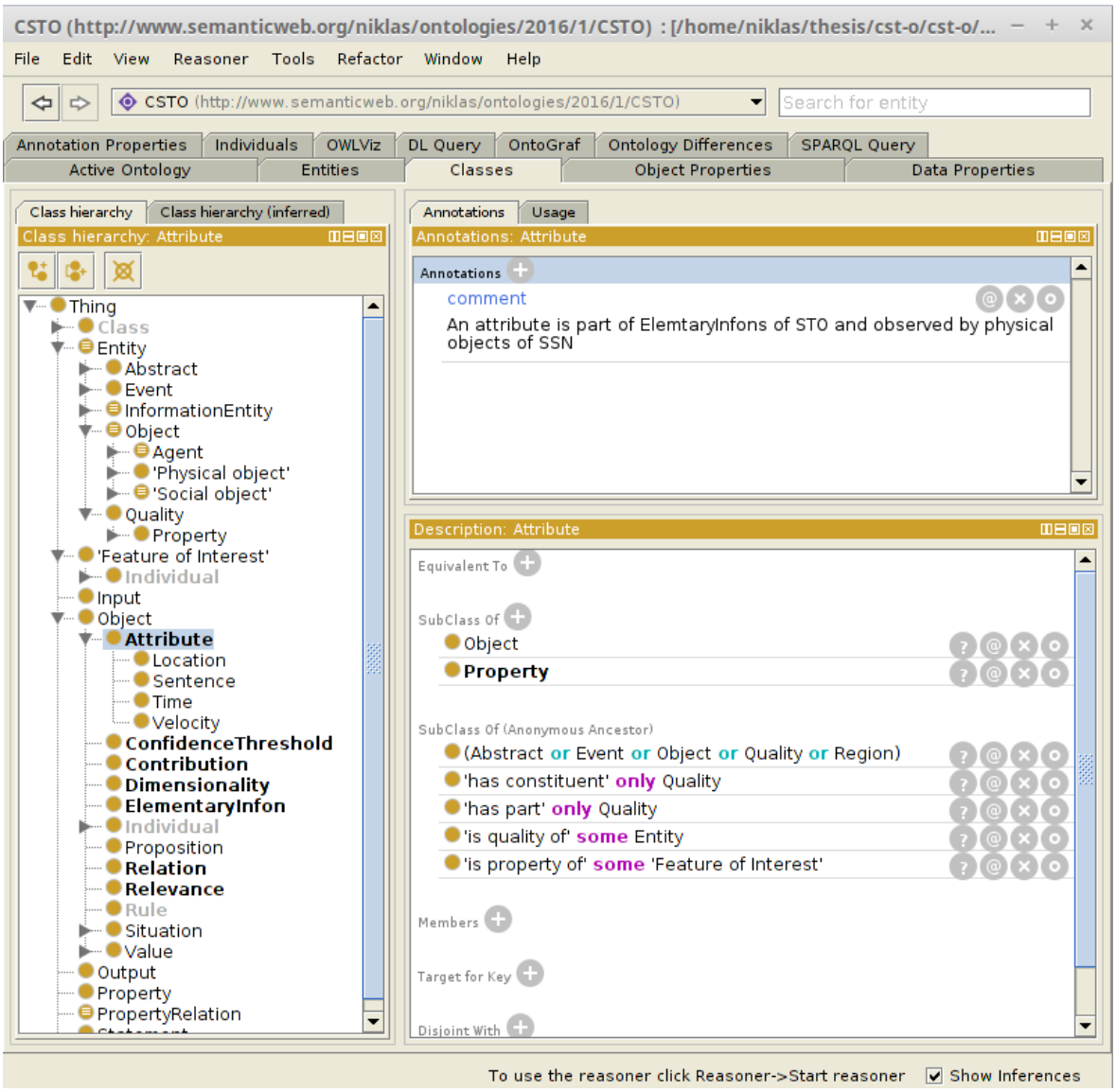

Figure 4.5 CSTO developed with Protégé

Bold entries indicate definitions made in the CSTO ontology, others originate from imported ontologies. For example, the axiom `sto:Attribute owl:subClassOf ssn:Property` was added to map the STO attribute and SSN property classes. The complete ontology with a complete mapping of all STO and SSN concepts is appended in appendix A.1 in RDF/XML format.

4.3.2 Package Description

The system is composed of four major components, namely manager, ontology, situation and context. The basic dependencies of these components are depicted in Figure 4.6.

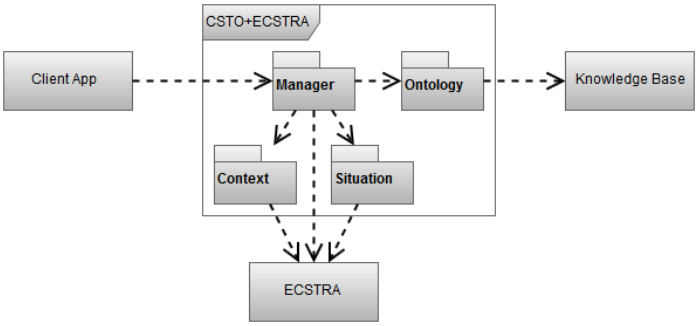

Figure 4.6 Concept of System Modules with External Dependencies

The task of the *manager* is to provide a façade to the client application and to act as an intermediary element between the other components. The *ontology* module is concerned with the access to the knowledge base. The components *context* and *situation* handle the initialisation of ECSTRA elements, i.e. the context collection (context) and the application space initialisation (situation).

Figure 4.7 shows the package diagram from an implementation point of view.

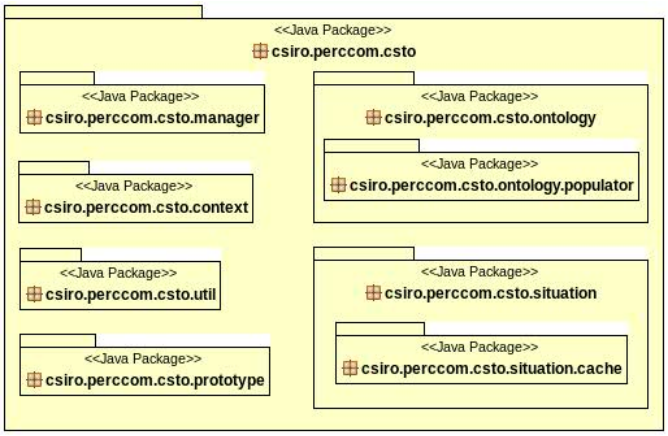

Figure 4.7 Package Diagram

Table 4.1 below contains the description for each package. The individual packages and their relations will be presented in more detail in the following section.

Package	Description
csiro.perccom.csto.manager	Contains the central manager class.
csiro.perccom.csto.context	Classes to handle context collection.
csiro.perccom.csto.ontology	Classes to handle ontology access.
csiro.perccom.csto.situation	Classes to handle application spaces.
csiro.perccom.csto.util	Classes with helper functions.
csiro.perccom.csto.prototype	Contains examples for client applications.

Table 4.1 Package Description

4.3.3 Class Diagram

Classes that are members of the aforementioned packages are depicted in an UML class diagram in Figure 4.8. The diagram includes generalisations and selected associations between the classes. The most important classes will be briefly explained in the following.

- **Manager.** The *CSTManager* implements the façade software design pattern with the aim to make the library easy to use and to reduce dependencies in client applications. Thus it serves as a container to maintain the most important objects in the system, e.g. the parsed ontology (OWL API), the reasoner (Pellet), the application space (ECSTRA), context collectors etc.
- **Prototype.** The *Prototype*-classes represents example client applications that use the depicted library. The CSTManager is initialised with a configuration about the ontologies and context collectors.
- **Situation.** The *SituationSpaceGenerator* reads the ontology, stores the information in a cache and then generates the situation spaces in the application space. As described in sections 3.3.1 and 3.3.2 this can be based on situation object and situation type definitions.
- **Populator.** The *SituationObjectsPopulator* is initialised with the reasoned and merged ontology and creates new situation objects based on types, as presented in section 3.3.3.
- **Context.** The *ContextCollector* classes are created by the CSTManager based on the configuration. Two ways of O-MI/O-DF integration are implemented. The first option is via a Java Servlet, i.e. instance that answer requests sent to a web server, which is handled within the application. The second option is to parse a XML file in O-DF format which is externally updated, for example by an external Java Servlet. The required configuration depends on the chosen option, either parameters for the O-MI subscriptions or the path to the O-DF files. This module is attached to the ECSTRA context collector classes.
- **Util.** The *Util*-classes provide various static helper functions regarding stream, string and ontology related aspects.
- **Ontology.** The *SPARQLDLQueryHandler* provides access functions to the query engine of the SPARQL-DL API. The *CSTOHelper* provides functions to access information specified in CSTO ontologies. The *NamespaceDictionary* provides quick access to namespace strings of the upper ontologies.

It is worth mentioning that the architecture is designed in a way that applications can extend the default classes provided by the library or develop individual implementations and set custom objects via the *CSTManager* instance.

