UNIVERSITY OF OSLO
Department of Informatics

Introducing New Context Features in the TRIO system

Jan-Ole Bårdevik

September 29, 2016

Acknowledgement

I would really like to thank my supervisors Stein Kristiansen, Thomas Plagemann and Pål Halvorsen for their support and insightful feedback and assistance during this master thesis, and my fellow students which parts of this thesis has been done in cooperation with.

I would also like to thank my family, friends and coworkers for your patience and moral support throughout this period of my life.

Abstract

People are living longer and longer lives, leading to more people in need of health care aid, either in a nursing home or in the individual's home. With the increasing percentage of the population in need of home care, society will need to find new methods to provide the health care that people need. In order to meet the increasing public demand for health care, more of the health care industry has to be automated.

Traditional surveillance systems often utilize cameras. The operation of a camera surveillance system requires someone watching the camera feed. In order to make such a system cost efficient, it is safe to assume that the person monitoring the camera feed is monitoring more than one camera feed.

This master thesis focuses on an alternative kind of surveillance which does not monitor people by cameras, but by an Ambient Assisted Living System [14] called TRIO [2]. The TRIO system requires a steady stream of input data in the form of positions of the person it is monitoring to work. We are exploring if we can use radars as the data input source for TRIO. More specifically we are exploring short range impulse radars designed and manufactured by Novelda [4]. The use of a radar as a source for positional data removes the need for interaction between the monitored person and TRIO. This makes the system usable for people with limited cognitive abilities, for instance, dementia or Alzheimer's disease.

The TRIO system monitors a person and calculates the risk associated with the persons movements over time.

The risk calculations are done by using contextual data from both the position of the monitored person and time passed. The main goal of TRIO is to alert health personnel if the calculated risk is too high.

We make it clear through experiments that the tested Novelda safety radar can not be used in the development of logical sensors and risk functions at this time. This is due to the inaccuracy of the positional data produced by the radar.

In this thesis, we have identified a weakness in the implementation of the contexts zone and period of day (POD). An apartment is divided into zones, where each room in the apartments is one zone. To add more context, the TRIO system has divided the time span of a day into different time spaces called a period of day. The current implementation of these contexts is lacking the possibility of using nested and overlapping PODs and zones. We have designed, implemented and verified a solution for POD, and discussed possible solution for zone implementation.

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Background and Motivation

The increasing living age of the people in the western hemisphere is leading to an almost exponential growth of people in need of home medical care. With the current living age and the number of childbirths, the need for home medical care will increase, but the number of people who can provide such care is not increasing in the same pace to satisfy the demand. In Norway, the number of people at the age of 80 years or older has more than doubled in the last 30 years. Statistics from Statistisk sentralbyrå in Norway say that the number of persons at the age of 67 or older has increased with 125 800. Most of this increase is in the age group 80 years and older. Thirty years ago, this group comprised about 2,6 % of the population, while at the entrance of the year 2007 the group comprised about 4,7 % of the population. [3]

In Norway, elderly people in need of home care is visited at home in different intervals depending on their needs. One person might need help once a week, and others might need help twice a day. This system gives the patient some aid and provides some surveillance of the general health and well-being of the patients. For instance, if an elderly person was to fall down in the bathroom, and was not able to get up, it might take a long time before he was found. If the person only gets visited by a nurse once a week, and the person was to fall down the day after the visit, it might take almost a week before someone would notice. As safety for situations like this, many elderly have a panic button around their neck, which they can press and health personnel will be notified. There is, however, many things that could go wrong with such a system. For instance, the safety button can be broken if the patient falls on it, or the patient may have forgotten to change the batteries. Other things that may cause problems is if the patient has forgotten the button in another room, or maybe the patient has dementia. A patient with dementia may not know what the button around their neck is. If they are not able to move, the patient could in a worst case scenario be lying on the floor until someone finds them.

We have for a long time been automating our lives, with everything from dishwashers to robotic vacuum cleaners. There has also been a development in the area of automated homes also called smart homes. A smart home is for instance, a home with door locks that can be opened by a code or an id chip. Such a lock can be connected to other systems in the home, such as heating and light control. Companies such as Samsung provides kits for making our home smarter, with remote controlled electrical plugs that can be turned on or off by a cellphone. There are other more advanced smart homes in the experimental stage which is more targeted at the needs of the aging population, systems that for instance, notifies a plumber if there is a water leak, and microwaves that recognizes the food put in and heats it accordingly. The smart homes may also have sensors which automatically follows the whereabouts of residents in the home, and notifies visitors of the whereabouts of the resident. Many people have security alarms in their homes with cameras installed in most of the rooms. These cameras are only supposed to be used in emergencies, but there has been reports of people that have been spied on in their home by their home security provider. One could use such a system to provide camera surveillance of a person in their home, but many will feel that this is an invasion of privacy.

Existing systems in the domain of smart homes and home medical care can not manage the future needs of the home medical care domain. The need for medical home care is increasing more than the people able to provide it. Society will therefore have to find alternative ways to provide people with the care they need, while reducing both the resource and manpower cost. One of the more viable solutions to this problem is to automate what can be automated, such as monitoring of people in need of medical supervision in their homes. We wish to explore and improve such a monitoring system.

In this respect, we have the TRIO project which is a project in cooperation with the UiO, Novelda and Oslo university hospital (OUS). The goal of this project is to explore and improve how a safety radar can be used in home medical care, discover what kind of health parameters can be measured with a safety radar, and how data collected from the safety radar can be handled and analyzed to assess risk. The TRIO system relies on a continuous input of positional data which the system uses to calculate if the monitored person is in a dangerous situation or not. This risk assessment will then be used to evaluate if medical assistance is needed [2]. The TRIO system as of now visualizes an apartment where each room in the apartment represent a zone. The system focuses on tracking a person in their home, visualizing the monitored person's position at all times. The visualization is done in a simulator which shows an overview of the apartment with all its zones. We therefore classify TRIO as an automatic home surveillance and tracking system.

To be able to use TRIO in a real life application, one must find something to use as a positional input source for the system. One way of getting positional data is by using a radar to track the resident in his or her home. However, as of now, the system is incomplete and important functionality is missing.

1.2 Problem Statement

This master thesis aims in general to explore ways to improve the TRIO system both with relation to possible input sources and existing functionality. First, we will experiment with short range impulse radars supplied by Novelda, in order to see if these radars are useable as an positional input source for our ambient assisted living system.

We then shift our focus to the TRIO system, focusing on exploring how the TRIO system is working now and what can be done to improve the system for use in the medical home care sector. More specifically, we are looking to expand and improve the functionality of the contexts in TRIO. The contexts in TRIO are: zone and period of day (POD). Zone contexts are used for defining different thresholds for what "normal" behavior is, in different rooms. In the current implementation, each zone represents a room. For instance, it is normal to not be moving for a longer time in the bedroom than in the kitchen. POD context works in the same way, POD lets us set thresholds depending of what time of day it is, as what is "normal" behavior depends on the time of day. It is not uncommon for a person to not be moving in eight hours at night, but it is uncommon for a person not to be moving in eight hours during the day.

The reason for focusing on the current implementation of these contexts is that they have weaknesses regarding proper risk assessment. TRIO calculates risk based on the movements of the monitored person. TRIO uses POD and zone to improve how correct the risk assessment is, because the risk of the monitored persons movements depends on both the time of day and where the person is located. The current implementation of these contexts is limited by the fact that the system as of now, can only handle one active POD at a time, and only one active zone at a time.

Thus, the research question addressed in this thesis include:

- Can the safety radars supplied by Novelda be used as a positional input device for TRIO? For TRIO to be useful, we need a reliable positional input source.
- Can the concept of POD be improved to handle independent PODs?
- Can the concept of zones be improved to handle independent zones?

If we improve the concepts of POD and zone, TRIO will be able to define more precise contexts, which leads to better risk assessment.

1.3 Scope and Limitations

We have tried to use Novelda's radar as positional input data, but as seen later, our results show that it is too inaccurate. When that failed, we used the mouse pointer on the screen to simulate a persons movements. This means that we are using the mouse cursor as the positional input source for the TRIO application. By moving the mouse around on the simulator in TRIO, we simulate the monitored person movements.

Another limitation of both the radars and the TRIO system is the ability to track more than one person at a time. The radar-script (code from Novelda that calculates positions) can only track one person at a time. If there are more than one person in radar range at the same time, the position data gets distorted. As of now, the TRIO system is made to follow one person at a time. TRIO only receives positional data which the system assumes is from only one person. The system may be expanded to track more than one person in the future, but for the system to work optimally, the system would need positional data with an id, e.g. as done in sports using tags and belts to track athletes movements in the field. [17] If the system is to collect statistical data on more than one person, the system would have to be able to tell them apart, or else the statistical data would be of no value.

The TRIO system is supposed to track and store behavioral data, like movement patterns and sleep patterns. We want to do this because we want the system to react to behavior that is unusual for the monitored person, possibly indicating that something has happened. For instance, not everybody is in the bathroom for the same duration. However, if a person is in the bathroom for a longer duration than observed in less than 10% of the historical data, this might be an indication that something is wrong. To achieve this, the TRIO system will have to be able to store data for historical reference. This functionality is however not implemented in the version of TRIO that we are using. Therefore, the risk functions in TRIO are not using historical data to decide whether the monitored person has been in the bathroom for too long, but it uses hard-coded values. This is something that should be improved in the future, but this is out of scope for this thesis.

1.4 Contributions

Our contributions in this thesis are multiple. First, we test and determinate that Novelda's safety radar used in this project is not viable to use as an positional input source for the TRIO system at this time. We have tested the Novelda safety radar with the triangulation script that was supplied by Novelda. To do this, we built a testing lab where we performed experiments. We did many different experiments to test the quality of the positions produced by the triangulation script and radars. For instance we did experiments with a still object, a still person and a person moving in a specific pattern. All experiments where done several times in order to normalize results and reduce the impact of anomalies. We notified Novelda about what we discovered during these experiments, Novelda tried to improve their triangulation script, but after some quick tests we found out that it still not good enough to be used as a positional input source for TRIO.

Following the radar experiments, we explored the TRIO system in detail, focusing on limitations regarding the main goal of TRIO. TRIO's main goal is to calculate risk from the movements of a person. We did in this phase discover a limitation in TRIO, which if improved would lead to more accurate risk calculation. The limitations we discovered is the inability for TRIO to have more than one active POD or zone at the same time.

We have explained why the implementation of POD has a weakness, and we have made design alternatives to improve the implementation. After choosing a potential improvement, we have implemented it in the TRIO system. We have performed some preliminary tests to evaluate if TRIO is working as expected after our implementation.

The other limitation of TRIO, is the inability to have more than one active zone at a time. This weakness has not been improved, but we have explained why this is a weakness, and we have created some design alternatives that may be used in the future.

Our improvement of POD lets us have more than one active POD at a time, and enables us to add PODs completely independent of existing PODs. This lets us divide the hours of the day into more specific contexts. We can for instance, add a POD that covers the whole day, i.e. in the future TRIO can record behavior patters for whole days. If the suggested zone improvement is implemented, it will let us be able to create a zone within another zone. Which is useful for creating furniture zones within room zones. Furniture zones can be used in risk functions to activate an alarm at different durations of time, depending on whether the monitored person is located on a piece of furniture or on the floor.

1.5 Research method

In this thesis, we are using a prototyping approach, corresponding to the design paradigm described by ACM Task Force in *Computing as a Discipline* [7]. We have done experiments with the radars and triangulation script supplied by Novelda. We did these experiments to determine if their positional output quality was good enough to be used as an positional data input for TRIO.

We have looked deeper into how the TRIO system is going to be used, we have imagined many scenarios in which the TRIO system can be used, and we have focused on scenarios where the TRIO system has limitations regarding exact risk calculation. In this process, we discovered some weaknesses in TRIO which should be improved. We have analyzed, implemented and tested a prototype of an improvement in TRIO.

1.6 Outline

In Chapter 2, we explain key aspects of the system we are researching and evaluating. We start with the TRIO system itself and the related technologies. Then, we go on to explain more in detail about the subject of ambient assisted living and what a system like this should do.

After explaining the domain, we focus in Chapter 3, on the Novelda radars, and research whether this type of radar can be used as positional input data source for the TRIO application. We go into details about the software used together with these radars. Then, we focus on experiments for determining the accuracy of the radars, in this Chapter we present our goals and results for the experiments as well.

In Chapter 4, we explain one of the key-factors in the TRIO application, which is contexts. We start with the definition of a context, what it is and how it is used in different fields and how it is used in the TRIO application, i.e., leading up to how we define risk functions in TRIO. We have also explored and discussed in detail the different contexts in TRIO, how they are implemented and what weaknesses they have, followed by possible solutions and which solution we think is best.

After this, we discuss the POD improvement for the TRIO system in Chapter 5. We start by explaining the specifics of the implementation, what the consequences of the implementation is and how it will affect the system. Then, we do some verification tests to verify that our improvement has not broken TRIO. Followed by a discussion of how our implementation may affect the TRIO system from a scalability and expandability point of view. Then, we do some performance tests on TRIO after our implementation.

Finally, we summarize and give conclusions in Chapter 6.

Chapter 2

Background

In this chapter, we explain what ambient assisted living is. We then explain one of the focus areas in this master thesis, which is to understand and evaluate how the Novelda's safety radar can be used in a home medical care setting. More precisely, how we can use the radar in an ambient assistant living system. What are the existing and future possibilities, where can it be used, and how we can use it. Consequently, we first explain the radar we are using, and the system which we are using the radar with. Finally, we shift our focus to the TRIO system, and how this system is built up, by explaining some of the key parts of this system. We also explore both general uses and improvements of the TRIO system.

2.1 Ambient Assisted Living

An Ambient Intelligence system is a electronic system that provides sensitive and responsive services to people, by being integrated into their daily environment in a unobtrusive way, according to *Emile Aarts et al* presented in [11]. To explain the concept Ambient Assisted Living, this Section gives a brief summary of what *Nehmeretal et al* presented in [14]. The key problem is the increasing number of people in need of home care, due to the increasing number of elderly people living in the western hemisphere. This paper considers the solution to be technological in the form of Ambient Assisted Living explained in the following text.

There is a huge potential for Ambient Assisted Living technology in the area of living assistance for handicapped and elderly people suffering from all kinds of disabilities. This technology is meant to notice if something is wrong with a patient - and in that case, it should react, for example, by notifying medical care givers, or try to communicate with the patient.

The need for this kind of technology is definitely increasing in industrialized societies especially in the western hemisphere, where the population age is continuing to grow. As the age of the population grows, there is a growing share of handicapped and elderly people unable to conduct their normal lives at home unassisted, which will be an enormous cost for society [12].