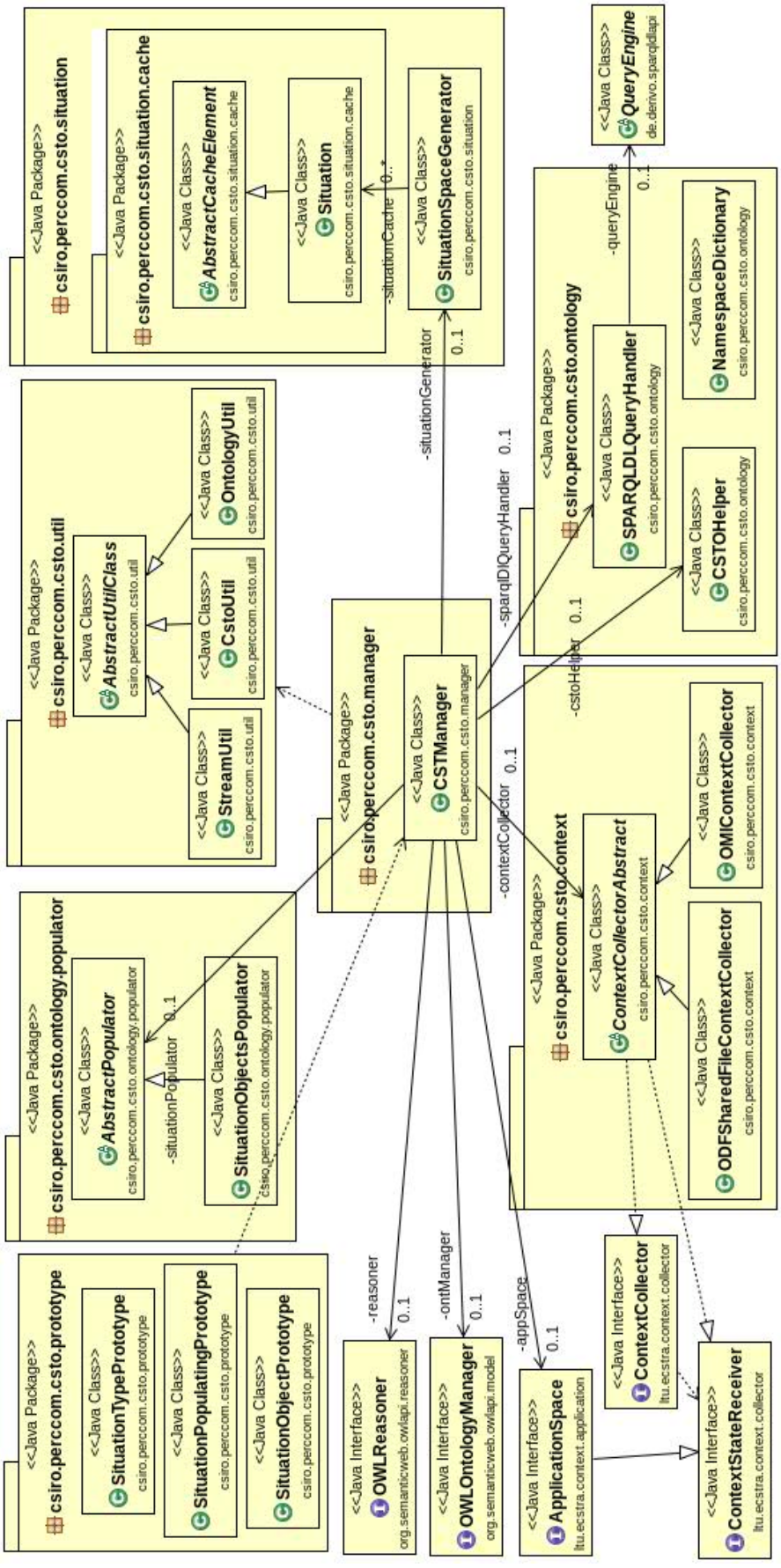

Figure 4.8 Class Diagram with Selected Dependencies

4.3.4 Execution Flow

This section presents the major execution flow of the system, showing the sequences for the initialisation process and for runtime requests.

Figure 4.9 illustrates the execution flow for runtime reasoning. The *CSTManager* acts as an intermediary layer to resolve dependencies based on involved individuals and automatically updates the corresponding context state in the application space. Afterwards it requests the corresponding reasoning result from the ECSTRA implementation and returns the result to the client application.

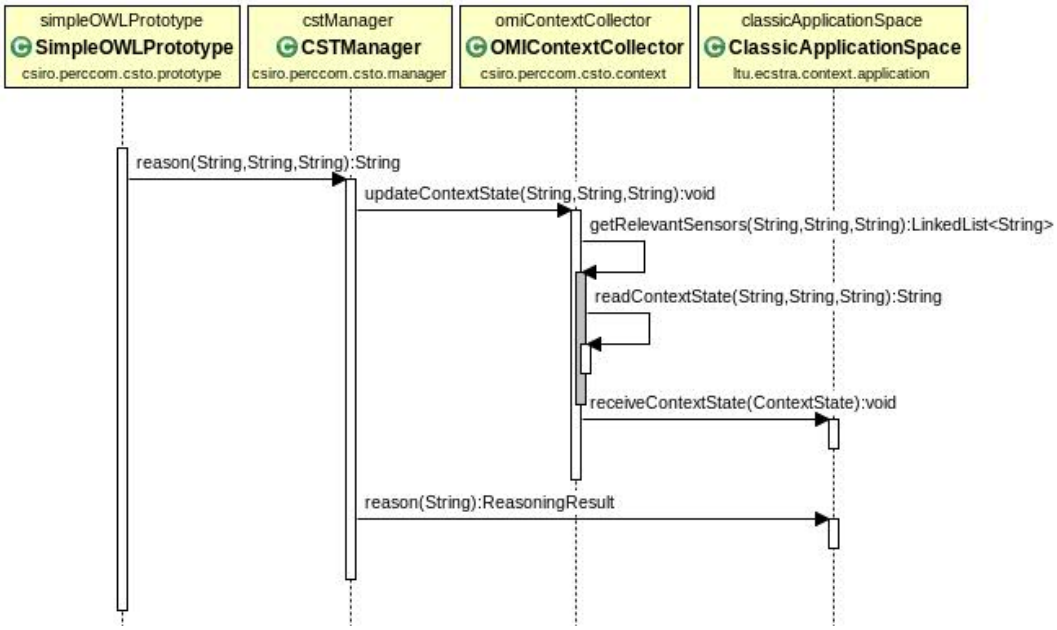

Figure 4.9 Sequence Diagram for Runtime Reasoning

The execution flow shown in Figure 4.10 highlights the major communication steps between key components to initialise the system.

The *SimpleOWLPrototype* represents a client application. It provides the basic configuration and starts the initialisation process. The *CSTManager* coordinates the whole process by preparing the knowledge base, further processing it, setting up the application space and creating context collectors.

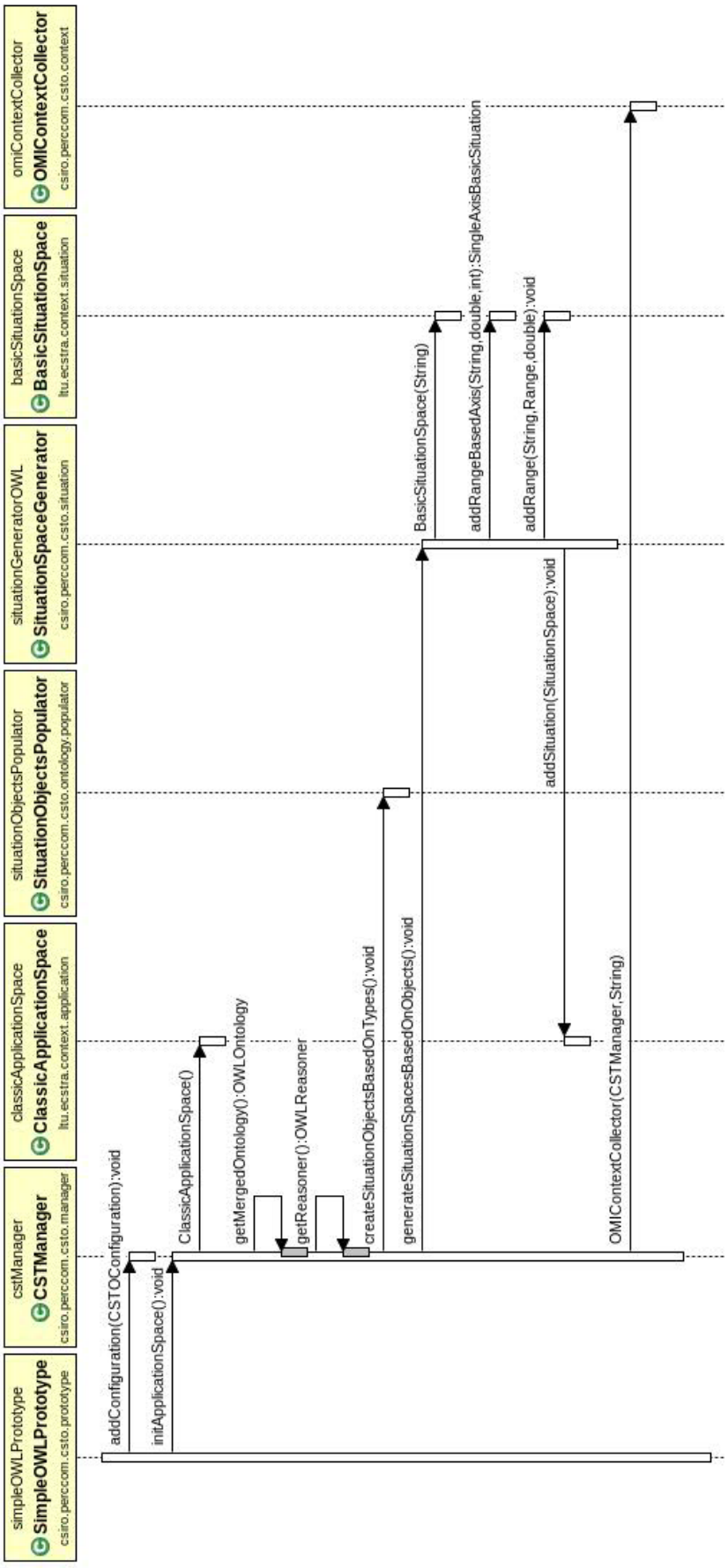

Figure 4.10 Sequence Diagram of Key Steps for System Initialisation

4.4 Summary

This chapter introduced the tools selected for the implementation, presented the architecture of the framework and described the system implementation. The result of the work is a Java library that enables the development of situation aware client application with an ontological knowledge base.

In the upcoming chapter the result will be applied to design and develop a situation aware system.

5 Use Case: Food Sharing Neighbourhood

The use case implementation is based on the scenario introduced in section 1.2. The objective is to provide a proof-of-concept. Firstly, to prove the applicability of the previously proposed framework and implemented system. Secondly, to provide a showcase for the development of a pervasive system that contributes to sustainability. The use case is concerned with the development of a situation aware application which gives optimal recommendations to users about the usage of food items to reduce food waste.

5.1 Overview

A web application which is based on the situation aware framework will be developed to provide users in this neighbourhood with recommendations about the food usage. Thus, corresponding situations based on the available sensed data mentioned above, and the system setup of this scenario will be defined in an ontology. Upon user requests the system reasons about the situations given the current context state accessed via O-MI/O-DF. Besides just identifying the urgency of consumption of the available items the system furthermore proposes recipes and shows an indication of the environmental impact that is achieved by preventing a particular food item from being wasted.

In this section the use case will be framed through formulated assumptions. Furthermore it is discussed how the sustainability aspect will be integrated into the application.

5.1.1 Assumptions

This use case is based on the following assumptions:

- The implementation is based on a simulated smart neighbourhood, composed of three households.
- Each household generates (simulated) sensor data values. To sense information about food items, it is assumed that each item is labelled with an RFID tag and smart fridges are equipped with RFID readers, to read these tags when items are placed inside the fridge.
- Information stored in the RFID tags includes the available amount of items and related expiration date.
- Sensor data providing information about when and how to access food items is simulated. For example, this input can be simulated based on human being's activity in the household or on the availability of smart access devices (e.g. smart locks like slock.it[24]).

[24] https://slock.it - Accessed 30/05/2016

- All sensor values are published through an IoT/neighbourhood avatar (an O-MI node in this case) that aggregates and publishes neighbourhood-related information in a standardised manner.

5.1.2 Sustainability Index

Giving recommendations about food usage to users regarding the expiration and available amount has priority. However, the impact of food waste depends from item to item, due to the difference in production, transport, storage, etc. In order to raise awareness about food waste impact, a sustainability index will be displayed to the user which is based on the carbon footprint, blue water footprint and economic cost of particular food items.

From a global perspective it is impossible to measure the impact of each food item. Thus the index displayed is based on agricultural commodity groups. FAO published a report about the impact of food waste based on these commodity groups [8]. The share of the commodity group with an equal weight for all three criteria on the overall food waste impact is used to define a sustainability index. Table 5.1 shows the contribution of commodity groups on the overall food waste.

Commodity Group	Carbon Footprint (%)	Blue Water Footprint (%)	Economic Cost (%)	Average Share (%)
Cereals	34	51	17	**34**
Starchy roots	5	2	9	**5.3**
Oil crops & Pulses	2	5	3	**3.3**
Fruits	6	18	19	**14.3**
Meat	21	8	22	**17**
Fish & Seafood	4	-	-	**1.3**
Milk & Eggs	7	8	7	**7.3**
Vegetables	21	8	23	**17.3**

Table 5.1 Sustainability Impact Share of Commodity Groups based on [8]

The sustainability index is calculated for proposed recipes. The major idea is to present an indication of the sustainable contribution and value of cooking a particular recipe through the implied food waste prevention. Each ingredient of the recipe that is recommended for usage is considered with the average share of the environmental impact of its commodity group for the calculation. The more ingredients that are recommended to be used involved in the recipe and the higher their impact share is, the higher is the sustainability index.

5.2 System Design

This section discusses the used tools and libraries, and the architecture to realise the system that was introduced previously.

5.2.1 Tools and Libraries

The application is developed as a JVM-based web application with the Spring framework[25]. In particular Spring Boot[26] and the Spring Web MVC framework[27] are used to form the basic structure of the web application. The frontend is realised with common web development tools like jQuery[28], Bootstrap[29] and Thymeleaf[30].

The O-MI node is implemented as an IoT data server, developed by Aalto ASIA[31]. It is realised in Scala and O-MI nodes are implemented as agents. The O-MI/O-DF integration was implemented in cooperation with the Security and Trust department of the University of Luxembourg.

As this application is based on the framework proposed in this thesis, the developed Java library and its dependent tools (ECSTRA, OWL API, Protégé, etc.) are used for the development.

Furthermore the system proposes recipes based on the food items that are recommended for usage. These are gathered via the RESTful Yummly Recipe API[32].

5.2.2 System Architecture

Figure 5.1 shows how the MVC architecture of the application is integrated in the overall framework. The developed library from chapter 4 is represented as a single element (*CSTO+ECSTRA Manager*). The *CSTManager*, which manages the communication between all the components, is initialised on the application start up and kept as a singleton inside a model of the application to provide a continuous access for controllers to the reasoning engine.

The sensor values are simulated by the O-MI agent. A Java servlet on the web server subscribes to the O-MI node and updates a XML file stored on the server. The newly added context collector of ECSTRA parses the file in O-DF format to retrieve latest sensor values.

[25] http://spring.io/ - Accessed 09/05/2016

[26] http://projects.spring.io/spring-boot/ - Accessed 09/05/2016

[27] http://docs.spring.io/autorepo/docs/spring/3.2.x/spring-framework-reference/html/mvc.html - Accessed 09/05/2016

[28] https://jquery.com/ - Accessed 09/05/2016

[29] http://getbootstrap.com/ - Accessed 09/05/2016

[30] http://www.thymeleaf.org/ - Accessed 09/05/2016

[31] https://github.com/AaltoAsia/O-MI/wiki - Accessed 09/05/2016

[32] https://developer.yummly.com/ - Accessed 30/05/2016

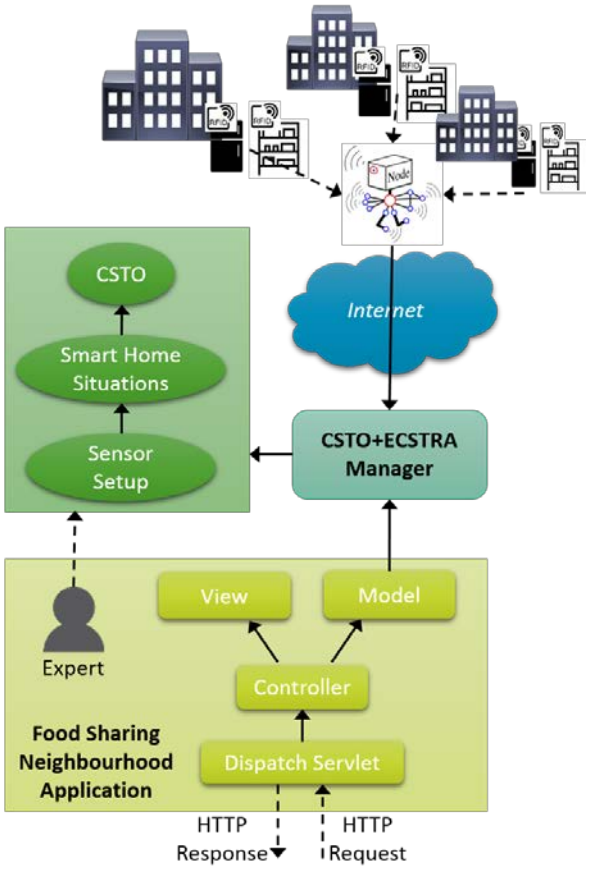

Figure 5.1 System Architecture of the Food Sharing Neighbourhood Application

Upon HTTP request reasoning requests will be sent to the *CSTManager* and the computed results will be processed in the controller and view to generate the HTTP response.

5.3 Implementation

The section presents the specification of situations and the system setup for the *Food Sharing Neighbourhood* application. As this web application is presented solely as a use of the proposed framework, the implementation of the application is not of great interest and will not be described in detail. Instead the final output with recommendations based on situation awareness is shown.

5.3.1 Situation and System Modelling

A key situation that will be defined for the system is called *FoodItemUsageRecommended,* referred to as SIT_{UR}. The modelling of the situation in CSTO will be explained in more detail to demonstrate the situation modelling process. It is assumed that the situation is based on two infons, a food item should be used if the expiration date is close or if the available stock is relatively high. (18) shows the corresponding situation type definition in situation theory which is intended to be modelled in CSTO to generate a situation space.

$$[\ \dot{SIT}_{UR} \mid \dot{SIT}_{UR} \vDash \ll expires, \dot{f}, \dot{e}, 1 \gg \wedge \ll stock, \dot{f}, \dot{s}, 1 \gg]$$
$\dot{f}$: parameter for food items, $\dot{e}$: parameter for soon expiration dates, (18)
$\dot{s}$: parameter for relative high stock in a household

In CSTO further the acceptable regions, contribution, relevance and the confidence threshold have to be specified.

The modelling of the situation type in CSTO is shown in (19).

```
FoodItemUsageRecommended owl:equivalentClass (
  sto:Situation
  and (sto:supportedInfon owl:value FoodItemExpiringInfon)
  and (sto:supportedInfon owl:value FoodItemHighStockInfon)
  and (sto:relevantIndividual owl:some Fooditem)
  and (csto:hasConfidenceThreshold owl:value UsageCT)
)
```
(19)

Only the *FoodItemAboutToExpireInfon* will be explained in more details for demonstration purposes. (20) shows its representation in OWL. The polarity values provided by STO are named *_1* for true and *_0* for false.

```
FoodItemAboutToExpireInfon owl:equivalentClass (
    sto:ElementaryInfon
    and (sto:relation owl:value expires)
    and (sto:anchor1 owl:value Fooditem)
    and (sto:anchor2 owl:value ExpiringShelfLife)
    and (sto:polarity owl:value _1)
    and (csto:hasRelevance owl:value FoodExpiringRelevance)
)
```
(20)

The OWL individual attribute *ExpiringShelfLife* of the context attribute class *ShelfLife* has two acceptable regions (21).

```
ExpiringShelfLife rdf:type ShelfLife
ExpiringShelfLife sto:hasAttributeValue ExpiringSoonValue
ExpiringShelfLife sto:hasAttributeValue ExpiringPromptlyValue
```
(21)

The acceptable regions consist of an interval and a contribution value. The closer a product comes to its expiration date, the bigger is the contribution to recommend the usage.

(22) shows the *ExpiringSoonValue* acceptable region for the shelf life context attribute, which is defined as an interval of one to three days. The syntax of interval specification is inherited from ECSTRA.

```
ExpiringSoonValue rdf:type ShelfLifeValue
ExpiringSoonValue sto:attributeValue "(1.0,3.0]"                    (22)
ExpiringSoonValue csto:hasContribution ExpiringSoonContribution
```

Other situations, infons, attributes, contributions etc. are modelled in the similar way. Situations modelled during the development of this prototype include for example *FoodItemRecyclingRecommended* and *UserAvailable*. The complete ontology with situation definitions can be found in appendix A.2.

The specification of the system setup is modelled in a separate ontology to allow reuse of the situation definitions. Modelling the system setup is a straight-forward process. The most important step is to instantiate all sensors and to specify which attributes of which individuals they observe. (13) shows the specification for a fridge RFID reader. Due to the nature of RFID, a single sensor can capture multiple attributes of multiple individuals.

```
RFIDSensor001 rdf:type FridgeRFIDSensor
RFIDSensor001 ssn:observes ShelfLife
RFIDSensor001 ssn:observes Stock                                    (23)
RFIDSensor001 csto:observesPropertyOf Fooditem
```

The complete ontology for the system specification is given in appendix A.3.

5.3.2 Emulation of System Environment

In order to give recommendations about the food usage it is necessary to provide input about the available food items. These sensor values are emulated with a Java application. The GUI of the application is shown in Figure 5.2.

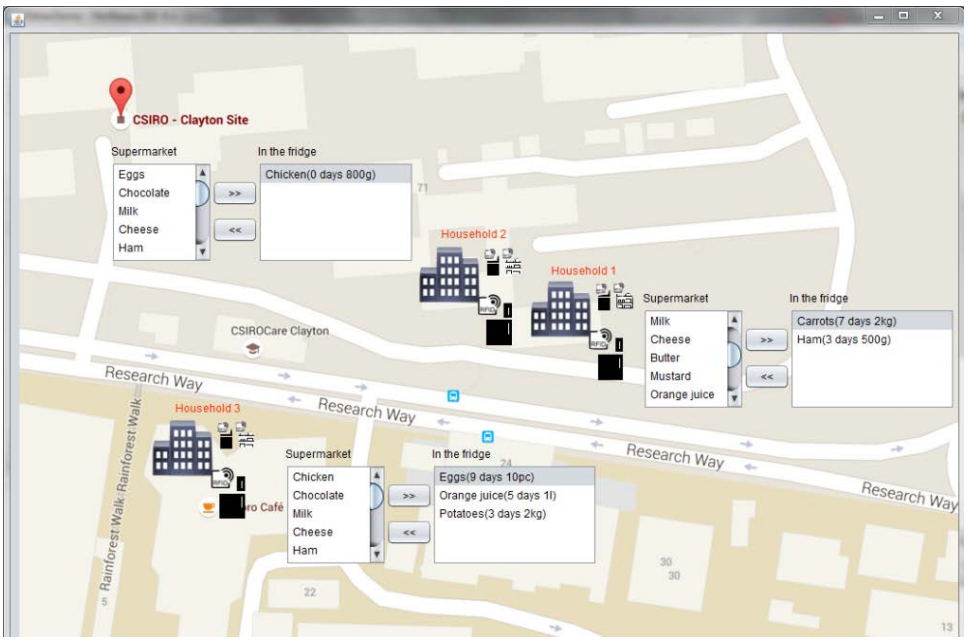

Figure 5.2 Emulation of Sensor Data

The application provides a graphical interface which represents the discussed neighbourhood. It enables the user to configure the smart fridges of the system. For each fridge items can be added and removed and for each available food item the amount and the expiration can be defined.

In the background of the application an O-MI node is running which publishes data based on the current configuration of the GUI.

5.3.3 End-user Interface

The user interface gives an overview of the neighbourhood, shows the recommendations of food usage (locally and close by), the sustainability index and proposes appropriate recipes. The screenshot in Figure 5.3 shows a screenshot of the application with an example of the final output for a user request.