2.1.1 Living Assistance Classification

The authors think that is worthwhile to distinguish between indoor and outdoor living assistance. This is how they define and differentiate inside and outside systems:

- **Inside systems**

  “ Systems for indoor living assistance work in a well defined locality scope: in apartments, homes, cars, hospitals, and elderly care homes. Indoor living assistance systems can be built upon a well known hardware/software installation in the location, thereby providing a stable environment. ” [14]

- **Outside systems**

  “ Outdoor living assistance systems support persons during activities outside their homes: while shopping, traveling and during other social activities. These systems are faced with highly unstable environmental conditions such as availability of wireless communication, accessibility of network services, and context information acquired via sensors, which adds another level of complexity and uncertainty to those systems. ” [14]

They also distinguish between three types of services:

1. Emergency treatment, the system should react if there is an emergency. For instance, notify health personnel if the person has fallen and is unable to get up.

2. Autonomy enhancement, services that help elderly people live alone for longer with minimal assistance, such as automated doors.

3. Comfort, services that focus on increasing comfort, such as automated lights.

Emergency treatment is considered to be the kernel of any living assistance system. It aims at the early prediction of and recovery from critical conditions that might result in an emergency situation. Examples of emergency situations are sudden falls, heart attacks, strokes, panics, etc. This is the type of service that we are focusing on in this master thesis.

2.1.2 Home Care Systems

“ Living assistance systems focusing on the support of handicapped and elderly people in their own homes are called home care systems. ” [14]

“ According to estimates in 2003, the number of people in Germany who are older than 70 will increase from 10% at the present to 18% by the year 2040. About 1.5 million Germans currently spend their life in elderly care homes. This number will double by 2020. ” [14]

Elderly people have a high risk of suffering from typical high age diseases which leads to the need for extra care which again leads to surveillance. Many of these illnesses may also limit movement which further increases the need for assistance. *" It is also noteworthy that 50% of all injuries occur to elders in their homes. "* [14] Many elderly people have decreased physical ability, and as much as 40% of people over 70 have chronic illnesses that result in activity limitations.

The costs of providing care for the population is increasing with a decreasing age-dependency ratio defined by the number of working individuals divided by the number of handicapped people in a country. This ratio will approach 1 in the next 10 to 20 years.

In light of this information it is clear that society has to react to this dramatic process. This is why it is becoming so important to create innovative solutions for living assistance systems. Ambient home care systems that are based on ambient intelligent technology is a good approach. This kind of system increases the time persons can live self-conducted lives in their own homes and reduce the need for intensive personal care. Thereby, they increase their quality of life and decrease the cost for society. *The ultimate goal of any emergency treatment system must be [14]*

(a) high recall in detecting every real emergency immediately

(b) high precision, to prevent invalid emergency detections and alerts as a consequence of misinterpretations.

To be able to create a system that is trustworthy, requirement (a) must be met. Requirement (b) is really important, as this would be a financial drawback and it would decrease trustworthiness. One of the goals of this kind of systems is to be able to predict emergencies, recognize critical health conditions before it escalates into an emergency, and then notify the patient and/or medical professionals. To be able to do this the system needs to continuously monitor data about a person's physical conditions such as heart rate, temperature, blood pressure and so on. A combination of all of these data will give the system enough information to see if anything is wrong. The system would also need to be able to recognize events. Events could be:

- person X has fallen down, or
- person X has not responded to a call for S seconds.

Then, the system should be able to combine these events like these to identify a situation:

- (person X has fallen down) AND (person X has not responded to a call for S seconds).

When the system can combine events like this, the system would be able to determine when something is wrong and assistance is required.

Another fact to consider is that elderly and handicapped people may lack the skills for handling such a system. It is therefore important to make the system as invisible and unobtrusive for the user as possible. The system would need to be able to use several sensors and get information from many sources, using many of them at once to come to conclusions about a situation. As ”invisible” systems tend to be less accurate and we need to accommodate expected trustworthiness. The system would also have to be very robust and still provide it’s services even if there are unexpected errors in parts of the system, component malfunctions, or power/battery break down. Software would also need to be updated remotely with minimal human interaction and without interrupting the system while doing so.

2.1.3 Ambient Intelligent Living Assistance Systems

Above, we have learned about ambient assisted living (AAL), here are some of the main characteristics of an AAL system:

- invisible, i.e. embedded in things like clothes, watches, etc.,
- mobile,
- context aware, able to communicate and cooperate with other sensors,
- anticipatory, i.e. acting on their own behalf without user involvement,
- communicating naturally, interacting with potential users by voice or gestures rather than mouse and text on a screen, and
- adaptive, i.e. capable of reacting to all kinds of abnormal / exceptional situations in a flexible way without disruption of their service.

All of the topics discussed above are important parts of ambient assisted living, although they are not all relevant for this master thesis. E.g., outdoor sensory is not something we will explore. Neither will we explore communication between different kinds of sensors. This is something that will need to be implemented to make a complete ambient assisted living system.

In this master thesis we are working with TRIO, which is the beginning of an indoor AAL system, so it is important to understand which criteria TRIO will have to fulfill to be a complete AAL system. We now know what an ambient assisted living system is and why we need such systems. The necessity for AAL systems will only increase as the population ages. Above we also have clear definitions of important criteria that an AAL system would have to fulfill. For instance, combining events to understand if a situation occurs which requires the system to take action.

If the Novelda’s safety radar is to be used in an AAL home care system it will be used in combination with other sensors and different software. This is because a radar alone would not be able to give enough data to identify all situations that are required by an ambient assisted living system as discussed above.

One of the goals of the radar is to help with ambient assisted living where we utilize new technology making people able to stay in their homes for longer with non-intrusive surveillance. For example, many elderly people want to be able to stay in their homes, but are in need of surveillance. One option for surveillance is cameras, but many feel that cameras are an invasion of their privacy. Another option is radars like Novelda's safety radar, which also supply surveillance in a less intrusive way. There are several reasons why a radar solution is better than cameras, for example, a camera needs a person monitoring the camera feed to be able to detect if critical situations occur. However, a radar connected to an ambient assisted living system would only alert medical care professionals when needed, thus needing considerably less human resources.

In this master thesis we want to explore how we can use Novelda's safety radar in an AAL system. By utilizing logical sensors for the monitoring of movements of a person in an apartment, we can discover potentially dangerous situations. As mentioned above, the radar also has the ability to penetrate a person, this means that in theory we could also monitor the vitals of a person when they are laying in bed. We would be able to do so without having to attach monitors to the person. This provides the users with more security without having to sacrifice comfort.

2.2 The Novelda Safety Radar

Novelda[1] AS has developed a short distance radar, that has many application areas. One of these application areas is in the home medical care arena. This radar can be used to monitor pulse, respiration and movement of a person. We are exploring if this short range impulse radar can be used to monitor the movements of a person in an apartment.

In this master thesis, we are more specifically working with the Novelda NVA-R6X1 development kit. In Figure 2.1, we can see the complete assembled kit we have been using. The radar is built with an analog transmitter antenna, an analog receiver antenna, a sampler and a digital I/O device to send the collected information to a PC for analyzing the radar. To be able to send these electromagnetic waves, the radar also contains a power supply, which means that the radar needs to be connected to a power source. In this case, the radar gets power from a USB power supply.

In order to understand how Novelda's safety radar works, we first need to understand how a regular impulse radar works.

A regular impulse radar emits an electromagnetic pulse that consists of electromagnetic waves and measures the elapsed time from sending of the signal until it receives the waves in return from the first object that has been hit by them. The waves travel at the speed of light and the elapsed travel time has been

[1] *" Novelda AS is a privately held, R&D driven, sensor company based in Norway and specializing in nanoscale wireless low-power technology for ultra-high resolution impulse radar."* [4]

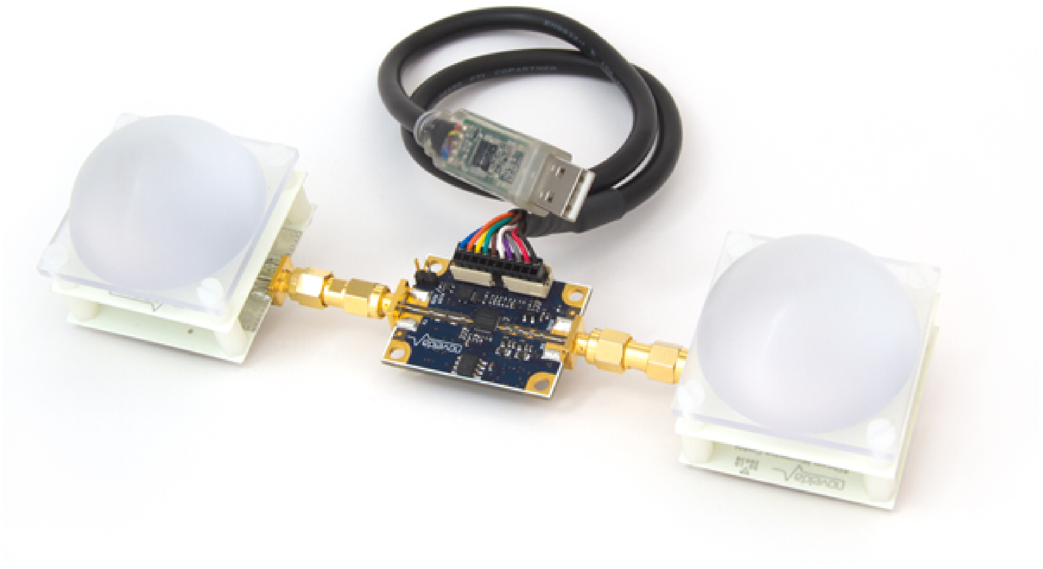


Figure 2.1: Completely assembled radar kit NVA-R6X1 [15]

measured, consequently, we can now calculate the distance between the radar and the object that was hit by the electromagnetic wave.

The safety radar from Novelda is a little different from other impulse radars, it does not have a clock, and it does not just send one pulse at a time, it actually sends 256 pulses directly after each other. While Novelda's safety radar does not have a clock, it does sample for returned waves at certain intervals. After the pulses are sent, the radar checks how much time has elapsed by counting how many sample intervals there has been between sending the waves until they return. The pulses sent out from the radar are in many different wave lengths which means that the radar will not only be able to detect the first object it hits, but also objects behind it because of the many pulses and many samplings. The fact that Novelda's radar requires no clock makes it consume less power and be more efficient than a traditional impulse radar as explained above.

To calculate a persons position, the system use triangulation between the object and the two radars. The radar triangulation is illustrated in Figure 2.2 where the blue circles on the x-axis (where y=0) are the radars. The red square is the object. In the figure, there is a dotted line which connect the two radars with the object. All distances of this triangle is now known, so we are able to calculate where, relative to origo (x=0,y=0) the object is located in the coverage area, which is illustrated in the figure.

The main goal of using the radar is to track people. By using at least two radars, we are able to find the location of a person or an object within the coverage area of the radars. To do this, we use triangulation like this: we measure the time radio waves use to travel from each radar to the first hit object and back to each radar. Then, we calculate the distance from each radar to the object using the travel time of the radar waves. The radar measures the travel time of the waves

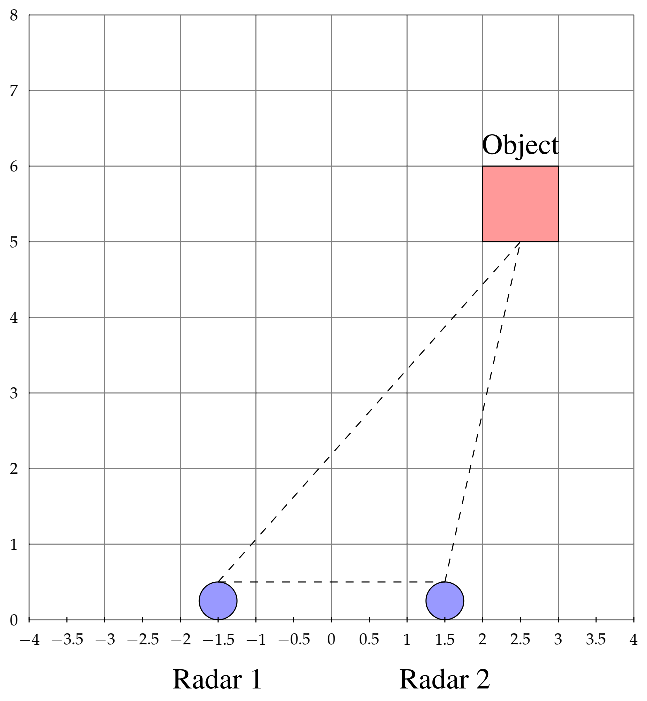


Figure 2.2: Visual representation of triangulation.

by doing consecutive samplings with a constant interval. With this information and knowledge of the distance between the radars we now have a triangle with distances. All that is needed then is to calculate based on the distance triangle to determine where an object relative to the radar.

We will focus on how the radar works in two dimensions. It is however worth mentioning that the radar sends out radio waves in the shape of a bubble so it is possible to utilize the third dimension as well. This however, adds a lot of complexity, so it is not something we will be exploring immediately. Although it is definitely something that can be explored in the future, when both the radars and the radar software have matured a bit.

The radar produces, as mentioned, radio waves. These have different wavelengths. Signals with different wavelengths behave differently regarding surrounding materials. While some radio waves have short wavelengths and will be reflected from materials with low density, radio waves with long wavelengths will not be reflected until the waves hit material with higher density. If we imagine that we have some material between the radar and the monitored person, the radar would still be able to track the person because of the utilization of radio waves with different wavelengths. While some of the radio waves would not hit the person, other radio waves with different wavelengths would. This way the radar can still monitor a person even if there is some obstacles between the radar and the person.

Another interesting fact is that because there are so many different wavelengths, some of these waves will also penetrate the skin of the monitored person and some waves will be reflected by the heart of the monitored person. This means that we could also use this radar to monitor the heartbeat or respiration of a person.

When we are tracking a person through anything other than just air, this will result in more noise in our readings. This is, however, handled by the triangulation script supplied by Novelda.

2.3 TRIO

The TRIO project is a project in cooperation with the University of Oslo(UiO), Novelda and Oslo university hospital (OUS). The goal of this project is to explore and improve how a safety radar can be used in home medical care, discover what kind of health parameters can be measured with a safety radar, and how data collected from the safety radar can be handled and analyzed to assess risk. This risk assessment will then be used to evaluate if medical assistance is needed [2]. The TRIO system as of now visualizes an apartment where each room in the apartment is divided into zones. The system focuses on tracking a person in their home, visualizing the monitored person's position at all times. The visualization is done in a simulator which shows an overview of the apartment with all its zones. We therefore classify TRIO as an automatic home surveillance and tracking system.

2.4 Logical Sensors

In this master thesis, we will also be talking about logical sensors, and the difference between a logical and a physical sensor. To explain the difference, we refer to Jarle Sjøberg's PhD thesis. A physical sensor transforms analog signals into digital values. A logical sensors aggregate data from other sensors, and is therefor dependent on input from other sensors to work. [19] According to this definition, a logical sensor is a piece of software that measures physical quantities and converts it to data.

Pieces of software can be used as measuring tools, and we can call such a piece of software a logical sensor. In fact, the logical sensors that we are using are utilizing data from Novelda's safety radar as our physical sensor.