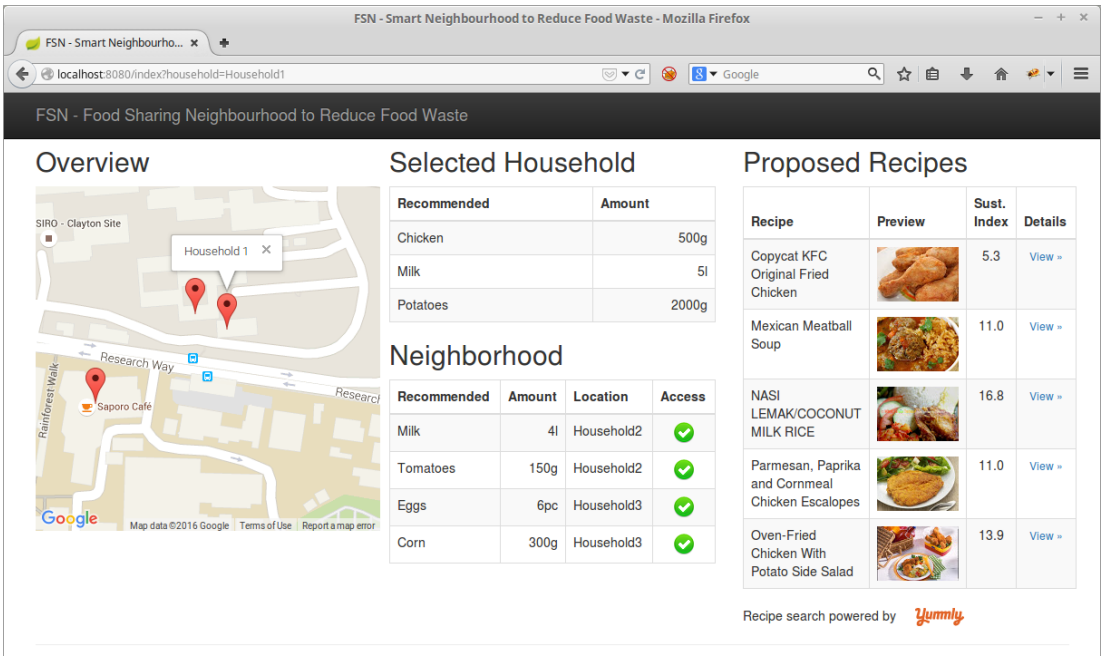

Figure 5.3 User Interface of the FSN Application

5.4 Summary

This chapter introduced a use case for the developed framework for situation aware applications with the global sustainability goals in mind. The developed web application gives recommendations about the best usage of food items to reduce waste during consumption stage. Furthermore it proposes recipes based on these ingredients and shows the environmental and economic implications of not wasting particular food items to motivate a behavioural change.

6 Discussion of Results

This chapter discusses the framework regarding the requirements identified for situation aware systems from section 2.3.1. The argumentation is supported by the implementation of the use case presented in the previous chapter.

The proposed framework is based on Situation Theory for situation modelling, ontologies for knowledge management and Context Space Theory for inference. Thus the framework combines the advantages of all these different approaches and fulfils the identified requirements regarding situation, knowledge and reasoning, respectively.

Furthermore, with the integration of O-MI/O-DF, a general integration of sensor data acquisition is achieved. However, actuators so far were considered for system modelling but not yet explicitly for the framework.

The implementation of the use case demonstrates the applicability of the framework to develop applications. With its central knowledge base, expert knowledge can be easily integrated and with a central managing component client applications can easily access the reasoning functionalities and all other components involved in the system. The use case also validates the frameworks regarding the aforementioned aspects. Furthermore the use case demonstrates the enabling effect of the framework to develop pervasive systems that are capable of contributing to the sustainability goals.

Runtime performance is determined through the reasoning process. The framework contributed to two major aspects regarding the reasoning process: The automated initialisation of the context space and a framework for handling involved individuals in situations. The performance of the initialisation process is not relevant because it is not performed during the runtime stage. Handling involved individuals however affects the reasoning process.

It is important to notice that the support of individuals is an additional feature which does not have to be used by client applications. Reasoning requests can be directly send to ECSTRA via the *CSTManager*. In this case, situation types should be defined in a way that generates situation spaces for each involved individual (presented in 3.3.3). ECSTRA performance has been evaluated to support up to 28000 situation spaces efficiently. A performance test for the proposed framework was performed to identify the added computational impact when it comes to finding sensors in the ontology based on the involved individuals and relevant context attributes in the situation.

For the performance analysis the ontology was populated with test values in increasing size and the computing time of retrieving corresponding values from the ontology was measured. In order to simulate realistic situation definitions and to have an accurate distribution

of elements in the ontology, axioms not only for the situation individuals but also for corresponding infons, attributes, individuals and sensors were created to represent an accurate distribution of elements in the ontology (as presented in chapter 3). A single situation definition is thus comprised of 18 axioms. Table 6.1 shows how many axioms in the ontology were created based on the generation of test situations.

Situations	Axioms
10	2580
100	4200
500	11400
1000	20400
1500	29400
2000	38400
2500	47400
3000	56400
3500	65400
4000	74400
4500	83400
5000	92400
5500	101400
6000	110400
6500	119400
7000	128400
7500	137400

Table 6.1 Test Data

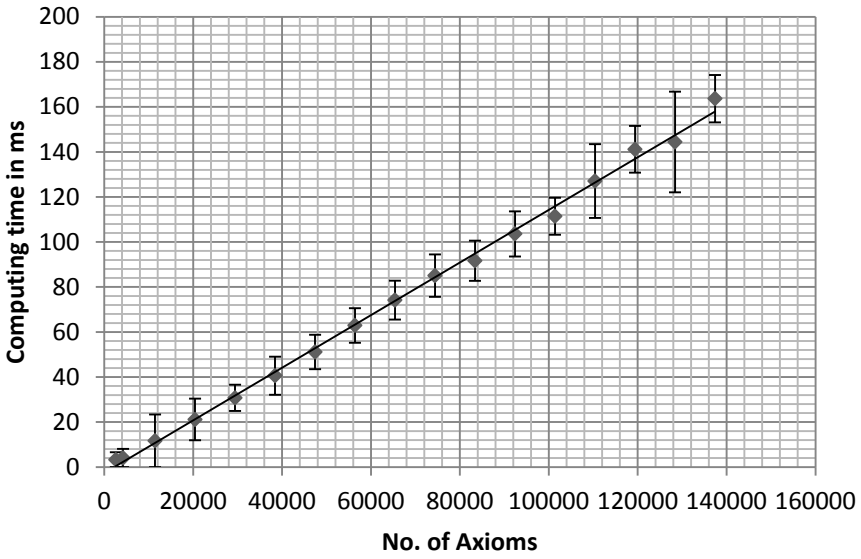

Figure 6.1 Performance for Ontology Access during Runtime Reasoning

Figure 6.1 shows the result of the performance test. For each data set 1000 measurements were performed. The figure shows the average value and the standard derivation of the measurements. The test was performed with 6GB heap space and an Intel Core i7 CPU on a Windows 64 bit machine.

Two major aspects can be concluded from the performance test. First of all, reasoners have an acknowledged heap space problem. A measurement with 8000 situation created a `java.lang.OutOfMemoryError: Java heap space` exception. Furthermore the measurements indicate an acceptable complexity of $O(n)$, thus the runtime increases linear with the size of the ontology. This may impact the performance for very large scale applications, if the feature to reason based on involved individuals is applied.

Table 6.2 summarizes this discussion of the thesis results. The criteria and rating are based on the comparison from chapter 2. The evaluation shows that the framework meets all mentioned criteria, except regarding actuators and some performance aspects. In comparison to related work presented in Table 2.2 it can be seen that this framework covers a

broad range of criteria. By the combination of related approaches and making use of the strength of these approaches the overall framework fulfils the identified requirements.

Criteria		Rating	Remarks
Situation	*Space and Time*	+	As the modelling is based on Situation Theory the framework fulfils all requirements regarding situation representation.
	Roles of Objects and Situation Types	+	
	Relations	+	
Knowledge	*Situational Knowledge*	+	As the situations and the system specifications are modelled in ontologies the framework fulfils all requirements regarding knowledge incorporation and reuse.
	System Knowledge	+	
	Knowledge Reuse and Sharing	+	
Reasoning	*Overall Capability and Universal Applicability*	+	As the approach is based on Context Space Theory, it fulfils the requirements towards enhanced reasoning capabilities.
	Uncertainty and Temporality	+	
	Prediction	+	
Application	*Sensor Data Acquisition*	+	Sensor data acquisition in IoT context was achieved by integrating O-MI/O-DF. Actuators have only been considered for system modelling. The performance of some (optional) features of the framework may be affected for large scale.
	Actuators	~	
	Performance and Applicability	~	

Table 6.2 Evaluation of the Proposed Framework

The work on this thesis was partially motivated by the improvement of expert knowledge integration into situation aware approaches. The framework itself does not provide a complete consideration of the knowledge engineering process. However, by founding the knowledge base on a solid, proven technical approach, i.e. ontologies, the framework provides a platform to specify, maintain, share and reuse defined knowledge. Moreover, Situation Theory provides a standardized approach to specify the situation definitions. With these approaches, errors and inconsistencies in the knowledge base are less likely and can be easier identified and solved.

7 Conclusion and Future Work

This thesis is based on an intensive study of situation aware approaches. As a result of this discussion, the combination of Situation Theory and Context Space Theory was motivated by automatic generation of Situation Spaces in CST with knowledge specified based on Situation Theory. To allow a formal process for knowledge specification and sharing, the approach is based on ontologies. STO, SSN and SAN were combined and extended to fit as a core ontology for a CST-based system. The theoretical contribution further includes proposed algorithms to extract the knowledge from the ontology and initialise the application space.

Further discussion led to a design of an overall framework based on the proposed core ontology and ECSTRA for CST-based reasoning. In order to meet the requirements for platform independent sensor data acquisition the IoT standards O-MI/O-DF were integrated into the system. The thesis contribution is a Java library which was designed to allow an efficient use of these components to develop situation aware applications.

As a proof-of-concept the framework was applied to a use case, which validated the feasibility of the approach. By showcasing a system to reduce food waste at consumption stage it demonstrated the enabling effects of situation aware systems regarding the contribution to sustainability.

The thesis objective was to provide a general framework for situation awareness with rich reasoning capabilities which can be applied efficiently to develop individual applications. This chapter concludes the work, presents limitations and opportunities for future improvements.

7.1 Conclusion

Several points can be concluded from this work. First of all, it is shown that the concepts for situation representation of Situation Theory and situation spaces of Context Space Theory can be mapped. Algorithms further showed how representation can be translated from one representation to the other.

Furthermore, this approach integrated an ontological knowledge base to CST-based reasoning. Ontologies, being a well-established semantic web technology, foster knowledge sharing and reuse. This supports the initial process of the definition of the knowledge base for the situation aware application. By selecting a holistic approach and covering not only knowledge about situations but also about the system setup it was possible to design and

implement a framework which is capable of providing situation awareness solely based on a provided knowledge base and configurations for sensor data acquisition.

Incorporating related standards allows interoperability of the system. The flexible design allows individual implementations, but by providing default components, e.g. based on O-MI/O-DF, it promotes a coherent embedment in the IoT environment.

The final discussion indicates that the framework aligns with most of the previous identified requirements for situation aware systems. However, limitations include the holistic consideration of actuation and the performance of additional ontology-related features for large scale systems.

7.2 Future Work

The proposed framework forms a holistic foundation for the development of situation aware applications. This foundation provides further opportunities for improvement of reasoning related aspects.

The ontology specification can be extended to define more relations of situations, for example that situations may not occur at the same time. These definitions can be used to verify the model during situation space generation.

The framework can be extended with an ontology database, e.g. a triple store or an OWL database which may provide a more efficient access to the ontology during runtime, tackling the performance issue for very large scale systems

Another interesting aspect is to consider a more dynamic environment. This concerns individuals entering or leaving the environment during runtime and also the discovery of new sensor sources entering the system.

The proposal in [69] to define rules about situations for the generation process can be adopted for the framework. Future work could include the implementation of this approach into the system, allowing situation specification based on algebraic operators.

Furthermore generating situation spaces based on incomplete knowledge about situations can be investigated more thoroughly, for example by adopting the Fuzzy Situation Theory Ontology.

The integration of actuators has only been considered for modelling of the system. Future work includes the integration of actuators, similar to the integration of sensors, for the system implementation.

References

[1] Gartner Inc. (2015, Nov 10). *Gartner Says 6.4 Billion Connected Things Will Be in Use in 2016, Up 30 Percent From 2015* [Online]. Available: http://www.gartner.com/newsroom/id/3165317.

[2] C. Perera, A. Zaslavsky, P. Christen and D. Georgakopoulos, "Context Aware Computing for The Internet of Things: A Survey," *Communications Surveys & Tutorials, IEEE,* vol. 16, pp. 414-454, 2014.

[3] United Nations Department of Economic and Social Affairs (UNDESA), "Trends in sustainable development - towards sustainable consumption and production," United Nations publications, New York, 2010.

[4] Project MainStream, "Intelligent assets: Unlocking the circular economy potential," Ellen MacArthur Foundation, 2016.

[5] S. Hajkowicz, H. Cook and A. Littleboy, "Our future world: Global megatrends report," CSIRO, 2012.

[6] J. Gustavsson, C. Cederberg, U. Sonesson, R. Van Otterdijk and A. Meybeck, "Global food losses and food waste," *Food and Agriculture Organization of the United Nations,* 2011.

[7] T. Fox and C. Fimeche, "Global food: waste not, want not," *Institute of Mechanical Engineers, London, Jan,* 2013.

[8] FAO. Food wastage footprint: Impacts on natural resources. FAO. Rome. 2013[Online]. Available: http://www.fao.org/docrep/018/i3347e/i3347e.pdf.

[9] M. Bagherzadeh, M. Inamura and H. Jeong, "Food waste along the food chain," *OECD Food, Agriculture and Fisheries Papers,* vol. 71, 2014.

[10] J. Ye, S. Dobson and S. McKeever, "Situation identification techniques in pervasive computing: A review," *Pervasive and Mobile Computing,* vol. 8, pp. 36 66, 2012.

[11] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith and P. Steggles, "Towards a better understanding of context and context-awareness," in *Handheld and Ubiquitous Computing,* 1999, pp. 304-307.

[12] K. Henricksen, *A Framework for Context-Aware Pervasive Computing Applications.* Queensland: University of Queensland, 2003.

[13] A. Bikakis, T. Patkos, G. Antoniou and D. Plexousakis, "A survey of semantics-based approaches for context reasoning in ambient intelligence," in *Constructing Ambient Intelligence*, M. Mühlhäuser, A. Ferscha and E. Aitenbichler, Eds. Springer, 2008, pp. 14-23.

[14] P. F. Petteri Nurmi, "Reasoning in context-aware systems," Department of Computer Science, University of Helsinki, 2004.

[15] S. Sigg, D. Gordon, G. von Zengen, M. Beigl, S. Haseloff and K. David, "Investigation of Context Prediction Accuracy for Different Context Abstraction Levels," *Mobile Computing, IEEE Transactions On,* vol. 11, pp. 1047-1059, 2012.

[16] S. Sigg, *Development of a Novel Context Prediction Algorithm and Analysis of Context Prediction Schemes.* kassel university press GmbH, 2008.

[17] C. Anagnostopoulos, P. Mpougiouris and S. Hadjiefthymiades, "Prediction intelligence in context-aware applications," in *Proceedings of the 6th International Conference on Mobile Data Management,* 2005, pp. 137-141.

[18] J. Ye, L. Coyle, S. Dobson and P. Nixon, "Using situation lattices to model and reason about context," *Proceedings of MRC 2007 (Coexist with CONTEXT'07),* pp. 1-12, 2007.

[19] C. B. Anagnostopoulos, Y. Ntarladimas and S. Hadjiefthymiades, "Situational computing: An innovative architecture with imprecise reasoning," *J. Syst. Software,* vol. 80, pp. 1993-2014, 12, 2007.

[20] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan and D. Riboni, "A survey of context modelling and reasoning techniques," *Pervasive and Mobile Computing,* vol. 6, pp. 161-180, 4, 2010.

[21] S. W. Loke, "Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective," *The Knowledge Engineering Review,* vol. 19, pp. 213-233, 2004.

[22] A. Padovitz, *Context Management and Reasoning about Situations in Pervasive Computing.* Monash University Melbourne, 2006.

[23] S. W. Loke, "Incremental awareness and compositionality: A design philosophy for context-aware pervasive systems," *Pervasive and Mobile Computing,* vol. 6, pp. 239-253, 4, 2010.

[24] L. A. Zadeh, "Fuzzy logic and approximate reasoning," *Synthese,* vol. 30, pp. 407-428, 1975.

[25] P. D. Haghighi, S. Krishnaswamy, A. Zaslavsky and M. M. Gaber, "Reasoning about context in uncertain pervasive computing environments," in *Smart Sensing and Context*, D. Roggen, C. Lombriser, G. Tröster and G. Kortuem, Eds. Springer, 2008, pp. 112-125.

[26] H. Wu, Carnegie Mellon University Pittsburgh, 2003.

[27] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *International Journal of Human-Computer Studies,* vol. 43, pp. 907-928, 1995.

[28] L. Chen, C. Nugent, M. Mulvenna, D. Finlay and X. Hong, "Semantic smart homes: Towards knowledge rich assisted living environments," in *Intelligent Patient Management*, S. McClean, P. Millard and E. El-Darzi, Eds. Springer, 2009, pp. 279-296.

[29] E. Simperl, "Reusing ontologies on the Semantic Web: A feasibility study," *Data Knowl. Eng.,* vol. 68, pp. 905-925, 2009.

[30] J. Ye, L. Coyle, S. Dobson and P. Nixon, "Ontology-based models in pervasive computing systems," *The Knowledge Engineering Review,* vol. 22, pp. 315-347, 2007.

[31] G. Shafer, *A Mathematical Theory of Evidence.* Princeton: Princeton University Press, 1976.

[32] S. McKeever, J. Ye, L. Coyle and S. Dobson, "Using dempster-shafer theory of evidence for situation inference," in *Smart Sensing and Context*, P. Barnaghi, K. Moessner, M. Presser and S. Meissner, Eds. Springer, 2009, pp. 149-162.

[33] J. Y. Halpern, *Reasoning about Uncertainty.* MIT press Cambridge, 2003.

[34] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Machine Learning: ECML-98*, C. Nédellec and C. Rouveirol, Eds. Springer, 1998, pp. 4-15.

[35] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc IEEE,* vol. 77, pp. 257-286, 1989.

[36] T. Van Kasteren, A. Noulas, G. Englebienne and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of the 10th International Conference on Ubiquitous Computing,* 2008, pp. 1-9.

[37] N. Baumgartner and W. Retschitzegger, "A survey of upper ontologies for situation awareness," in *Proc. of the 4th IASTED International Conference on Knowledge Sharing and Collaborative Engineering, St. Thomas, US VI,* 2006, pp. 1-9.

[38] M. Baldauf, S. Dustdar and F. Rosenberg, "A survey on context-aware systems," *International Journal of Ad Hoc and Ubiquitous Computing,* vol. 2, pp. 263-277, 2007.

[39] T. Strang and C. Linnhoff-Popien, "A context modeling survey," in *Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004,* Nottingham/England, 2004, .

[40] M. R. Endsley, "Toward a theory of situation awareness in dynamic systems," *Human Factors: The Journal of the Human Factors and Ergonomics Society,* vol. 37, pp. 32-64, 1995.

[41] M. R. Endsley, "Theoretical underpinnings of situation awareness: A critical review," *Situation Awareness Analysis and Measurement,* pp. 3-32, 2000.

[42] H. Chen, T. Finin and A. Joshi, "An intelligent broker for context-aware systems," in *Adjunct Proceedings of Ubicomp,* 2003, pp. 183-184.

[43] H. Chen, T. Finin and A. Joshi, "The SOUPA ontology for pervasive computing," in *Ontologies for Agents: Theory and Experiences*Anonymous Springer, 2005, pp. 233-258.

[44] A. Boytsov, "Extending Context Spaces Theory by Proactive Adaptation," *Smart Spaces and Next Generation Wired/Wireless Networking,* 2010.

[45] A. Boytsov, *Context Reasoning, Context Prediction and Proactive Adaptation in Pervasive Computing Systems.* Luleå tekniska universitet, 2011.

[46] A. Padovitz, S. W. Loke, A. Zaslavsky and B. Burg, "Towards a general approach for reasoning about context, situations and uncertainty in ubiquitous sensing: Putting geometrical intuitions to work," in *2nd International Symposium on Ubiquitous Computing Systems (UCS'04), Tokyo, Japan,* 2004, .

[47] A. Boytsov, "Extending Context Spaces Theory by Predicting Run-Time Context," *Smart Spaces and Next Generation Wired/Wireless Networking,* 2009.

[48] C. J. Matheus, M. M. Kokar and K. Baclawski, "A core ontology for situation awareness," in *Proceedings of the Sixth International Conference on Information Fusion,* 2003, pp. 545-552.

[49] C. J. Matheus, M. M. Kokar, K. Baclawski, J. A. Letkowski, C. Call, M. L. Hinman, J. J. Salerno and D. M. Boulware, "SAWA: An assistant for higher-level fusion and situation awareness," in *Defense and Security,* 2005, pp. 75-85.

[50] S. S. Yau and J. Liu, "Hierarchical situation modeling and reasoning for pervasive computing," in *Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance. SEUS 2006/WCCIA 2006. the Fourth IEEE Workshop On,* 2006, pp. 6 pp.

[51] G. Birkhoff, *Lattice Theory.* American Mathematical Soc., 1940.

[52] J. Yang, J. Wang and Y. Chen, "Using acceleration measurements for activity recognition: An effective learning algorithm for constructing neural classifiers," *Pattern Recog. Lett.,* vol. 29, pp. 2213-2220, 2008.

[53] K. Devlin, "Situation theory and situation semantics," *Handbook of the History of Logic,* vol. 7, pp. 601-664, 2006.

[54] M. M. Kokar, C. J. Matheus and K. Baclawski, "Ontology-based situation awareness," *Information Fusion,* vol. 10, pp. 83-98, 2009.