One challenge regarding use of the radar is that the radar software needs to be easy to use for someone without intimate knowledge of how the radar works. This is because the radar software will also be used by medical personnel with limited knowledge about the radar, and/or programming.

This is where logical sensors will be playing a big part, because if people that have intimate knowledge of the radar create enough logical sensors, people without this knowledge or computer science will be able to utilize the radar software by using logical sensors only.

The illustration of logical sensors usage is best done with an example. Imagine that we have a logical sensor that finds the coordinates of a person in a room. We use this logical sensor repeatedly over time, to record the coordinates of the person to track movements. We can use a persons movements to detect if something is out of the ordinary. For example, if the person we are monitoring usually moves around in their apartment a lot, but then after some time almost stops moving around, this could be an indication that something is wrong. However, if a person falls down, the same logical sensor used over time can detect this if the person moved fast for a short distance and then suddenly stopped moving. This could indicate that the person has fallen. With the use of complex event processing, explained in Section 2.5, the AAL (ambient intelligent living) system can notify health personnel if needed. We have explained more about what an AAL system is in section 2.1

We can have another logical sensor which could identify a heartbeat, and then we could check the time between identified heartbeats to discover any irregularities. To create a logical sensor that can identify heartbeats, we need intimate knowledge of the radar such as placements and limitations, while the use of this logical sensor does not. This means that programmers can develop advanced logical sensors and medical professionals can determine how to use these logical sensors to discover risks and help determine if something is wrong. A logical sensor could collect data that may help doctors map the health condition of a person over time.

2.5 Complex Event Processing

One of the big challenges when working with sensors is that they are continuously generating data. For instance, Novelda has created a MATLAB script called *demorig2*, which is a triangulation script that triangulates the location of a person within the coverage area of the radars. If the covering area range of the script is set to three meters, the script will produce about 20 locations each second. The locations per second will decrease as the range increases. This is because as the range increases, the elapsed time from when the radars have sent the waves until the waves return to the radar increases, which means fewer locations produced.

Not all of these produced locations are interesting to us, and because there is so many of them we have to filter out the locations which are irrelevant and keep the locations that may be part of an event. An event can in our case be the logical sensors that check if the monitored person is stationary. For this logical sensor all the locations that indicate that the person is moving are filtered out because they are not relevant. All locations that indicate that the person is stationary are, however, relevant and will be considered part of the stationary event.

This event can not be divided, the person is either moving or stationary and is therefore considered an atomic event. An atomic event by itself may have little or no value, but many atomic events over time or combined can be very useful. We call the combination of several atomic events for ’complex events’.

For instance, in Chapter 2.4, we explain logical sensors and how logical sensors can be used to e.g. detect a fall or keep track of movement statistics. Movements are positions over time, as a position recorded in TRIO is considered to be an atomic event. Positions are then combined with time that gives TRIO knowledge of where the monitored person is located and if they are moving.

Another example, if TRIO monitors the activity levels of a person, then TRIO should be able to detect if the monitored person's movements in the apartment increase or decrease over time. This process can be automated so that the TRIO alerts health personnel if the activity level changes more than a certain threshold. This threshold should set by health care professionals or learned from the person's activity level over time. To achieve such a function we must create an activity level logical sensor. The activity level logical sensor keeps track of the monitored persons activity level each day. We can then create another logical sensor which keeps track of the changes in activity level, this is done by comparing the activity levels of a person over time.

As we can see, complex event processing is a big part of an AAL system like TRIO.

2.6 Esper

As mentioned, we are working with a system called TRIO. TRIO uses Esper [1] as its complex event processor. Esper is a system that processes data streams continuously. TRIO is, as mentioned, an automated home surveillance system, meaning that the system gets a continuous stream of input positions, which we use Esper to process. Esper is like a reversed database where all of the data is filtered through queries inserted into Esper with an event processing language (EPL), where EPL is an SQL like querying language. Esper is a domain specific language for processing events, and EPL is a declarative language for dealing with high frequency time-based event data. Another commonly used term for this technology is SQL streaming analytics. Esper lets us detect events in data streams using EPL to declare what events we are looking for, i.e., all the positioning data that is inserted to the system goes into Esper, and we then use queries against Esper to determine events. Esper is a complex event processing engine, which we use in the TRIO system.

2.7 Summary

In this chapter we have have explained in detail what an ambient assisted living is, and what the demands for an ambient assisted living system are. We have explained how the radar we are using in this thesis works, and how it differs from a regular impulse radar. The TRIO system we are working on has been explained with the relevant technologies used in TRIO.

We know that the Novelda radar is a possible positional input source for the TRIO system. Next, we research whether this radar is a viable device for localizing a person in a room, as correct positioning is critical for a system such as TRIO.

Chapter 3

Radar Experiments

As the Novelda radar is a possible positional input source for TRIO, we wish to explore if this is an viable option in terms of positional accuracy. We need to test the limits of the technology to figure out if it is usable for our application.

We find it worth mentioning once again that the software we use for triangulation, and visualization of the radar data is limited to only being able to track one person at a time.

3.1 Supplied Software

We also have some supplied software from Novelda to explore the data collected from the radars. The radar raw data is sent from the radars through USB ports on a computer in the vicinity of the radars. It is worth mentioning that Novelda has a new model of radars that are able to connect to computers via wifi, but this is not the model we are working with. This will make the placements of the radars easier, because we do not need to physically connect them to the AAL system with wires. Our radars, however, are placed in a test lab at UIO, and they are connected to a computer in the same room, with the software described below.

3.1.1 RadarScope3

There is some radar software already supplied by Novelda, like the Novelda software bundle, which contains RadarScope3 [15]. RadarScope3 is software that connects with the radar and lets us change different parameters on the radar while looking at the raw data the radar produces through different gauges. This might prove useful in certain situations, but what we are focusing on in this master thesis is the location of a person or an object tracked by the radars and this software does not offer us much to achieve that goal. The radar software RadarScope3 is at the moment only available on Windows.

3.1.2 MATLAB script called demorig2

The aforementioned computer that is connected to the radars also have MATLAB installed with a script called demorig2 [16], this MATLAB script is supplied by Novelda. This script utilizes the direct output data from the radars and uses this data to triangulate the monitored persons position using lateration generated coordinates of the object or person captured by the radars. We can access the coordinates generated by the MATLAB script from other software by connecting to the MATLAB script from a port. The MATLAB script will then continuously send a timestamp and coordinates divided in three axes which will look something like this: (timestamp, x, y, z). The script demorig2 is limited to only being able to track one person at a time.

Before we continue, it is worth mentioning that since we only have two radars, we will only be doing lateration in a two dimensional plane, which means that in our case, the axis z will always be 0, but it is included for future use. As we connect more radars to the script, it would be fairly easy to change it so that it would do lateration in a three dimensional plane.

3.1.3 TRIO v1

The Distributed MultiMedia Systems (DMMS) group at UiO, has made a program which can determine risk by monitoring the movement of a person in an apartment. The program is called TRIO, which, in Norwegian, is an abbreviation for "Trygghetsradar i hjemmebasert omsorg", which translates to Safety Radar for Home Care [2].

The TRIO v1 program initially took its positional input from a computer mouse, by moving the mouse cursor in a simulation window in the program. So, we [1] modified TRIO to also be able to take its positional input from demorig2 [16]. We did this to test the quality of information and the consistency of the triangulation performed by the radars and the MATLAB script. We found that the lateration generated by the MATLAB script was not good enough at this time, more on this later in Chapter 3.3.

3.1.4 TRIO v2

After we started this work, large parts of TRIO v1 has been rewritten. The major changes made in TRIO v2 are:

- Radar input from both mouse and radars.
- Recording of the coordinates after the MATLAB script.
- Recording of the data from the radars before lateration.

[1]This part one done in cooperation with fellow students doing their master thesis for DMMS, specifically Andreas Berglihn [6] and Kenneth Kjelsås Strand.

- Improved risk functions
- Improved implementation of logical sensors (we can now see the logical dependencies of the logical sensors in a graphical interface)

The changes of the software were so extensive that we decided to stop focusing on the old software where we had made some modifications. Almost all the modifications we had made to the old software were now present in the new software, and there were also a lot of modifications that we had not considered.

3.2 Testing lab

We needed a place to explore what the radar can do and how it works. To be able to do this in a scientific manner, we have been given a laboratory in which we set up with two radars connected to a computer. This is a Windows machine with the software supplied by Novelda. Here, we try to understand how the radar works, what settings we should use and how they impact the signals received by the radar. We can see how different factors change the signals, like for instance, signal-to-noise ratio, how much noise we can have and still get good measurements. The bigger area we set the radars to cover, the more noise there is. We need to determine at what ranges the radars will produce "good" data. By good data we mean that we can still locate objects and get clear readings. If the signal-to-noise ratio is too high we will not be able to locate objects anymore.

The radars are at the moment fastened on the wall on the far side of the room so that our equipment that receives data from the radars does not interfere with our testing. The reason we have two radars is that we are working on triangulation by lateration as seen in Figure 2.2.

In Figure 3.1, we can see a two dimensional illustration of our testing lab setup. The circle with A and B inside represents the radars that are mounted on the wall. The distance between these radars are two meters in this drawing, as recommended by Novelda. The numbers one to twelve represents one by one meters grids. The grids one through four are not being used in any of our documented experiments, because these grids are too close to the radars so the triangulation script is not working as it is supposed to. We could have moved the radars closer together to try to counter this behavior, but we decided to focus on the radar results from the other grids. The general size of our testing area is decided by the size of the room we used as a testing lab. On top of Figure 3.1 we see 417cm, this is the total width of the room, and the 300cm number on the right side is the total depth of the grid. We have configured the triangulation script to not measure beyond this grid, to reduce interference from our equipment [2] that is located beyond this grid.

[2]Our equipment is a desk and chairs, our laptops, and the computer that the two radars are connected to.

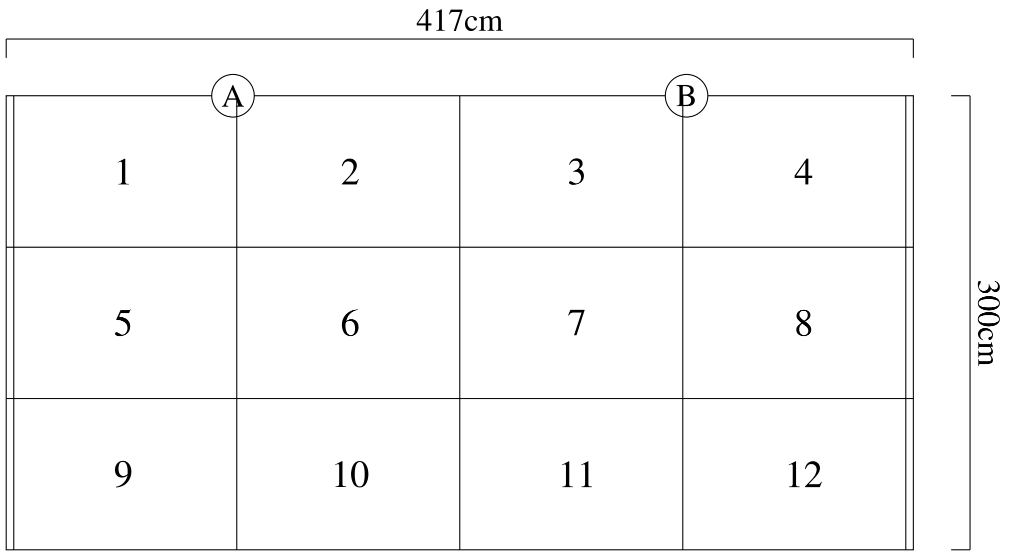
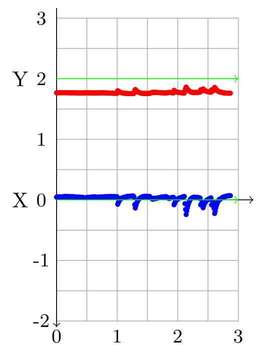


Figure 3.1: Illustration of the grid on the floor in our testing lab.

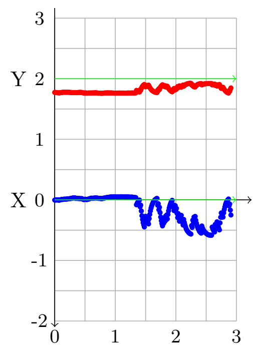
3.3 Experiments

In order to gain a better understanding of the radars and the radar software, we performed some experiments. The goal was to test the quality of the location coordinates produced by the MATLAB script. This was before TRIO v2 was done. Therefore, we first needed to modify the TRIO v1 software so that TRIO v1 would be able take its positional input from the radars. After this, we determined which tests that was needed to be done to be able to analyze the quality of the location data produced by the radars. We did different tests like standing still, moving in a specific pattern and placing inanimate objects in front of the radar. All of the experiments were done several times to normalize the results. A complete overview of the experiments we conducted in this phase is in Appendix A. To be able to evaluate the results from each experiment, we modified the TRIO software to be able to record the positions generated by the MATLAB script based on the radar data. The stored positions consisted of a frame number and coordinates, for both x-axis and y-axis for each frame number, we stored the result of each experiment to a file for later analysis. We used these experiment files and, with the help of LaTeX, generated graphs to illustrate both ground truth and the actual positions generated by the MATLAB script and radars. We used these graphs to evaluate the quality of the positions generated by the MATLAB script and radars.

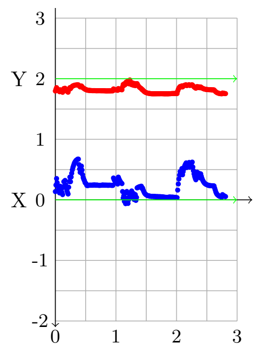
To illustrate these examples, we have included some of these graphs. We start with Experiment 8 and 9 from Appendix A, which is illustrated in Figure 3.2. What we can see in Figure 3.2 is a graph where the ground truth for x-axis and y-axis position is represented by the X and Y on the left side of each graph. The ground truth for x and y position are also illustrated in the graph by a green line for each axis, the green line for the x-axis is barely visible in Figure 3.2, this is because the recorded positions are very close to the x-axis ground truth position.



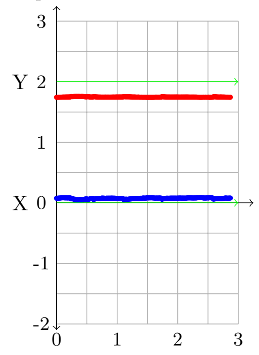
(a) Experiment 8.1 vertical placement



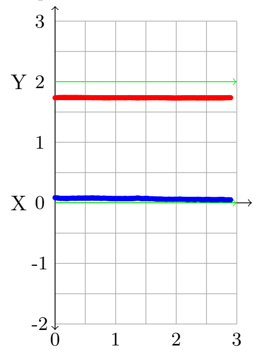
(b) Experiment 8.2 vertical placement



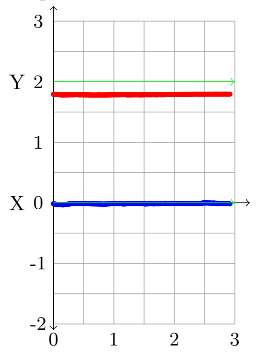
(c) Experiment 8.3 vertical placement



(d) Experiment 9.1. Horizontal



(e) Experiment 9.2. Horizontal



(f) Experiment 9.3. Horizontal

Figure 3.2: Person standing still 2 meters from the radars in center of the two radars, horizontal radar placement vs vertical radar placement. The horizontal line at Y is the ground truth for the y-axis. The horizontal line at X is the ground truth for the x-axis. Actual Y-axis is represented in red, actual X-axis is represented in blue.

However, we can see that the green line for y-axis is visible, this is because the recorded y-axis positions are a bit off the ground truth y-axis position. In these experiments, there is a person standing still two meters from the radars in the centre of the radars. The actual position of the person is the ground truth for our experiment, more specifically the ground truth for x-value is 0 and the ground truth for the y-value is 2. The difference between experiment 8 and 9 is that in experiment 8 the radars are placed vertically, and in experiment 9 the radars are placed horizontally to explore the difference in accuracy for the two different placements. We did each test three times, and we can see that the horizontal placement provides better results for this experiment. We have some off set for the y value but the results are quite consistent. The radars were, according to Novelda, supposed to have millimeter precision, so we assume that this is something that will get better as the triangulation script gets improved. However this experiment proved to be one of the most successful experiments that we conducted.

To illustrate for the reader the actual positions recorded we also have another figure to illustrate the actual readings of experiment 8 and experiment 9. In Figure 3.3, we can see experiment 8. We see the grid layout of the lab and a big blue circle that represents the actual location of the person(ground truth). The positions recorded in the experiments are illustrated with smaller circles in different colors, experiment 8.1 in red, experiment 8.2 in green and experiment 8.3 in cyan. In Figure 3.4, we can see experiment 9 illustrated in the same way.

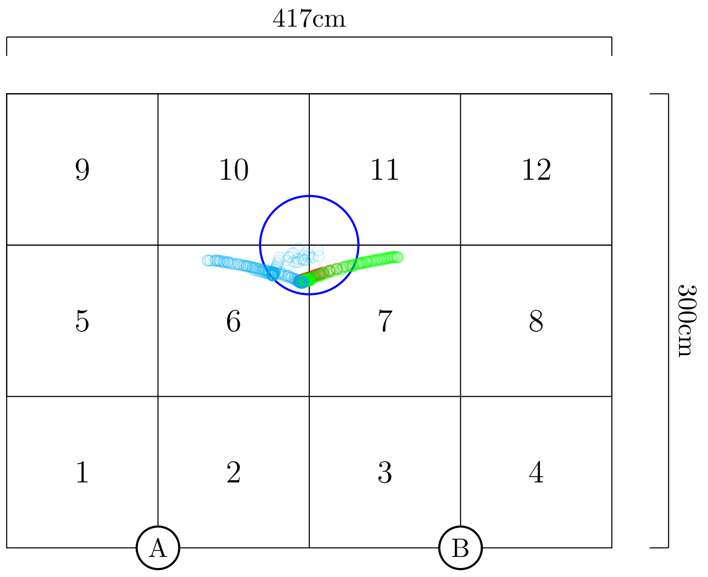


Figure 3.3: Experiment 8.1 - 8.3. Experiment 8.1 represented in red, 8.2 represented in green, 8.3 represented in cyan. Person standing still 2 meters in front of the radars in the centre. Vertical mounted radars. Results are not very precise but they offer a good indication of the location of the person, although they give the impression that the person is moving.

In Figure 3.5, we have done a similar experiment where we tested the accuracy difference between vertical and horizontal radar placement. In this experiment, a person have been standing still in grid number five as seen in Figure 3.1. This is experiment number 10 and experiment number 24 as seen in Appendix A. The ground truth for both the y-value and the x-value is in this case 1.5 as illustrated by the green line. We can here see that the both with horizontal and vertical placement we do not get very good results The x-value for experiment 10 is off with as much as one meters, and for experiment 24 the x-value is in the worst case off by over two meters. The y-values are off by one meter for both experiments. As we can see here none of the recorded values are where they are supposed to be. In Figure 3.6, we can see an overview illustration of the recorded results for experiment 24, and in Figure 3.7, we can see an overview illustration of experiment 10.

As we can see from these few examples we can get results indicating that the radars are tracking a person, if the monitored person is in the centre of the two radars. However when the monitored person move to a side or to close to the

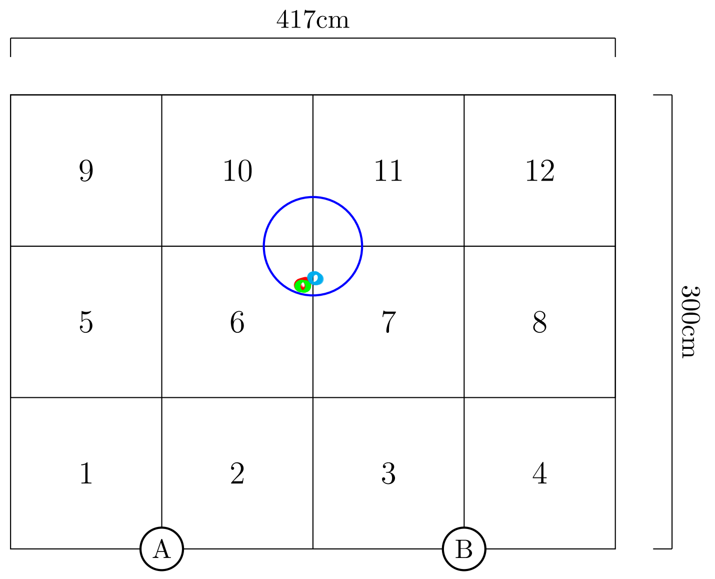


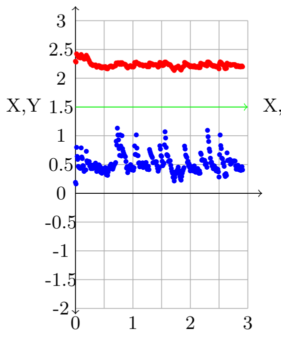
Figure 3.4: Experiment 9.1 - 9.3. Experiment 9.1 represented in red, 9.2 represented in green, 9.3 represented in cyan. Person standing still 2 meters in front of the radars in the centre. Horizontally mounted radars. Results are quite precise, they offer a good indication of the location of the person. We can see that the Y-axis is a bit too short, but good results for a general position.

radars the quality of the positional results decline.

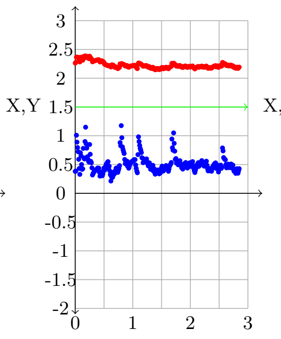
The problem with the positions produced by the radars is not only that the coordinates are not very accurate, it is that the coordinates suddenly spike meters from ground truth[3]. This is not good enough for a system such as TRIO which depends on correct positional data input to work properly.

As we can see in the graph in Figure 3.6, none of the positions are even within the grid in which the person is located. The results are at best one whole grid position off, and at worst over two grid positions off the position of the person. Each grid is one by one meters, so this means that the recorded positions are off by as much as two meters.

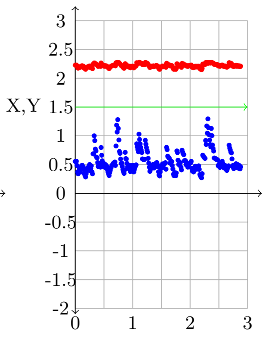
[3]Ground truth = the actual position



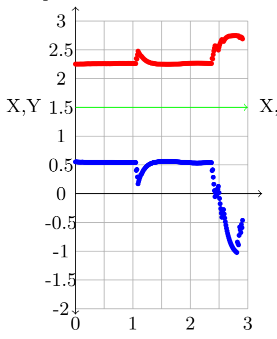
(a) Experiment 10.1 vertical placement



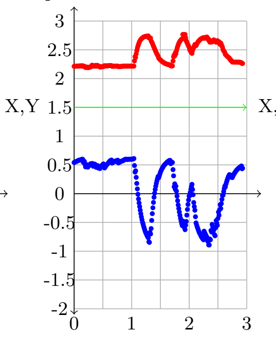
(b) Experiment 10.2 vertical placement



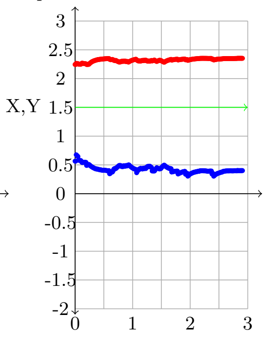
(c) Experiment 10.3 vertical placement



(d) Experiment 24.1. Horizontal



(e) Experiment 24.2. horizontal



(f) Experiment 24.3 horizontal

Figure 3.5: Person standing still in grid position 5, horizontal radar placement vs vertical radar placement. Green line at 1.5 is the ground truth for the x-axis and y-axis. Actual Y-axis represented in red, actual X-axis represented in blue.

2

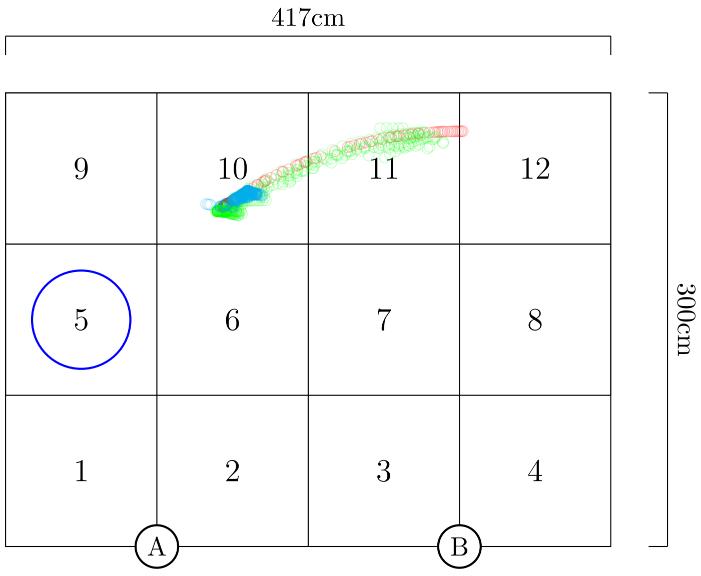


Figure 3.6: Experiment 24.1 - 24.3. 24.1 represented in red, 24.2 represented in green, 24.3 represented in cyan. Person standing still in grid 5. Bad results, not a single measurement inside grid 5.

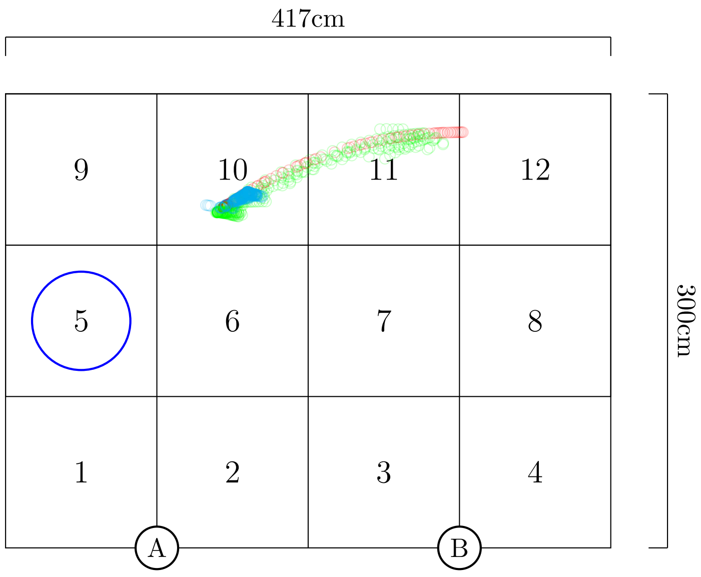


Figure 3.7: Experiment 10.1 - 10.3. 10.1 represented in red, 10.2 represented in green, 10.3 represented in cyan. Person standing still in grid 5. Bad results, not a single measurement inside grid 5.

1

3.4 Improved Radars and Other Systems.

The radar we tested in this thesis proved not to be good enough at this point, but it may be usable in the future, with new versions of the radar and improved triangulation scripts. TRIO has only been tested with this radar and a mouse for positional input, however TRIO may use any positional input source as long as it can deliver coordinates for the x and y axis together with a timestamp or tick counter.

There are many other systems on the market which can provide such an positional input. Ranging from bluetooth trackers, radio-wave trackers to other radars. One of these systems is called Bagadus [17]. All though the Bagadus system tracks people, it is made for another purpose. Bagadus is a system created for sports analysis, more precisely soccer. Bagadus utilizes sensors on the body of the players it is monitoring together with cameras and antennas, for both tracking and analysis.

As mentioned the Bagadus system is not created for home care, but that does not mean that the tracking part of Bagadus can not be used in other systems. The Bagadus system records movement and accelerometer data and video for each player on the field. What is interesting for our use case, is how they are tracking players on the field. The Bagadus system uses the ZXY Sport Tracking (ZXY 2013) [4] to track the players in the field. ZXY Sport Tracking uses rf (radio frequency) based technology to determine the position of the wearable transponders the players are wearing. This tracking system requires that there are receivers in the vicinity and the people that should be tracked needs to be wearing a transponder.

Another interesting part about Bagadus is how it uses the positional data gathered from the ZXY tracking system. The Bagadus system takes the positional input data from both the cameras and the ZXY system it synchronizes and combines them. This allows us to follow specific players from the field with the cameras. Maybe this approach could be used for TRIO as well. By having separate tracking systems working together, and then combine and synchronize the results we could achieve the positional accuracy TRIO requires. We could for instance, add more radars to do independent triangulation, and then synchronize compare the results to increase accuracy, but this is out of scope for this thesis.

There are some drawbacks to the tracking system for our usage, as it requires the persons that is to be tracked to wear a transponder. The constant usage of such a transponder will have some consequences good and bad, such as.

Cons

- The transponders will have to be recharged.
- The user may forget to wear the transponder.
- The transponder may be uncomfortable to wear all the time.

[4]ZXY: http://chyronhego.com/sports-data/zxy

Pros

- Precise tracking.
- Possibility for other metrics from the same system, such as heartbeat.
- Possibility for accelerometer and altimeter for easier fall detection.

Another point is that elderly people often have to get up at night to go to the bathroom, and with a system like this, they would have to remember to take their transponder with them. This is something that is easily forgotten, and persons with e.g. dementia might forget to use the tracker at all. This could be a false sense of security for the user if they are not wearing the transponder at all times.

If a tracking system like this is a viable substitute for the radar will have to be explored more in detail, but this is out of scope for this thesis.