[55] G. Tamea, M. Cusmai, A. Palo, F. D. Priscoli and A. Cimmino, "Situation awareness in airport environment based on semantic web technologies," in *Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), 2014 IEEE International Inter-Disciplinary Conference On,* 2014, pp. 174-180.

[56] N. Baumgartner, W. Gottesheim, S. Mitsch, W. Retschitzegger and W. Schwinger, "BeAware!—situation awareness, the ontology-driven way," *Data Knowl. Eng.,* vol. 69, pp. 1181-1193, 2010.

[57] N. Baumgartner and W. Retschitzegger, "Towards a situation awareness framework based on primitive relations," in *Information, Decision and Control, 2007. IDC'07,* 2007, pp. 291-295.

[58] D. Furno, V. Loia, M. Veniero, M. Anisetti, V. Bellandi, P. Ceravolo and E. Damiani, "Towards an agent-based architecture for managing uncertainty in situation awareness," in *Intelligent Agent (IA), 2011 IEEE Symposium On,* 2011, pp. 1-6.

[59] D. Furno, V. Loia and M. Veniero, "A fuzzy cognitive situation awareness for airport security," *Control and Cybernetics,* vol. 39, pp. 959-982, 2010.

[60] M. Stocker, Ed., *Situation Awareness in Environmental Monitoring.* Kuopio: University of Eastern Finland, 2015.

[61] M. Stocker, M. Ronkko and M. Kolehmainen, "Situational knowledge representation for traffic observed by a pavement vibration sensor network," *Intelligent Transportation Systems, IEEE Transactions On,* vol. 15, pp. 1441-1450, 2014.

[62] A. Padovitz, S. W. Loke and A. Zaslavsky, "Towards a theory of context spaces," in *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference On,* 2004, pp. 38-42.

[63] O. Lassila and R. R. Swick, "Resource description framework (RDF) model and syntax specification," 1999.

[64] D. Brickley and R. V. Guha, "RDF Schema 1.1," *W3C Recommendation,* vol. 25, pp. 2004-2014, 2014.

[65] D. L. McGuinness and F. Van Harmelen, "OWL web ontology language overview," *W3C Recommendation,* vol. 10, pp. 2004, 2004.

[66] L. Masinter, T. Berners-Lee and R. T. Fielding, "Uniform resource identifier (URI): Generic syntax," 2005.

[67] P. Hitzler, M. Krotzsch and S. Rudolph, *Foundations of Semantic Web Technologies.* CRC Press, 2009.

[68] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson and A. Herzog, "The SSN ontology of the W3C semantic sensor network incubator group," *Web Semantics: Science, Services and Agents on the World Wide Web,* vol. 17, pp. 25-32, 2012.

[69] A. Boytsov, A. Zaslavsky, E. Eryilmaz and S. Albayrak, "Situation awareness meets ontologies: A context spaces case study," in *Modeling and using Context*Anonymous Springer, 2015, pp. 3-17.

[70] M. A. Musen, "The Protégé project: A look back and a look forward," *AI Matters,* vol. 1, pp. 4-12, 2015.

[71] M. Horridge and S. Bechhofer, "The owl api: A java api for owl ontologies," *Semantic Web,* vol. 2, pp. 11-21, 2011.

[72] K. Dentler, R. Cornet, A. Ten Teije and N. De Keizer, "Comparison of reasoners for large ontologies in the OWL 2 EL profile," *Semantic Web,* vol. 2, pp. 71-87, 2011.

[73] E. Prud'Hommeaux and A. Seaborne, "SPARQL query language for RDF," *W3C Recommendation,* vol. 15, 2008.

[74] E. Sirin and B. Parsia, "SPARQL-DL: SPARQL query for OWL-DL." OWLED, Tech. Rep. 258, 2007.

[75] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean, "SWRL: A semantic web rule language combining OWL and RuleML (May 2004)," *W3C Member Submission,* 2004.

[76] A. Padovitz, S. W. Loke and A. Zaslavsky, "The ECORA framework: A hybrid architecture for context-oriented pervasive computing," *Pervasive and Mobile Computing,* vol. 4, pp. 182-215, 2008.

[77] A. Boytsov, "ECSTRA – Distributed Context Reasoning Framework for Pervasive Computing Systems," *Smart Spaces and Next Generation Wired/Wireless Networking,* 2011.

[78] The Open Group. Open messaging interface technical standard (O-MI). [Online]. *(Open Group Standard),* 2014. Available: https://www2.opengroup.org/ogsys/catalog/C14B.

[79] The Open Group. Open data format standard (O-DF). [Online]. *(Open Group Standard),* 2014. Available: https://www2.opengroup.org/ogsys/catalog/C14A;.

[80] K. Framling, S. Kubler and A. Buda, "Universal messaging standards for the IoT from a lifecycle management perspective," *Internet of Things Journal, IEEE,* vol. 1, pp. 319-327, 2014.

Appendix

A Ontologies

This appendix contains the developed ontologies in RDF/XML format. The Context Space Theory Ontology (CSTO) in appendix A.1 forms the upper ontology for the framework proposed in chapter 3 and 4. The Food Sharing Neighbourhood Ontologies (FSNO) are CSTO-based ontologies implemented for the use case described in chapter 5. The ontology presented in A.2 contains the situation model, the ontology in A.3 the system's setup.

A.1 CSTO

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY csto "http://www.semanticweb.org/csiro/perccom/csto#" >
    <!ENTITY sto "http://vistology.com/ont/2008/STO/STO.owl#" >
    <!ENTITY DUL "http://www.loa-cnr.it/ontologies/DUL.owl#" >
    <!ENTITY ssn "http://purl.oclc.org/NET/ssnx/ssn#" >
]>

<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.w3.org/2002/07/owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:DUL="http://www.loa-cnr.it/ontologies/DUL.owl#"
     xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#"
     xmlns:sto="http://vistology.com/ont/2008/STO/STO.owl#"
     xmlns:csto="http://www.semanticweb.org/csiro/perccom/csto#">
    <Ontology rdf:about="http://www.semanticweb.org/csiro/perccom/csto">
        <imports rdf:resource="http://purl.oclc.org/NET/ssnx/ssn"/>
        <imports rdf:resource="http://vistology.com/ont/2008/STO/STO.owl"/>
        <imports rdf:resource="http://www.irit.fr/recherches/MELODI/ontologies/SAN"/>
    </Ontology>

    <ObjectProperty rdf:about="&csto;hasConfidenceThreshold">
        <rdfs:domain rdf:resource="&sto;Situation"/>
        <rdfs:range rdf:resource="&csto;ConfidenceThreshold"/>
        <rdfs:subPropertyOf rdf:resource="&owl;topObjectProperty"/>
    </ObjectProperty>
    <ObjectProperty rdf:about="&csto;hasContribution">
        <rdfs:domain rdf:resource="&sto;Value"/>
        <rdfs:range rdf:resource="&csto;Contribution"/>
    </ObjectProperty>
    <ObjectProperty rdf:about="&csto;hasRelevance">
        <rdfs:domain rdf:resource="&sto;ElementaryInfon"/>
        <rdfs:range rdf:resource="&csto;Relevance"/>
    </ObjectProperty>
    <ObjectProperty rdf:about="&csto;observesPropertyOf">
        <rdf:type rdf:resource="&owl;FunctionalProperty"/>
        <rdfs:range rdf:resource="&ssn;FeatureOfInterest"/>
        <rdfs:domain rdf:resource="&ssn;Sensor"/>
```

```
</ObjectProperty>

<DatatypeProperty rdf:about="&csto;confidenceThresholdValue">
    <rdfs:domain rdf:resource="&csto;ConfidenceThreshold"/>
    <rdfs:range>
        <rdfs:Datatype>
            <onDatatype rdf:resource="&xsd;double"/>
            <withRestrictions rdf:parseType="Collection">
                <rdf:Description>
                    <xsd:maxInclusive rdf:datatype="&xsd;double">1.0</xsd:maxInclusive>
                </rdf:Description>
                <rdf:Description>
                    <xsd:minInclusive rdf:datatype="&xsd;double">0.0</xsd:minInclusive>
                </rdf:Description>
            </withRestrictions>
        </rdfs:Datatype>
    </rdfs:range>
</DatatypeProperty>
<DatatypeProperty rdf:about="&csto;contributionValue">
    <rdfs:domain rdf:resource="&csto;Contribution"/>
    <rdfs:range>
        <rdfs:Datatype>
            <onDatatype rdf:resource="&xsd;double"/>
            <withRestrictions rdf:parseType="Collection">
                <rdf:Description>
                    <xsd:maxInclusive rdf:datatype="&xsd;double">1.0</xsd:maxInclusive>
                </rdf:Description>
                <rdf:Description>
                    <xsd:minInclusive rdf:datatype="&xsd;double">0.0</xsd:minInclusive>
                </rdf:Description>
            </withRestrictions>
        </rdfs:Datatype>
    </rdfs:range>
</DatatypeProperty>
<DatatypeProperty rdf:about="&csto;relevanceValue">
    <rdfs:domain rdf:resource="&csto;Relevance"/>
    <rdfs:range>
        <rdfs:Datatype>
            <onDatatype rdf:resource="&xsd;double"/>
            <withRestrictions rdf:parseType="Collection">
                <rdf:Description>
                    <xsd:maxInclusive rdf:datatype="&xsd;double">1.0</xsd:maxInclusive>
                </rdf:Description>
                <rdf:Description>
                    <xsd:minInclusive rdf:datatype="&xsd;double">0.0</xsd:minInclusive>
                </rdf:Description>
            </withRestrictions>
        </rdfs:Datatype>
    </rdfs:range>
</DatatypeProperty>

<rdf:Description rdf:about="&sto;Attribute">
    <rdfs:subClassOf rdf:resource="&ssn;Property"/>
</rdf:Description>
<rdf:Description rdf:about="&sto;Dimensionality">
    <rdfs:subClassOf rdf:resource="&DUL;UnitOfMeasure"/>
</rdf:Description>
<rdf:Description rdf:about="&sto;ElementaryInfon">
    <rdfs:subClassOf rdf:resource="&DUL;InformationObject"/>
</rdf:Description>
<rdf:Description rdf:about="&sto;Individual">
    <rdfs:subClassOf rdf:resource="&ssn;FeatureOfInterest"/>
</rdf:Description>
<rdf:Description rdf:about="&sto;Relation">
```