At this time we have only discussed non optical tracking systems, the reason is that with an optical tracking system invasion of privacy is a big concern, as the data feed may be intercepted or misused. If invasion of privacy was not a concern we could have used cameras in every room to keep track of the person. We could have used tracking technology that depended on optical imagery to obtain a persons location in their apartment. There are many such systems developed over the years, for instance, at the University of Oxford had a project with the title Stable Multi-Target Tracking in Real-Time Surveillance Video [5]. In this project they used cameras to track peoples location. The interesting part about this project is that they where able to track multiple persons while being able to tell them apart. A system such as this could be used in TRIO to track multiple persons in an apartment. However such a system is not usable in TRIO because of the invasion of privacy concerns.

3.5 Summary

After many tries with different approaches and different settings we decided that the coordinates produced by the MATLAB script are not good enough to be used for the development of logical sensors because they require high accuracy.

We have brought this information back to Novelda, and we have had a technician from Novelda to explore the problem. The technician from Novelda was surprised by the bad results we were getting. We do not have a lab that is made for this purpose, we are utilizing a room at UiO. So, either way it is useful data for Novelda to see how the scripts and radars behave in different environments.

The technician from Novelda made some modifications to the MATLAB script, both to try to improve triangulation and to record the data that the radars produce both before and after triangulation. So, we now have the possibility to record experiments. Both the coordinates produced by the MATLAB script and the raw data that comes directly from the radars. This helped some, but not enough for the use of data input from the radars. This made us move our focus away from the radars and instead focusing on improving TRIO.

Large deviations in the coordinates like this would make the logical sensors produce false positives which reduces the reliability of the system. We have noticed some other strange behavior as well, such as after some time the triangulation script seems to loose track of us. Another thing we noticed is that if there is nothing in the radar range, the triangulation script still produces positions, giving the impression that it is tracking something, even though there is nothing to track.

In the next Chapter, we move on to contexts. We explain what contexts are per definition and in computer science, as well as how contexts are implemented in TRIO. Then, we explore how TRIO utilizes contexts for risk calculation using risk functions.

Chapter 4

Context

The concept of Contexts is important in the TRIO system. If we look up context in the Oxford [10] dictionary, we find the following definitions:

- the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood: *the proposals need to be considered in the context of new European directives.*
- the parts of something written or spoken that immediately precede and follow a word or passage and clarify its meaning. *skilled readers use context to construct meaning from words as they are read.*

4.1 Context in Computer Science

To explain how context is used in computer science, we need a definition for the concept of context. This is a much used definition in computer science [9]:

> Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

This definition helps us to recognize what context is, and what it is not. According to our understanding of the definition we only view the information that is relevant for the entity as a context for the entity.

Let us explain the definition by using an example: Imagine that we are developing a mobile application, more specifically a weather forecast application. We start with identifying which of the available mobile sensors we should use for this application. We only need the sensors that are relevant for the application. We quickly realize that many of the available sensors are not relevant for our application, such as the accelerometer in the phone (sensor that measures the movements of the phone). However, the GPS sensor which can tell the application the location of the user is very relevant. This is because the user often want the weather forecast for the user's current location. Consequently, we identify the user's location as a context variable for the application, which in this case would be the entity. A context is a way to differentiate all available information from the information that is relevant for the entity.

4.2 Context-Awareness

Related to the concept of contexts is the concept of context-awareness which determines a system's ability to recognize contexts and change its actions according to the context changes.

> A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [9]

This is one definition of context awareness. By this definition a system is context aware if it uses context either to supply the user with relevant information or services.

Another definition is “ The ability of a computer system to sense details of the external world and choose its course of action depending on its findings. ” [8] By this definition, we explain a context-aware system like this: A context-aware system is a system that adapts its behavior and/or actions when something the computer system can "sense" in the external world changes.

This definition from B. Schilit, N. Adams and R. Want is also a much used definition:

> Context-aware systems adapts according to the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time. A system with these capabilities can examine the computing environment and react to changes to the environment. [18]

In this definition, the authors has explained what they consider to be contexts, but the meaning of the definition is similar to our first definition above. It is a systems ability to comprehend changing contexts and change its actions accordingly. In the first definition from A . K. Dey the system is only able to provide information and/or services to the user, while in this definition the system can also react to the changes in the environment regardless of the user. This definition fits our situation better since we do not always define the user as the entity for the system or context.

To clarify what we define as context-aware systems, consider the following example. We have a room that is equipped with a sensor for turning on the lights. This sensor measures two parameters: how much light there is, and if there is movement. The sensor only turns on the light if there is less natural light in the room than a certain threshold (the room is "dark"), and there is movement in the room. The context in this situation is the amount of light in the room and if there is movement in the room, which means that we have two context variables. The default situation is that the light is turned off, but if certain context variables supersede specific limit values the sensor will turn on the light. In this example, the entity is the sensor.

4.3 Context-aware Ambient Assisted Living

An AAL system as explained in Section 2.1, is being used to automate home care. A context in AAL is all about making the system able to recognize different events, situations and scenarios, and then adjust its actions accordingly. This means that we use context to help the system make smarter decisions, and the more specific contexts we define, the better the system will be able to interpret different events, i.e, leading to a more precise and reliable system.

To illustrate how using contexts leads to a more precise and reliable system we give an example:

Assume we have an AAL system where we measure the heart rate of a patient, and a certain number of heart beats per minute (HBPM) is normal. A limit for normal HBPM could be set very wide, from the minimum HBPM that a human can survive to the maximum HBPM that a human can survive. An AAL system configured like this would then notify health personnel if the HBPM went below the min or above max thresholds. This means that the system will only alert health personnel if there is a crisis. We want a system that alerts health personnel before it becomes a crisis, i.e. we want a proactive system. To be proactive, the system has to alert health personnel at multiple max and min HBPM thresholds. To determine appropriate min and max values for alerts, the system will need to take the patients health condition and actions into consideration. A person's actions can have an impact on a persons HBPM, e.g. if the person is running on a thread mill or laying on the couch, it would lead to higher or lower HBPMs without anything being wrong. This is where contexts come in, we can view the persons actions as different contexts, and have different warning thresholds for each context. This makes the system context aware and more precise, which again leads to fewer unnecessary notification of health personnel.

4.4 Context in Esper

What is important to understand in this chapter, is that Esper does not define context the same way as we do in this master thesis. Esper supports a concept of context as a cloud of events and classifies them into one or more sets. These sets are called context partitions. An event processing operation that is associated with a context operates on each of these context partitions independently. [13] Context in Esper is a set of circumstances or facts that surround a particular event, situation, etc. So, by Esper's concept, we see that Esper uses one or more events to create event sets which we view as context partitions. A context in Esper is created from one or more context partitions, and the only limitation for creating contexts in Esper is that we must be able to define a start and stop event for the context. The impact of context partitions is that we can make contexts that depend on other contexts, which lets us partition up the context creation and lets us reuse contexts. This helps us create more elaborate and complex contexts. Events are defined in Esper using EPL, which is Esper's SQL-like query language.

Contexts in Esper are used as follows: We add contexts to Esper using EPL to define the parameters for when the context should start and stop. The start and stop can be a special situation, event, or even a time period – it can be anything we want, as long as we can define it with EPL. Which events we want Esper to view as contexts depends on our goals with the software, our specific situation and how we wish to implement contexts in our system.

Contexts in Esper are only used to decide when other EPL statements should be active, in our case an EPL statement can be a logical sensor. So a context by itself with no EPL statements referring to it will do nothing. Like if we have an EPL statement which we only want to be active under certain conditions, we can create an EPL statement which defines these conditions then add this EPL statement to Esper as a context, then we add the context we just created as a requirement for the first EPL statement.

It's worth mentioning that a context consumes no computing power by just being added to Esper, its logic will only be active when an EPL statement refers to it. A logical sensor that depends on a context does not consume any computing power while the context is inactive.

Earlier we have viewed contexts as concept and not from an implementation point of view, but contexts in Esper is about how we implement what we define as contexts into Esper. Esper supports a liberal view of what a context can be, i.e. we can easily implement our conceptual contexts straight into Esper as Esper contexts.

4.5 Contexts in TRIO

We now explain how contexts are used in the TRIO system, both from a conceptual view and an implementation view, and how context is used by different parts of the system. As we know from the previous sections in Chapter 4, context is any information that can be used to characterize the situation of an entity. The goal of TRIO is to detect abnormal behavior and alert health personnel if abnormal behavior is detected. To achieve this, TRIO needs to be able to differentiate normal behavior from abnormal behavior, and what is normal behavior depends on the context, because a person behaves differently in different contexts. In TRIO, a risk function determines the risk of abnormal behavior, and to give the risk functions more accurate risk determination they are often tied up to specific contexts, i.e. that many risk functions are only active in specific contexts.

In TRIO, we currently use two conceptual context variables to characterize the context the person is in: *zones*, and *period of day (POD)*. The context *zones* will be explained in Section 4.6, and the context *POD* will be explained in Section 4.7. The apartment that is monitored by TRIO is divided into zones, where each room in the apartment represents a zone. We have done this because normal behavior depends on which room the person is in. For instance, if a person is in the bedroom and not moving, we would view this as normal considering the context of the person being in the bedroom.

In TRIO the context POD is divided into three eight-hour periods: day, evening, and night. This is because a person's behavior changes depending on the time of day. Behavior that is normal during the day, may not be normal during the night. For instance, if the monitored person is not moving for several hours, it would be considered normal during the POD night, but abnormal during the POD day. In TRIO, we can combine the POD and zone context to get more specific contexts. Contexts can be used as a criteria for when a risk function should be active. The risk functions will be explained in more detail in Section 4.8. For example, if the monitored person is in the living room all night instead of the bedroom, we assume that something is wrong, and so should TRIO. To achieve this, we have a risk function that uses the POD night and that the person is in the zone living room as an indication of abnormal behavior and alert health personnel.

The zone and POD which the person that is monitored by the TRIO system resides in, currently will be referred to as ”current zone” and ”current POD”.

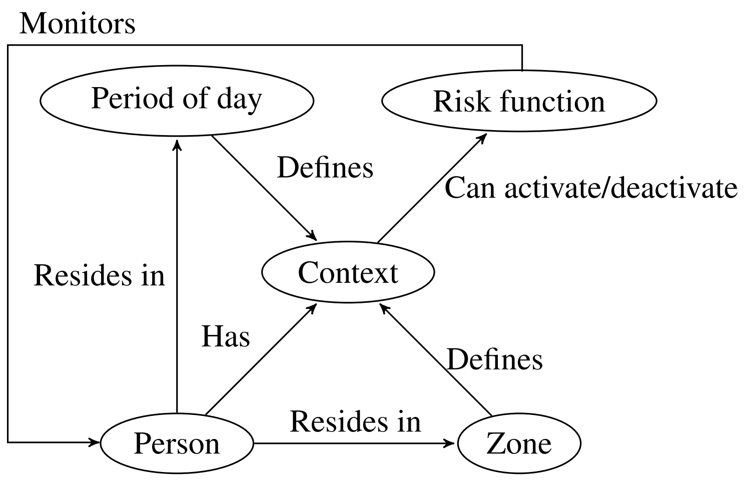


Figure 4.1: How different parts of TRIO is connected to context

In Figure 4.1, we see a drawing which illustrates how all the different parts of TRIO are connected to context. Contexts are the common ground which binds all the parts of TRIO together. The main parts of the TRIO system connected to context are:

- POD,
- risk function,

- person, and
- zone.

As we can see in figure 4.1 a person has a context, and this context is defined by the current zone and/or POD. If the context that a person is in matches the context set in the definition of a risk function the risk function will activate, unless it is manually disabled. To illustrate this, let us assume that we have a risk function that alerts health personnel if the person monitored by the system is in the bathroom for more than one hour during the night. This risk function will only be active in the right context, which in our example is when the current zone is ”bathroom” and current POD is ”TRIONight” [1]. The risk function will start measuring the duration of stay when the current zone is "bathroom" and the current POD is "TRIONight", if the duration of stay exceeds one hour TRIO will alert health personnel.

We spent a lot of time analyzing the TRIO system for weaknesses, to explore what could be altered to improve how TRIO calculates risks. We found some weaknesses regarding the zone and POD context. The weaknesses we uncovered, and how they can be improved are explained in detail in Section 4.6 and 4.7.

4.6 Zone Context

In this Section we first explain the current implementation of the zone context. Then, we move on to an example of the weakness in the current zone context implementation, followed by a suggestion of a possible improvement. Finally we discuss some possible improvement implementations.

4.6.1 Current Implementation of Zone-Context

In TRIO, we divide the area that is being monitored by the radars into different zones. In our TRIO setup we have four zones which are one square meter each, these four zones represent the apartment. In order to utilize the aforementioned zones, TRIO must be able to track where in the apartment a person is, TRIO contains several logical sensors for this purpose. To give a clear understanding of how this works, we explain this from an implementational point of view.

There are many sensors that are a part of the tracking in TRIO, but the most important one regarding zone presence is the "ChangeZone" sensor. The "ChangeZone" sensor takes its input events from another logical sensor called "FilteredZone", which generates one event per recorded position, continuously indicating which zone the monitored person is located in. The events produced by the "FilteredZone" sensor contains a zone id. When an event contains a different zone id than the previous event, the "ChangeZone" sensor detects it and changes the current zone accordingly. The EPL for this sensor looks like this:

[1]TRIONight is the name of the POD in TRIO which covers night time

Listing 4.1: Example EPL: Changing zones TRIO

```
insert into ChangeZone
select z1 as previous,z2 as current,
current_timestamp() as ts
from pattern
[
every
(z1=FilteredZone -> z2=FilteredZone(zone.id != z1.zone.id))
]
```

In Listing 4.1 we have two input events, z1 and z2. Z1 is the zone the last position was in, and z2 is the zone the current position is in. If z1.zone.id does not match z2.zone.id it means that the monitored person has moved from one zone into another zone.

Listing 4.2 shows a simplified EPL statement that adds a risk function, specifically, we here see the risk function that determines if the person has resided in the bedroom for a duration above a certain threshold during the POD "TRIODay". The "TRIODay" contexts are added as context in Esper, and is therefore specified by adding the context TRIODay as seen in line 1 in Listing 4.2. Then, we create a window called FilteredRiskTooLongInBedroomAtDay where we store the values ri and ro. The ri variable is the current risk input, meaning the current duration in the zone bedroom, the ro variable is the risk threshold, i.e. the duration that have been set as a limit for normal behavior. The two variables ri and ro are used for calculating risk, if the value of ri exceeds the value of ro the risk function will alert. It is worth mentioning that the logical sensor that records the duration of stay in zone "bedroom" is defined in another EPL. This is a simplification but it illustrates how a risk function can be added to TRIO.

Listing 4.2: Example EPL: Person residing to long in bedroom during daytime

```
1 context TRIODay
2 insert into FilteredRiskTooLongInBedroomAtDay
3 select * from
4 RiskInputTooLongInBedroomAtDay.std:lastevent() as ri,
5 RiskOutputTooLongInBedroomAtDay.std:lastevent() as ro
```

In Figure 4.2, we illustrate what the current zone implementation in TRIO supports. We have 4 zones which are numbered, zone 0 is outside of the apartment, and zones 1 - 3 are rooms inside the apartment. The arrow in the figure illustrates the movement of a monitored person. Before a person enters the apartment, TRIO assumes that the person is outside, meaning that the person resides in zone 0. When the person enters zone 1, TRIO will see this as they are leaving zone 0 and entering zone 1. This leads to a ChangeZone event, the ChangeZone event is caught by the ZoneListener which lets TRIO set the new current zone. Whenever a person enters a new zone it will leave the zone from which it came, this means that a person can only reside in one zone at a time.

Figure 4.3 is related to Figure 4.2, the arrow in Figure 4.2 illustrates the zone changes illustrated in Figure 4.3. Each zone change triggers a ChangeZone event

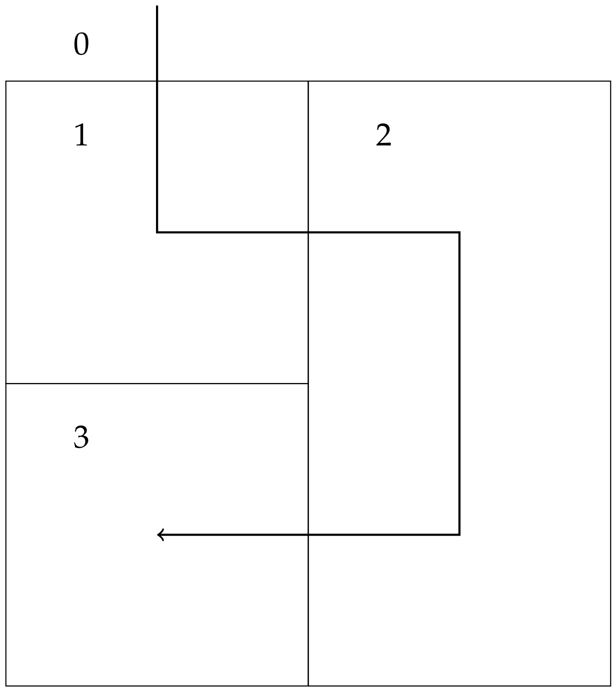


Figure 4.2: Allowed zones in the current implementation

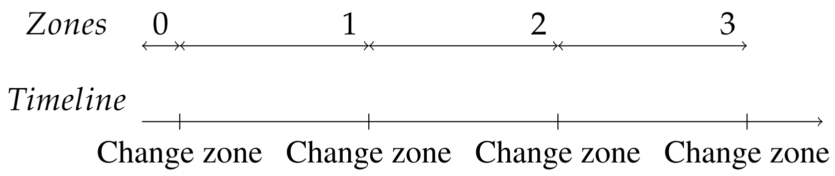


Figure 4.3: Timeline for zone transitions in the current implementation

and the monitored person is only in one zone at the time.

4.6.2 Example of Current Zone-Implementation Weakness

Now that we know a bit about how the zone contexts works, we illustrate why the current implementation has a weakness. This weakness is best explained by an example: Let us assume that we have a logical sensor that alerts health personnel if the monitored person is sleeping for too long. For this risk function the zone context would be bedroom and the risk function would have to work like this:

If the monitored person resides in the bedroom for a longer duration than it is considered normal to sleep, the risk function in TRIO will notify health personnel. This is all good and well when the monitored persons reaches their bed. However if we imagine that the monitored person has fallen on their way to the bed, and is unable to get up, it will be a long time until health personnel is alerted by TRIO.

If a person is lying on their bedroom floor for eight hours we consider this to be abnormal behavior. If a monitored person has fallen and is unable to get up, it is important that the TRIO alerts health care personnel. With the current implementation of the zone context, TRIO has no way of knowing if the monitored person resides on the bedroom floor or in the bed. With this example, we have illustrated why the current implementation of zone context in TRIO has a weakness regarding proper risk assessment.

4.6.3 Possible Improvement of Zone-Context

In Section 4.6.2, we illustrated a risk function that had some weaknesses regarding proper risk assessment, in this SubSection we will introduce a possible improvement to amend this weakness.

With the current implementation of the zone context in TRIO a person may only reside in one zone at a time. However, if we improve the zone context in TRIO to allow for nested and overlapping zones, a person could reside in more than one zone at a time. With this functionality we could add all furniture in the apartment as independent zones. If all furniture in the apartment are defined as zones, TRIO would know whether a person is located on a piece of furniture or on the floor.

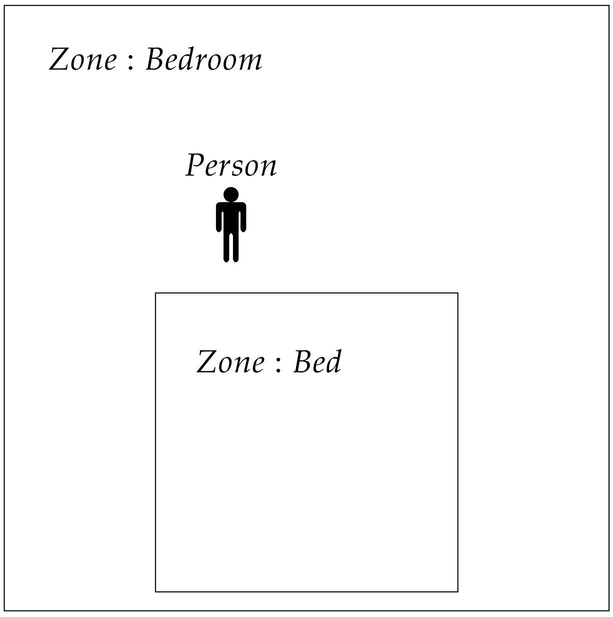


Figure 4.4: Bedroom zone with bed zone and person.

In Figure 4.4, we have an illustration of how TRIO would see the bedroom zone after such an improvement, the bed zone is located at the bottom of the bedroom zone and a person is located on the floor. With this new functionality, we can modify the aforementioned risk function so that it uses the context zone bed instead of bedroom. Moreover, we can add a new risk function that should contact health personnel if a person is in the bedroom zone, and not in the bed zone and is not moving for a certain amount of time. If the person is not moving while in the bedroom zone, but not within the bed zone the system should notify health personnel much faster, as this is an indication that something is wrong. For instance, if a person is beside the bed zone for a prolonged period of time that may indicate that the person has fallen and is in need of assistance.

By this example, we have illustrated why this modification to the system can be an improvement. This functionality gives the system more accurate risk assessment possibilities which provides additional security for the monitored person.

4.6.4 Possible implementations

There are several ways we can implement the zone context to allow for nested and overlapping zones. Before we discuss these it is worth mentioning that the implementation of zones within zones will have consequences for the visual representation of zone metrics. The simulator in TRIO will get cluttered with information about duration in zone and number of times in a zone and so forth. Therefore, the simulator will have to be modified to accommodate these changes.

Here we discuss three possible zone improvement implementations:

1. Change the way TRIO perceives a change of zones: When the monitored person moves from one zone into a zone that is within another zone, TRIO should instead of leaving the previous zone stay in it and enter the new zone as well. To accomplish this TRIO can no longer utilize change zone events when entering a new zone, as a change zone event leaves the current zone and enters the new zone. Instead we would have to implement and utilize a start zone event when entering a zone, and a stop zone event when leaving a zone.

2. In the current implementation of TRIO, a zone is a conceptual context but it is not defined as a context in Esper. We could rewrite TRIO so that both POD and zone are defined as contexts in Esper, then, each coordinate can belong to one or more zones. To rewrite TRIO in such a way will bring the implementation closer to the concept definition, but it may lead to other problems.

3. We can change the ways zones are built up, so that a zone can have a parent zone. This way TRIO can check if the current zone has a reference to a parent zone, this way we should be able to support nested zones in TRIO. This approach will however make it harder to introduce overlapping zones in TRIO.

Design Alternative one The first alternative is most likely the most correct way of implementing the zone expansion. The reason is that the whole system does not have to be rewritten, and this approach will support both overlapping and nested zones. However TRIO will need major rewrites regarding all zone logic.

Design Alternative two This alternative however, will require a more fundamental change to how the zones are implemented and used in TRIO. By using this approach one could assume that this implementation would be more similar to the POD improvement implementation. This makes a lot of sense, but it is hard to determine how complex and time consuming this implementation would be as it is not clear what other parts of the system would have to be rewritten as a result of it. We quickly dismissed this alternative, because it would be such a substantial rewrite of all EPL and zone logic in TRIO.

Design Alternative three We need to explain this alternative in some greater detail, we find this best to do with a model: As we can see in Figure 4.5 the space with the number *1* is Zone *1* and within this zone we have zone *2* and within zone *2* we have zone *3*. The grayed out circle with an *x* is one possible position a person could be in, in this position the person is in only zone *1*. Another position a person could be in is the position *y*, this position is in all three zones and TRIO would know this because position *y* is in zone *3*, and zone *3*'s parent is zone 2 and zone *2*'s parent is zone *1*. In other words the parent id reference in each zones links to the parent zone.

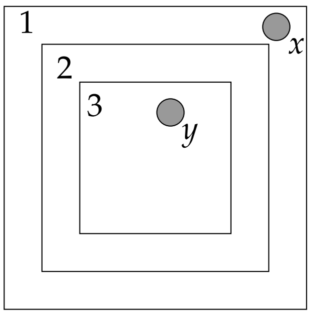


Figure 4.5: Alternative three zones within zones

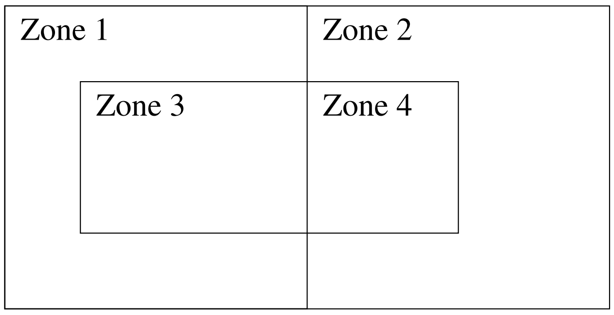


Figure 4.6: Alternative overlapping zones

We mentioned that overlapping zones could be a little more to difficult achieve, this is because each zone can only have one parent zone. However in Figure 4.6 we propose a workaround for this issue. In Figure 4.6, zone 1 and zone 2 are two regular zones, zones 3 and 4 are zones within their parents zones. If we view zone 3 and zone 4 as a whole, we can view these zones as one overlapping zone. The reason for this is that we can add more than one zone as a context for a risk function. By extension this means that zone 3 and 4 will be viewed as one context by the risk function, which means that overlapping zones are possible to achieve with this implementation as well. The usage scenario for an overlapping zone is not that big. Overlapping zones may however be an important functionality in the future, therefore we wanted to illustrate that it is possible to achieve with this implementation as well.

4.6.5 Suggested Implementation

After some consideration, we have decided that the first alternative is the best choice. The first alternative lets us add zones completely independent of existing zones, which makes adding new zones easier, and more intuitive. The second alternative would also let us add zones independent of other zones, but it would required an almost complete rewrite of TRIO. The third alternative was not chosen because of the clumsy handling of overlapping zones.

Here we explain the first alternative in detail: The timeline illustrated in Figure 4.8 illustrates how the changing of zones in the Figure 4.7 is perceived by TRIO after the improvement. The timeline in Figure 4.8, is an illustration of the zone transitions occurring if a person is following the path of the dashed line from Figure 4.7. Each time we enter a zone, a StartPeriod event occurs and each time we leave a zone, a StopPeriod event occurs. When a person following the dashes line from Figure 4.7 enters zone 1, there is, as we can see in Figure 4.8, a StopPeriod event for zone 0 and a StartPeriod for zone 1. Although this is now implemented differently, the end result will be the same: we will leave zone 0 and enter zone 1.

However, when the person enters zone 4 from zone 2, there is a StartPeriod event for zone 4, but no StopPeriod event for zone 2, and the person is now located in both zone 2 and 4. When the person leaves zone 4 there is a StopPeriod event for zone 4, and the person is now only located in zone 2. There is no StopPeriod event for zone 2 until the person enters zone 3. This way we have illustrated that with this implementation, TRIO will support both nested and overlapping zones.

The start and stop periods when a person is entering and leaving a zone in Figure 4.8 is different than the current implementation illustrated in Figure 4.3. In the old implementation of zones, each time we enter a new zone we leave the previous zone. This is because the old implementation utilizes ChangeZone events which change the current zone if the monitored person enters a new zone. With the new implementation entering a zone generates a StartPeriod event for that zone, and leaving it generates a StopPeriod event for that zone.

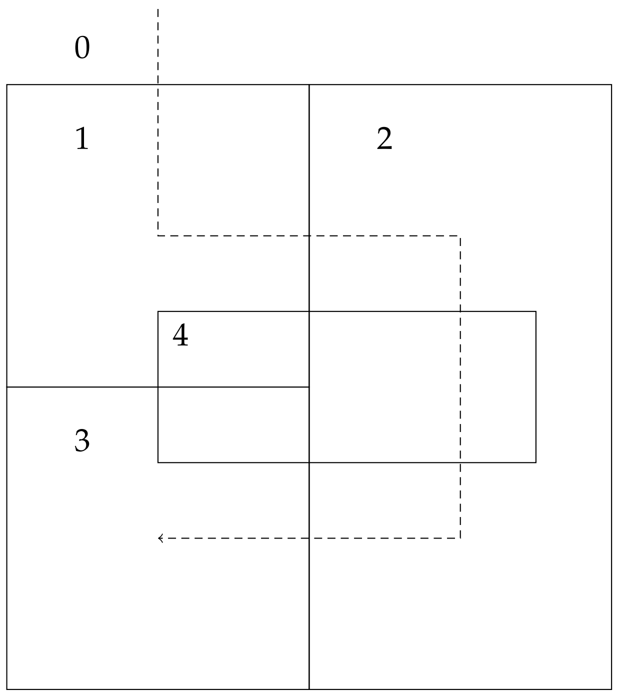


Figure 4.7: Illustration of improved zone implementation

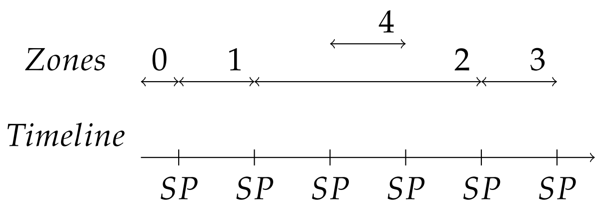


Figure 4.8: Example of timeline for zone transitions improved implementation

4.7 Period of Day Context

In this Section we will explain the current implementation of the POD-context, then we will show an example of the current POD implementation. Followed by a possible improvement of the POD-context.