```
        <rdfs:subClassOf rdf:resource="&DUL;FormalEntity"/>
    </rdf:Description>
    <rdf:Description rdf:about="&sto;Rule">
        <rdfs:subClassOf rdf:resource="&DUL;FormalEntity"/>
    </rdf:Description>
    <Class rdf:about="&csto;ConfidenceThreshold">
        <rdfs:subClassOf rdf:resource="&sto;Object"/>
    </Class>
    <Class rdf:about="&csto;Contribution">
        <rdfs:subClassOf rdf:resource="&sto;Object"/>
    </Class>
    <Class rdf:about="&csto;Relevance">
        <rdfs:subClassOf rdf:resource="&sto;Object"/>
    </Class>
</rdf:RDF>
```

A.2 FSNO-Situations

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY STO "http://vistology.com/ont/2008/STO/STO.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY CSTO "http://www.semanticweb.org/csiro/perccom/csto#" >
    <!ENTITY foodcomm "http://www.semanticweb.org/csiro/perccom/foodcommodities#" >
    <!ENTITY fsn-sit "http://www.semanticweb.org/csiro/perccom/csto/fsn/situation-types#" >
]>

<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.w3.org/2002/07/owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:fsn-sit="http://www.semanticweb.org/csiro/perccom/csto/fsn/situation-types#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:CSTO="http://www.semanticweb.org/csiro/perccom/csto#"
     xmlns:foodcomm="http://www.semanticweb.org/csiro/perccom/foodcommodities#"
     xmlns:STO="http://vistology.com/ont/2008/STO/STO.owl#">
    <Ontology rdf:about="http://www.semanticweb.org/csiro/perccom/csto/fsn/situation-types#">
        <imports rdf:resource="http://www.semanticweb.org/csiro/perccom/csto"/>
        <imports rdf:resource="http://www.semanticweb.org/csiro/perccom/foodcommodities"/>
    </Ontology>

    <Class rdf:about="&fsn-sit;Amount">
        <rdfs:subClassOf rdf:resource="&STO;Attribute"/>
    </Class>
    <Class rdf:about="&fsn-sit;AmountPerPerson">
        <rdfs:subClassOf rdf:resource="&STO;Attribute"/>
    </Class>
    <Class rdf:about="&fsn-sit;AmountPerPersonValue">
        <rdfs:subClassOf rdf:resource="&STO;Value"/>
    </Class>
    <Class rdf:about="&fsn-sit;AmountValue">
        <rdfs:subClassOf rdf:resource="&STO;Value"/>
    </Class>
    <Class rdf:about="&fsn-sit;Apple">
        <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
        <rdfs:subClassOf rdf:resource="&foodcomm;Fruits"/>
    </Class>
    <Class rdf:about="&fsn-sit;Banana">
```

```
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;Fruits"/>
</Class>
<Class rdf:about="&fsn-sit;Beef">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;Meat"/>
</Class>
<Class rdf:about="&fsn-sit;Capsicum">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;Vegetables"/>
</Class>
<Class rdf:about="&fsn-sit;Cheese">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
</Class>
<Class rdf:about="&fsn-sit;Chicken">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;Meat"/>
</Class>
<Class rdf:about="&fsn-sit;Egg">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;MilkAndEggs"/>
</Class>
<Class rdf:about="&fsn-sit;Fooditem">
    <rdfs:subClassOf rdf:resource="&STO;Individual"/>
</Class>

<Class rdf:about="&fsn-sit;FooditemAboutToExpireInfon">
    <equivalentClass>
        <Class>
            <intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="&STO;ElementaryInfon"/>
                <Restriction>
                    <onProperty rdf:resource="&STO;anchor1"/>
                    <hasValue rdf:resource="&fsn-sit;Fooditem"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&STO;anchor2"/>
                    <hasValue rdf:resource="&fsn-sit;ShelfLifeAboutToExpire"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&STO;polarity"/>
                    <hasValue rdf:resource="&STO;_1"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&STO;relation"/>
                    <hasValue rdf:resource="&fsn-sit;hasShelfLife"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&CSTO;hasRelevance"/>
                    <hasValue rdf:resource="&fsn-sit;FoodItemAboutToExpireInfonRelevance"/>
                </Restriction>
            </intersectionOf>
        </Class>
    </equivalentClass>
</Class>
<Class rdf:about="&fsn-sit;FooditemHighStockInfon">
    <equivalentClass>
        <Class>
            <intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="&STO;ElementaryInfon"/>
                <Restriction>
                    <onProperty rdf:resource="&STO;anchor1"/>
                    <hasValue rdf:resource="&fsn-sit;Fooditem"/>
                </Restriction>
```

```
                <Restriction>
                    <onProperty rdf:resource="&STO;anchor2"/>
                    <hasValue rdf:resource="&fsn-sit;HighAmountPerPerson"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&STO;polarity"/>
                    <hasValue rdf:resource="&STO;_1"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&STO;relation"/>
                    <hasValue rdf:resource="&fsn-sit;hasRelativeStock"/>
                </Restriction>
            </intersectionOf>
        </Class>
    </equivalentClass>
</Class>
<Class rdf:about="&fsn-sit;FooditemUsageRecommended">
    <equivalentClass>
        <Class>
            <intersectionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="&STO;Situation"/>
                <Restriction>
                    <onProperty rdf:resource="&STO;relevantIndividual"/>
                    <someValuesFrom rdf:resource="&fsn-sit;Fooditem"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&STO;supportedInfon"/>
                    <hasValue rdf:resource="&fsn-sit;FooditemAboutToExpireInfon"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&STO;supportedInfon"/>
                    <hasValue rdf:resource="&fsn-sit;FooditemHighStockInfon"/>
                </Restriction>
                <Restriction>
                    <onProperty rdf:resource="&CSTO;hasConfidenceThreshold"/>
                    <hasValue rdf:resource="&fsn-sit;FooditemUsageRecommendedCT"/>
                </Restriction>
            </intersectionOf>
        </Class>
    </equivalentClass>
</Class>

<Class rdf:about="&fsn-sit;Fridge">
    <rdfs:subClassOf rdf:resource="&STO;Individual"/>
</Class>
<Class rdf:about="&fsn-sit;HighAmountPerPerson">
    <rdfs:subClassOf rdf:resource="&fsn-sit;AmountPerPerson"/>
</Class>
<Class rdf:about="&fsn-sit;Household">
    <rdfs:subClassOf rdf:resource="&STO;Individual"/>
</Class>
<Class rdf:about="&fsn-sit;Milk">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;MilkAndEggs"/>
</Class>
<Class rdf:about="&fsn-sit;Person">
    <rdfs:subClassOf rdf:resource="&STO;Individual"/>
</Class>
<Class rdf:about="&fsn-sit;Potato">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;Vegetables"/>
</Class>
<Class rdf:about="&fsn-sit;Rice">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
```

```
    <rdfs:subClassOf rdf:resource="&foodcomm;Cereals"/>
</Class>
<Class rdf:about="&fsn-sit;Salmon">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;FishAndSeafood"/>
</Class>
<Class rdf:about="&fsn-sit;ShelfLife">
    <rdfs:subClassOf rdf:resource="&STO;Attribute"/>
</Class>
<Class rdf:about="&fsn-sit;ShelfLifeAboutToExpire">
    <rdfs:subClassOf rdf:resource="&fsn-sit;ShelfLife"/>
</Class>
<Class rdf:about="&fsn-sit;ShelfLifeNotExpired">
    <rdfs:subClassOf rdf:resource="&fsn-sit;ShelfLife"/>
</Class>
<Class rdf:about="&fsn-sit;ShelfLifeValue">
    <rdfs:subClassOf rdf:resource="&STO;Value"/>
</Class>
<Class rdf:about="&fsn-sit;Tomato">
    <rdfs:subClassOf rdf:resource="&fsn-sit;Fooditem"/>
    <rdfs:subClassOf rdf:resource="&foodcomm;Vegetables"/>
</Class>

<NamedIndividual rdf:about="&fsn-sit;Amount">
    <rdf:type rdf:resource="&STO;ATTR"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;AmountGreaterZero">
    <rdf:type rdf:resource="&fsn-sit;Amount"/>
    <STO:hasAttributeValue rdf:resource="&fsn-sit;AmountGreaterZeroValue"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;AmountGreaterZeroContribution">
    <rdf:type rdf:resource="&CSTO;Contribution"/>
    <CSTO:contributionValue rdf:datatype="&xsd;double">1.0</CSTO:contributionValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;AmountGreaterZeroValue">
    <rdf:type rdf:resource="&fsn-sit;AmountValue"/>
    <STO:attributeValue>(0;Infinity)</STO:attributeValue>
    <CSTO:hasContribution rdf:resource="&fsn-sit;AmountGreaterZeroContribution"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;AmountPerPerson">
    <rdf:type rdf:resource="&STO;ATTR"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;AmountPerPersonValue">
    <rdf:type rdf:resource="&STO;VAL"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;AmountValue">
    <rdf:type rdf:resource="&STO;VAL"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Banana">
    <rdf:type rdf:resource="&fsn-sit;Banana"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Beef">
    <rdf:type rdf:resource="&fsn-sit;Beef"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Capsicum">
    <rdf:type rdf:resource="&fsn-sit;Capsicum"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Chicken">
    <rdf:type rdf:resource="&fsn-sit;Chicken"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Corn">
    <rdf:type rdf:resource="&fsn-sit;Corn"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Egg">
```

```
    <rdf:type rdf:resource="&fsn-sit;Egg"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FoodItemAboutToExpireInfonRelevance">
    <rdf:type rdf:resource="&CSTO;Relevance"/>
    <CSTO:relevanceValue rdf:datatype="&xsd;double">0.6</CSTO:relevanceValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Fooditem">
    <rdf:type rdf:resource="&STO;IND"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemAboutToExpireInfon">
    <rdf:type rdf:resource="&STO;INF"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemAvailable">
    <rdf:type rdf:resource="&STO;SIT"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemAvailableConfidenceThreshold">
    <rdf:type rdf:resource="&CSTO;ConfidenceThreshold"/>
    <CSTO:confidenceThresholdValue rdf:datatype="&xsd;double">0.8</CSTO:confidenceThresholdValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemAvailableInfon">
    <rdf:type rdf:resource="&STO;INF"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemHighStockInfon">
    <rdf:type rdf:resource="&STO;INF"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemUsageRecommended">
    <rdf:type rdf:resource="&STO;SIT"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemUsageRecommendedCT">
    <rdf:type rdf:resource="&CSTO;ConfidenceThreshold"/>
    <CSTO:confidenceThresholdValue rdf:datatype="&xsd;double">0.8</CSTO:confidenceThresholdValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;FooditemUsageWithCautionRecommended">
    <rdf:type rdf:resource="&STO;SIT"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;HighAmountPerPerson">
    <rdf:type rdf:resource="&fsn-sit;HighAmountPerPerson"/>
    <STO:hasAttributeValue rdf:resource="&fsn-sit;HighAmountPerPersonValue"/>
    <STO:hasAttributeValue rdf:resource="&fsn-sit;MediumAmountPerPersonValue"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;HighAmountPerPersonContribution">
    <rdf:type rdf:resource="&CSTO;Contribution"/>
    <CSTO:contributionValue rdf:datatype="&xsd;double">1.0</CSTO:contributionValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;HighAmountPerPersonValue">
    <rdf:type rdf:resource="&fsn-sit;AmountPerPersonValue"/>
    <STO:attributeValue>high</STO:attributeValue>
    <CSTO:hasContribution rdf:resource="&fsn-sit;HighAmountPerPersonContribution"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Household">
    <rdf:type rdf:resource="&STO;IND"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;MediumAmountPerPersonContribution">
    <rdf:type rdf:resource="&CSTO;Contribution"/>
    <CSTO:contributionValue rdf:datatype="&xsd;double">0.8</CSTO:contributionValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;MediumAmountPerPersonValue">
    <rdf:type rdf:resource="&fsn-sit;AmountPerPersonValue"/>
    <STO:attributeValue>medium</STO:attributeValue>
    <CSTO:hasContribution rdf:resource="&fsn-sit;MediumAmountPerPersonContribution"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-sit;Milk">
    <rdf:type rdf:resource="&fsn-sit;Milk"/>
</NamedIndividual>
```

```
    <NamedIndividual rdf:about="&fsn-sit;PinkLady">
        <rdf:type rdf:resource="&fsn-sit;Apple"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;Potato">
        <rdf:type rdf:resource="&fsn-sit;Potato"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;Salmon">
        <rdf:type rdf:resource="&fsn-sit;Salmon"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLife">
        <rdf:type rdf:resource="&STO;ATTR"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeAboutToExpire">
        <rdf:type rdf:resource="&fsn-sit;ShelfLifeAboutToExpire"/>
        <STO:hasAttributeValue rdf:resource="&fsn-sit;ShelfLifeAboutToExpireNowValue"/>
        <STO:hasAttributeValue rdf:resource="&fsn-sit;ShelfLifeAboutToExpireSoonValue"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeAboutToExpireNowContribution">
        <rdf:type rdf:resource="&CSTO;Contribution"/>
        <CSTO:contributionValue rdf:datatype="&xsd;double">1.0</CSTO:contributionValue>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeAboutToExpireNowValue">
        <rdf:type rdf:resource="&fsn-sit;ShelfLifeValue"/>
        <STO:attributeValue>[0.0;1.0]</STO:attributeValue>
        <CSTO:hasContribution rdf:resource="&fsn-sit;ShelfLifeAboutToExpireNowContribution"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeAboutToExpireSoonContribution">
        <rdf:type rdf:resource="&CSTO;Contribution"/>
        <CSTO:contributionValue rdf:datatype="&xsd;double">0.8</CSTO:contributionValue>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeAboutToExpireSoonValue">
        <rdf:type rdf:resource="&fsn-sit;ShelfLifeValue"/>
        <STO:attributeValue>(1.0;3.0]</STO:attributeValue>
        <CSTO:hasContribution rdf:resource="&fsn-sit;ShelfLifeAboutToExpireSoonContribution"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeGreaterZero">
        <rdf:type rdf:resource="&fsn-sit;ShelfLifeValue"/>
        <STO:attributeValue>(0;Infinity)</STO:attributeValue>
        <CSTO:hasContribution rdf:resource="&fsn-sit;ShelfLifeGreaterZeroContribution"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeGreaterZeroContribution">
        <rdf:type rdf:resource="&CSTO;Contribution"/>
        <CSTO:contributionValue rdf:datatype="&xsd;double">1.0</CSTO:contributionValue>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeNotExpired">
        <rdf:type rdf:resource="&fsn-sit;ShelfLifeNotExpired"/>
        <STO:hasAttributeValue rdf:resource="&fsn-sit;ShelfLifeGreaterZero"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;ShelfLifeValue">
        <rdf:type rdf:resource="&STO;VAL"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;hasAmount">
        <rdf:type rdf:resource="&STO;Relation"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;hasRelativeStock">
        <rdf:type rdf:resource="&STO;Relation"/>
    </NamedIndividual>
    <NamedIndividual rdf:about="&fsn-sit;hasShelfLife">
        <rdf:type rdf:resource="&STO;Relation"/>
    </NamedIndividual>
</rdf:RDF>
```