4.7.1 Current implementation of POD

The period of day contexts in TRIO is explicitly added as contexts to Esper, and they are defined like this: Daytime lasts from 08:00 to 16:00 followed by evening which lasts from 16:00 to 24:00 and night which lasts from 24:00 to 08:00. An overview of the current periods of day is illustrated in Figure 4.9, but here the periods of day uses the names they have in TRIO.

When the time of day reaches 16:00 TRIO changes the current POD from TRIODay to TRIOEvening. When TRIO switches the current POD from

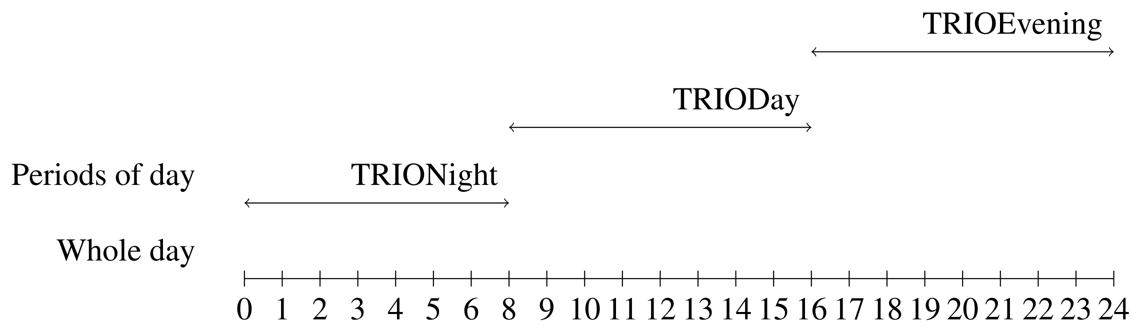


Figure 4.9: Illustration of periods of day, current implementation

TRIODay to TRIOEvening, the simulator generates an StopPeriod event for the current POD. Which in this case is TRIODay, and a StartPeriod for the next POD, which in this case is TRIOEvening. After this operation, the current POD in TRIO is now set to TRIOEvening. Meaning that there is only support for one active POD at a time in the current implementation of POD in TRIO.

Listing 4.3: Example EPL: Adding context TRIODay to Esper

```
create context TRIODay
start
org.trio.ContextStartPeriod(name = "TRIODay") as periodStart
end
org.trio.ContextEndPeriod(name = "TRIODay") as periodEnd
```

In Listing 4.3, we see how the POD TRIODay context is added to Esper. The simulator generates start events for the beginning of each POD and end events at the end of each POD. The POD have descriptive names that describes the POD as we can see in Listing 4.3. The start of the context TRIODay is when the simulator generates a StartPeriod with the name TRIODay, and it ends when the simulator generates a StopPeriod with the name TRIODay.

4.7.2 Example of Current POD-Implementation Weakness

In the previous SubSection 4.7.1 we explained the POD context's implementation in TRIO. We know that a day is currently divided into three POD's, more specifically TRIODay, TRIOEvening and TRIONight. We also know that with the current implementation there can be only one active POD at a time.

We believe that this is a weakness regarding risk assessment in TRIO, the following explains why: One of the uses of TRIO, is to gather statistics of the behavior of the person that TRIO is monitoring, such as, sleeping patterns, number of times in the bathroom and so on. With the current implementation TRIO is not able to gather any statistics for the whole day, as any statistics TRIO gathered is reset when one POD ends and another begins. Therefor, it is not possible at this time for TRIO to gather statistics of the behavior of a monitored person beyond the active POD when utilizing PODs. This has some ramifications for the risk assessment in TRIO, for instance: TRIO has no way of knowing how many times a person has been in the bathroom during a day, or how long for that matter when using the logical sensors from the PODs. If TRIO gets expanded further and it stores the recorded statistics for each POD in for instance a database, we would be able to aggregate these results later on. However, TRIO would not be able to react to them as they are collected, which can lead to incorrect risk assessment. We give two examples to illustrate when this can lead to inaccurate risk assessment.

Example one: If a person is in the bathroom for a total amount of two hours during TRIODay, two hours during TRIOEvening and two hours during TRIONight, this would not be considered excessive if TRIO views these durations separately. However, if we combine these durations we can see that the monitored person has been in the bathroom for a grand total of six hours during one day, which we find to be quite excessive. With the current implementation of POD in TRIO we are not able to measure this using POD's.

Example two: The POD's in TRIO are in the current implementation divided in to three eight hours PODs, and the POD called TRIONight is active from midnight until 8 am. A useful statistic to gather when monitoring a persons behavior is sleep patterns, and we would also want TRIO to alert health personnel if a monitored person is in the bedroom for a duration above a certain threshold. Let us imagine that the monitored person has been in bed for over 10 hours, and let us assume the threshold for contacting health personnel is 10 hours. With the current implementation of POD in TRIO, TRIO would have no way of knowing if a person has been in bed for more than eight hours, because the duration of the POD TRIONight is only eight hours. When the TRIONight POD ends, the duration of stay metrics gets reset as they are only gathered for the currently active POD. This means that for a risk function such as this to work, we would have to change the duration of the POD TRIONight to be more than 10 hours. Unless we make the duration of TRIONight very large, the monitored person would have to go to bed within a short time frame in order for the risk function to work properly. To accomplish this, we have to make the duration of the other PODs in TRIO smaller, which will impact the behavior and effect of the other risk functions in TRIO.

These examples illustrate how only being able to have one active POD at a time limits risk assessment in TRIO.

4.7.3 Possible improvement of POD-Context

In this section, we introduce a possible improvement of the POD implementation, we begin by explaining what we wish to improve. We wish to be able to calculate the risk of a persons behavior, even though the persons actions reach beyond the current POD's in TRIO. To achieve this, we wish to add functionality of nested and overlapping PODs to TRIO, which will let us be able to add POD's independent of existing POD's. In Figure 4.10, we illustrate how both overlapping and nested PODs can be added to TRIO with this extension of POD. For instance, in Figure 4.10, we have added two new PODs: TRIOWholeDay and TRIOSleep, TRIOWholeDay is a new POD that covers the entire day, and TRIOSleep covers

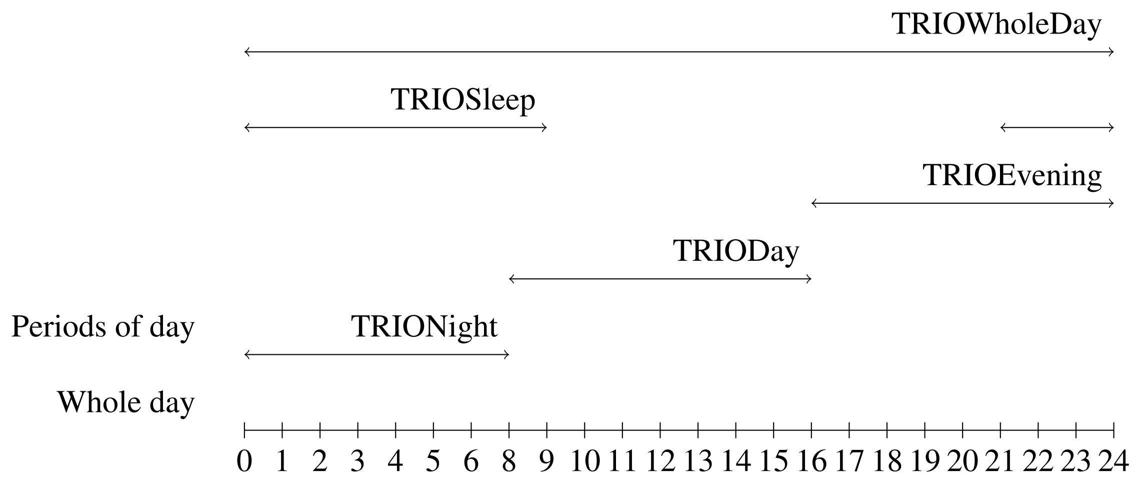


Figure 4.10: Illustration of periods of day, improved implementation

the timespan in which it is expected a person to sleep.

This improvement will let the TRIO system react to events from the whole day or other more specified PODs, leading to more accurate risk calculation. Let us review the examples from the previous SubSection with the new functionality in mind: In example one from Section 4.7.2, we explained how TRIO was not able to collect the total duration of stay from the bathroom zone during a whole day. With this improvement however, we can use TRIOWholeDay as a context POD for the risk function. TRIO would then be able to detect if a person was in the bathroom for too long during the entire day.

In example two from Section 4.7.2, we explained the problem with detecting sleep duration anomalies due to the duration of the POD TRIONight. In Figure 4.10, we have added a new POD called TRIOSleep. The new POD TRIOSleep can be used by risk function mentioned in example two, as we can see in the figure TRIOSleep covers a considerably larger timespan than TRIONight. TRIOSleep specifically covers the timespan from 21 p.m to 9 a.m on the next day, which is a timespan of 12 hours, this can however be changed to accommodate the sleeping routines of the monitored person. The risk function from example two can use this POD to be able to alert health personnel if the monitored person is in bed for over 10 hours.

These examples illustrate how the support of zones within zones and overlapping zones increases the accuracy of the risk calculation in TRIO.

4.7.4 Possible implementations

Now that we have a clear understanding of what we wish to improve, we focus on the actual solution of the problem. First we studied the current implementation of POD in TRIO and how they are connected to logical sensors that records zone metrics for each POD. We explored what would be needed to be changed for the

TRIO system to be able to support nested PODs and overlapping PODs. During this process we learned that the logical sensors for recording zone metrics, and the simulator, was the main challenge. To explain the challenges with these parts of the implementation we briefly explain them:

A logical sensor for zone metrics is for instance, the CurrentStayInPeriod sensor which measures the duration a person has been in a zone in the current POD. Another logical sensor for zone metrics is the NumberOfTimesInZone which measures how many times a person has been in a zone for the current POD.

The simulator in TRIO is a graphical representation of what data that is available to TRIO. Later Figure 5.2, we can see an overview of the simulator, where in the top left we have a visual representation of the zones in the apartment that TRIO monitors. In the zone overview we can see metrics like the number of times a person has been in that zone within the current POD, and the duration the person has been in a zone in the current POD. In the bottom left of the figure, we can see an overview of all PODs, and active PODs are highlighted in blue. On the right side of the figure we can see an overview of all logical sensors in TRIO, active PODs are highlighted in red.

The challenges with these parts of TRIO regarding PODs within PODs and PODs overlapping other PODs are as follows: In the current implementation of TRIO, the logical sensors that collect metrics for a POD are only defined once. This means that for instance the CurrentStayInPeriod sensor is reset whenever the current POD ends and the next begin. Meaning that all metrics like the CurrentStayInPeriod, and the NumberOfTimesInZone are only available for one POD at a time, because they are only created once, and then they are reused for all PODs.

- **Design Alternative One** Change how logical sensor used by PODs are implemented, from only being defined once and then reused by all PODs, to making logical sensors for each POD. This way all PODs have their own data, which will only be reset when the same POD gets activated again. For verification and simulation purposes, the simulator have to be changed for it to be to visualize metrics for more than one POD at a time.

- **Design Alternative Two** Another alternative is to move away from the notion of PODs within PODs and overlapping PODs to another approach: We could change all risk functions, so that the risk function themselves keep track of metrics like number of times in zone and duration in zone. This means that we would create the logical sensors needed to calculate the durations for each risk function. This approach would diminish the need for POD as the risk function would have their own logical sensors which where keeping track of the metrics they need, e.g. duration in zone. This makes the risk functions function properly and is a pretty fast solution to the problem. However, this solution will make it more difficult to generate risk functions since we will need to make a logical sensor for each.

4.7.5 Suggested Implementation

Both of the discussed solutions will work and they both have advantages and disadvantages, e.g. if we choose *solution two* we will have to rewrite all present risk functions and all future risk functions need to be more complex. I.e. that risk functions will no longer use shared logical sensors depending on PODs, but rather implement their own logical sensors regarding PODs and contexts for each risk function as needed. This means that we rewrite TRIO so that risk functions are not dependent on logical sensors from PODs. By choosing solution two we are also decreasing the usefulness of the simulator. In the original implementation the simulator visualized the logical sensors being used by the risk functions. This way it was easy to verify the results of our risk functions because we had the data the risk function utilized available in the simulator.

The solution will work, and it would cost very little time to change, but it would also, as mentioned make all new risk functions more complex to create. When the risk functions are more complex, they will cost more time to create, so while this solution may save us time now, it can end up costing us more in the long run. It is also worth mentioning that if we have similar risk functions there are no shared logical sensors for PODs, this can lead to duplicate logical sensors regarding PODs in the risk functions. If we have too many duplicate logical sensors this might become a resource drain.

To conclude, solution two removes some of the functionality from the simulator and it makes the adding of new risk functions more complex while adding the new POD functionality. Although this solution is fast to implement and we get the improvement we want. We have decided that solution two is not a good choice.

Solution one will be a little more complex to implement. We will change the way logical sensors are created, meaning we create all logical sensors regarding PODs for each POD individually. For instance, we create a CurrentStayInPeriod logical sensor for each POD when the PODs are created. To ensure this does not lead to inferior performance we will utilize the fact that each POD is a context, so we make these logical sensors only active for the specific PODs, ensuring the least possible impact on performance. However, this solution can also impact performance if we have too many concurrent PODs (for each concurrent POD, we will have concurrent POD logical sensors). The creation of these new POD logical sensors will make it harder to see the connection between the logical sensors that are being used in the simulator. This solution does not compromise current functionality, thus we feel that solution one, although more complex to implement, is the best solution.

4.8 Risk functions in TRIO

In TRIO we have implemented a concept called risk functions. Risk functions are logical sensors which are designed to determine risk. We need to determine risk to determine when the system should alert health personnel. Here is a description of how contexts are connected to risk functions:

1. A risk function should determine risk based on behavior.
2. A person behaves differently in different contexts.
3. Context is any information that can be used to characterize the situation of an entity.
4. We use context variables to characterize the context the person is in.

   There can be:

   (a) A set of zones the person resides in at any point in time, and/or
   (b) A set of periods of day the person resides in at any point in time
5. Purpose of contexts: Due to 1. and 2. we need to enable risk functions only in particular contexts.

The goal of the TRIO system is to alert health personnel if the person that is being monitored by the system behaves in a way that increases risk. To do this the system needs to recognize normal behavior, and what is normal behavior depends on the context. Thus, we use contexts together with risk functions to create more trustworthy risk calculations.

A risk function also need to consider what is normal behavior for the monitored person before contacting health personnel. For instance, if a person normally sleeps for 7 hours, but then, one day sleeps for 12 hours that is considered to be abnormal for that person. In the future, TRIO should learn what normal behavior is for the person it is monitoring. However, in the current implementation, TRIO is not able to learn the behavior of a person, therefore we define thresholds for what is normal when defining a risk function.