A.3 FSNO-Setup

```
<?xml version="1.0"?>

<!DOCTYPE rdf:RDF [
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY ssn "http://purl.oclc.org/NET/ssnx/ssn#" >
    <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
    <!ENTITY STO "http://vistology.com/ont/2008/STO/STO.owl#" >
    <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY CSTO "http://www.semanticweb.org/csiro/perccom/csto#" >
    <!ENTITY fsn-su "http://www.semanticweb.org/csiro/perccom/csto/fsn/setup#" >
    <!ENTITY fsn-sit "http://www.semanticweb.org/csiro/perccom/csto/fsn/situation-types#" >
]>

<rdf:RDF xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.w3.org/2002/07/owl"
     xmlns:fsn-su="http://www.semanticweb.org/csiro/perccom/csto/fsn/setup#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:fsn-sit="http://www.semanticweb.org/csiro/perccom/csto/fsn/situation-types#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:ssn="http://purl.oclc.org/NET/ssnx/ssn#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:CSTO="http://www.semanticweb.org/csiro/perccom/csto#"
     xmlns:STO="http://vistology.com/ont/2008/STO/STO.owl#">
    <Ontology rdf:about="http://www.semanticweb.org/csiro/perccom/csto/fsn/setup">
        <imports rdf:resource="http://www.semanticweb.org/csiro/perccom/csto/fsn/situation-types#"/>
    </Ontology>

    <ObjectProperty rdf:about="&fsn-su;hasLocation">
        <rdfs:domain rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Entity"/>
        <rdfs:range rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#Entity"/>
        <rdfs:subPropertyOf rdf:resource="http://www.loa-cnr.it/ontologies/DUL.owl#hasLocation"/>
    </ObjectProperty>

    <Class rdf:about="&fsn-su;FridgeRFIDSensor">
        <rdfs:subClassOf rdf:resource="&ssn;SensingDevice"/>
        <rdfs:subClassOf>
            <Restriction>
                <onProperty rdf:resource="&CSTO;observesPropertyOf"/>
                <someValuesFrom rdf:resource="&fsn-sit;Fooditem"/>
            </Restriction>
        </rdfs:subClassOf>
    </Class>
    <Class rdf:about="&fsn-su;LocationValue">
        <rdfs:subClassOf rdf:resource="&STO;Value"/>
    </Class>
    <Class rdf:about="&fsn-su;Person">
        <rdfs:subClassOf rdf:resource="&STO;Individual"/>
    </Class>
    <Class rdf:about="&fsn-su;ShelfRFIDSensor">
        <rdfs:subClassOf rdf:resource="&ssn;SensingDevice"/>
    </Class>
```

```
<NamedIndividual rdf:about="&fsn-su;FridgeRFIDSensor1">
    <rdf:type rdf:resource="&fsn-su;FridgeRFIDSensor"/>
    <fsn-su:hasLocation rdf:resource="&fsn-su;Household1"/>
    <ssn:observes rdf:resource="&fsn-sit;Amount"/>
    <ssn:observes rdf:resource="&fsn-sit;AmountPerPerson"/>
    <CSTO:observesPropertyOf rdf:resource="&fsn-sit;Fooditem"/>
    <ssn:observes rdf:resource="&fsn-sit;ShelfLife"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;FridgeRFIDSensor2">
    <rdf:type rdf:resource="&fsn-su;FridgeRFIDSensor"/>
    <fsn-su:hasLocation rdf:resource="&fsn-su;Household2"/>
    <ssn:observes rdf:resource="&fsn-sit;Amount"/>
    <ssn:observes rdf:resource="&fsn-sit;AmountPerPerson"/>
    <CSTO:observesPropertyOf rdf:resource="&fsn-sit;Fooditem"/>
    <ssn:observes rdf:resource="&fsn-sit;ShelfLife"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;FridgeRFIDSensor3">
    <rdf:type rdf:resource="&fsn-su;FridgeRFIDSensor"/>
    <fsn-su:hasLocation rdf:resource="&fsn-su;Household3"/>
    <ssn:observes rdf:resource="&fsn-sit;Amount"/>
    <ssn:observes rdf:resource="&fsn-sit;AmountPerPerson"/>
    <CSTO:observesPropertyOf rdf:resource="&fsn-sit;Fooditem"/>
    <ssn:observes rdf:resource="&fsn-sit;ShelfLife"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;Household1">
    <rdf:type rdf:resource="&STO;Location"/>
    <STO:hasAttributeValue rdf:resource="&fsn-su;Household2"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;Household1Location">
    <rdf:type rdf:resource="&fsn-su;LocationValue"/>
    <STO:attributeValue>-37.907367;145.134175</STO:attributeValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;Household2">
    <rdf:type rdf:resource="&STO;Location"/>
    <STO:hasAttributeValue rdf:resource="&fsn-su;Household2Location"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;Household2Location">
    <rdf:type rdf:resource="&fsn-su;LocationValue"/>
    <STO:attributeValue>-37.908061;145.132620</STO:attributeValue>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;Household3">
    <rdf:type rdf:resource="&STO;Location"/>
    <STO:hasAttributeValue rdf:resource="&fsn-su;Household3Location"/>
</NamedIndividual>
<NamedIndividual rdf:about="&fsn-su;Household3Location">
    <rdf:type rdf:resource="&fsn-su;LocationValue"/>
    <STO:attributeValue>-37.907223;145.133800</STO:attributeValue>
</NamedIndividual>
```

B Guides

This appendix provides two guides to the developed library of this thesis: A guide for further development of the library (B.1) and a guide for the development of client applications based on the library (B.2).

B.1 Library Development Guide

This guide is intended for developers who want to further extend or modify the library developed in this thesis. The Java project was realised with Apache Maven (available at https://maven.apache.org/) and can be easily deployed for development, tested on both Linux and Windows.

Prerequisites:

- JDK v1.8 or higher.
- Apache Maven v3.0.5 or higher.

Required Steps:

1. Download source files (contact niklas.kolbe [at] gmail.com).
2. Recommended: Eclipse project with Maven plugin (m2e).
3. `mvn clean` needs to be run initially to install local libraries.
4. `mvn install` will create the complete *.jar* file, as specified in the *pom.xml* file. All required external files and libraries will be added to the archive.
5. `mvn exec:java` runs the three examples provided in the *prototype* package.

The project comes with additional resource files which are required for execution:

- **src/main/resources/ontologies/upper -** contains the upper ontologies - most importantly the *csto.owl* file which needs to be imported by application ontologies.
- **src/main/resources/ontologies -** contains the example ontologies for the three prototypes, based on situation objects (*csto-infection-objects.owl*), situation types (*csto-infection-types.owl*) and populating objects based on types (*csto-infection-populating.owl*).
- **src/main/resources/sensor -** contains files for the acquisition of sensor values.

With these information and the system description from chapter 4 the library can be extended or modified and recompiled to be used by other applications.

B.2 Library Usage Guide

This guide is intended for developers who would like to develop a client application which imports the library developed in this thesis.

Prerequisites:

- JDK v1.8 or higher.
- Compiled library (contact niklas.kolbe [at] gmail.com).

Steps:

1. Import required *jar* library into the project (usage of Maven recommended).
2. Reuse or define new CSTO ontologies about situations and system setup.
3. Create a *CSTManager* instance.
4. Load and add ontologies.
5. Add context collectors.
6. Implement and add custom objects, if required.
7. Initialise application space.
8. Send reasoning requests and receive reasoning results.

The *prototype* package of the library demonstrates how these steps are implemented. The required libraries, ontologies and custom objects depend on the usage of the system. A more complex implementation than the examples of the library was realised for the *Food Sharing Neighbourhood* application (chapter 5) which can serve as another reference implementation.

C Publication

The research of this thesis was formulated in a paper and presented at the Conference on Internet of Things and Smart Spaces, ruSMART 2016 (http://rusmart.e-werest.org/).

Kolbe, N., Zaslavsky, A., Kubler, S., Robert, J., *Reasoning over Knowledge-based Generation of Situations in Context Spaces to Reduce Food Waste,* In: International Conference on Internet of Things and Smart Spaces (ruSMART), September 2016, St. Petersburg, Russia.