4.9 Summary

In this chapter, we have focused on what context and context awareness is. We have learned that TRIO is a context aware system with its own definition of contexts, both regarding concept and implementation. We have discussed how contexts are used in different fields, and why context is an important part of an AAL system. Risk functions are the tools we use in TRIO to determine whether the monitored person may be in danger. Contexts are used together with risk functions to improve the accuracy of the risk assessment.

Contexts are a crucial part of TRIO, and our defined contexts in TRIO are POD and zone. These contexts are tools for making the risk assessment in TRIO more accurate, but they do have their limitations. We are limited by only being able to have one POD active at a time, and one zone active at a time. If we are able to create zones and POD independently of each other, TRIO may be able to provide a more accurate risk assessment in certain situations. However there is not enough time to improve them both. We therefore have to choose which improvement we should focus on. They are both important, but we have decided to focus on the POD improvement as this gives more value to the system. With only three PODs TRIO is very limited when analyzing behavioral patterns for instance, for the whole day or other specific time sections. Therefore the improvement of zone-context is out of scope for this thesis, the zone improvement is something that should be implemented in the future to further increase the accuracy of risk calculation in TRIO. More on the POD improvement in Chapter 5.

Chapter 5

Improved Period Of Day Context

The POD functionality of the current implementation is explained in Chapter 4.7 and illustrated in Figure 4.9. We have so far only used sequential PODs, meaning, when one POD ends another begins, and there are no overlapping PODs. The current implementation only supports one active POD at a time. This is working fine, but it limits the possibility of adding new PODs. As of now, if we want to add another POD we will have to edit the existing PODs accordingly.

To illustrate, let us say that we want to create a new POD for dinner time, let's call it "TRIODinner", which would start at 2 pm and last until 6 pm. As we can see in Figure 4.9, the new POD TRIODinner would be overlapping with the existing PODs TRIODay and TRIOEvening. To be able to add the new POD TRIODinner we would, in the current implementation as illustrated in the figure, have to add TRIODinner between TRIODay and TRIOEvening. To add the TRIODinner in between, we then have to edit TRIODay so that it ends at 2 pm and TRIOEvening so that it starts at 6 pm. After editing the existing PODs, we are able to add TRIODinner in between, but we have fundamentally altered the POD structure. Which means all risk functions using TRIODay and TRIOEvening would now have to be rewritten to continue working as intended. Because both TRIODay and TRIOEvening is now one hour shorter, risk functions using these PODs will not work in the TRIODinner POD without being rewritten. We do not want to edit all risk functions just because we have added another POD, we want to be able to add new PODs without having to think about existing PODs or risk functions. Consequently, we want to improve the current POD implementation so that it handles adding new PODs to the system independent of existing PODs.

To do this, we first look at all possible POD variations the system would need to handle:

- POD within another POD
- POD overlapping other PODs

In Figure 5.1, we have added new PODs to visualize all the variations the system now would need to handle. For instance, TRIOWholeDay which contains all other PODs, and we have added TRIODinner as mentioned above, which overlaps with TRIODay and TRIOEvening.

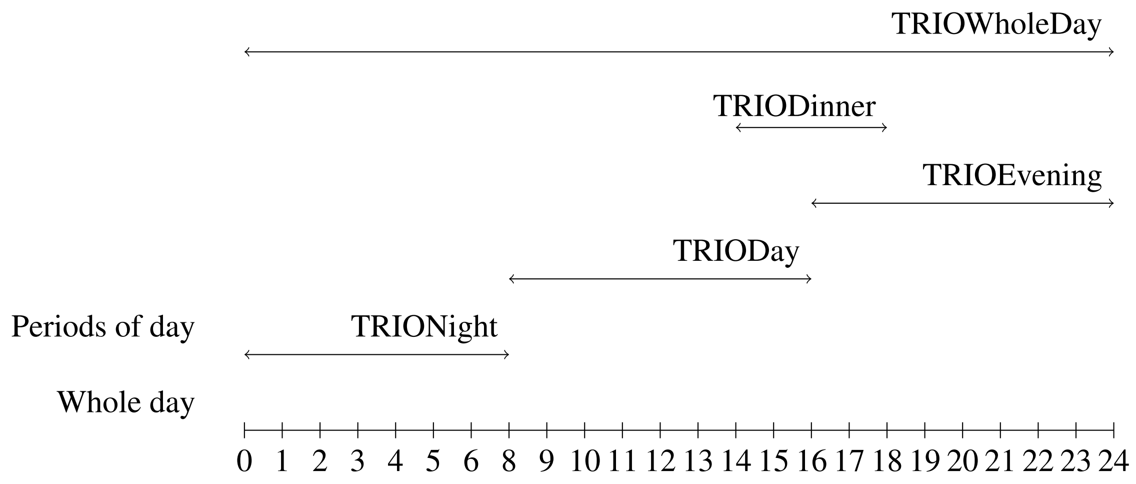


Figure 5.1: Illustration of periods of day, improved implementation

Now that we have a clear view of which POD variations we want the system to be able to handle, we evaluate what we can do to achieve this. TRIO is a complex system, so to get a better understanding of what the implementation of these new PODs will change, we decided to proceed with a trial and error approach. Meaning that we add the new PODs as illustrated in Figure 5.1, which represent all variations of PODs, and then investigate which parts of the system that is no longer working as intended.

After adding the PODs from Figure 5.1, to the current TRIO implementation, we systematically tested TRIO to see which parts of the system that was not working as intended anymore. We discovered that most parts of the system was working as intended, but not all.

Adding the new PODs changes the behavior of all logical sensors and risk functions that depend on PODs. To better understand the problem, we elaborate the description of one of the logical sensors.

The logical sensor we chose for explanatory reasons is the logical sensor CurrentStayInPeriod. To explain this more in detail, we have added a simplified implementation of the CurrentStayInPeriod sensor in Listing 5.1. CurrentStayInPeriod is a logical sensor which measures the duration a person has been in a zone for each POD. In Listing 5.1, we see that the duration of stay is called contextDuration, this sensor measures duration of stay for each zone for each context. This data is reset when there is a ContextBeforeEndPeriod event, which each POD context produces when it ends.

As mentioned, before we started to improve the POD implementation, all PODs were sequential, meaning when a new POD started the previous POD ended as we can see in Figure 4.9, which meant no overlapping and no PODs within other PODs. As said, the CurrentStayInPeriod duration gets reset whenever a POD is ending, which made sense given the sequential POD implementation. In Figure 5.1, we see TRIODay, and after that we have TRIOEvening and overlapping these two PODs we have TRIODinner. The duration of stay in the POD TRIOEvening will be reset when we leave the POD TRIODinner, causing the current duration of stay for the POD TRIOEvening to be incorrect after TRIODinner ends. By extension, this means that none of the logical sensors depending on the CurrentStayInPeriod is correct.

Listing 5.1: Example EPL: CurrentStayInPeriod

```
  insert into CurrentStayInPeriod
  select
    a.duration as contextDuration,
    current_timestamp() as ts,
    a.name as name,
    org.trio.Library.zoneStayWithinPeriod(a.ts, a.duration, b.
        enterTime, b.duration) as duration,
    b.id as id
    from pattern
      [
        every a=org.trio.ContextStartPeriod ->
        every b=DurationInZone
        and not org.trio.ContextBeforeEndPeriod
      ]
```

This also has repercussions for all other logical sensors that depends on this logical sensor. Like for instance, the risk function that measures if a person has been in the bedroom for too long a duration during the POD TRIOEvening. This risk function will no longer work as expected, because the duration of stay will be reset when the new POD TRIODinner ends.

5.1 Implementation

We started the implementation by adding new PODs which are overlapping, to get a better understanding of what would have to be changed in the current implementation. We did this because, although the TRIO software is complex, the simulator is a good visualization tool for discovering errors.

We used the simulators visualization to get a clear overview of which logical sensors had to change, by utilizing the fact that the logical sensor output are shown in the simulator in real time. By doing this, we realized that all logical sensors that where written for PODs would have to be rewritten. This is because all logical POD-sensors are limited by the fact that all PODs have been assumed to be sequential. For instance, all logical sensors needed for calculating the duration a person have been in a zone were built up around the fact that there was only one active POD at a time. As mentioned above, the data about how long a user had been in a zone was reset after each ending POD. We found that we need to create all POD specific sensors for each POD to be able to keep all data for all PODs. To do this we split up the logical sensors into categories, for instance, all logical sensors concerning duration in zone into one category, sensors regarding activity level into another category, and so on.

To make the code transparent for further expansions we moved the creation of these logical sensors to the creation of the POD context. So for each POD context we create, we also generate all the logical sensors concerning that POD context. Thus, making the creation of new logical sensors and PODs easy.

This solution might sound slow and heavy to run, since we are creating all logical sensors for each POD, but we are utilizing the fact that PODs are contexts and the logical sensors are not active unless the context they belong to are active. Meaning that they should draw no resources when the POD they belong to is not active. However, if we have more than one active POD at a time, the new solution will consume more resources. We are however further exploring the affects of this in Section 5.2.2.

5.2 Verification

An important part of a project like this is to verify that the changes made do not break the original implementation. In Figure 5.2, we can see the overview of the whole application without the risk function interface.

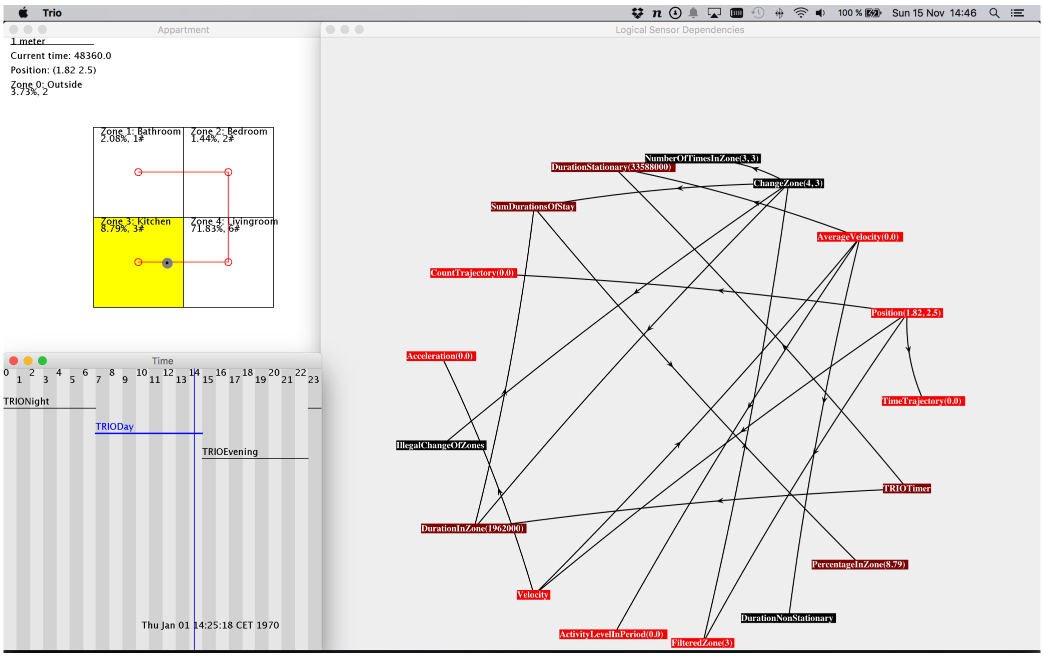


Figure 5.2: TRIO application overview original implementation

The way we verify the old implementation, is by using the simulator which continuously visualize recorded data as illustrated in the simulator part of the application, as we can see in Figure 5.3.

In Figure 5.4, we can see all the logical sensors in TRIO and how they are connected. For instance, the logical sensor called position, is visualized with a red box with white letters and the text Position (1.82, 2.5)[1]. These numbers represent

[1]To make it clearer, I have added a circle around the logical sensor we are discussing.

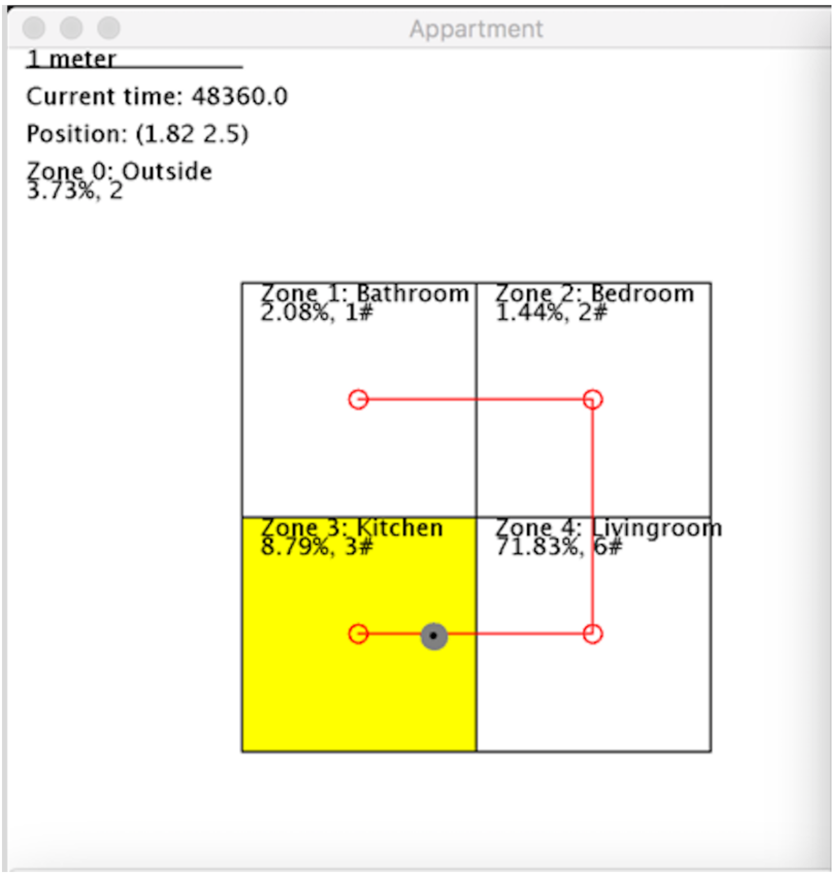


Figure 5.3: TRIO simulator original implementation

the coordinates 1,82 and 2,5 which are the coordinates of the black circle in the active zone[2] in Figure 5.3. This position updates and changes according to the input positions.

We can also see from Figure 5.4, that the position sensor supplies the FilteredZone sensor, which measure which zone the monitored person is residing in, which then is used by the logical sensor ChangeZone which measures when the monitored person moves to another zone.

In Figure 5.3, we can see which zone the monitored person currently resides in. We also get an overview of all of the zones which the radars cover, and inside these zones we can see a percentage and a digit followed by a #. The percentage illustrates how many percent of the current POD the person have been inside the zone, and the # visualize how many times the person have entered this zone during the current POD.

In Figure 5.5, we have visualized our PODs, at the top of this figure we have some numbers ranging from 0 - 23, these illustrate the hours in a day. Below these numbers, we have some horizontal lines with some names over them, these lines represent the specific PODs and from which times these are active. The blue

[2]The active zone has a yellow background.

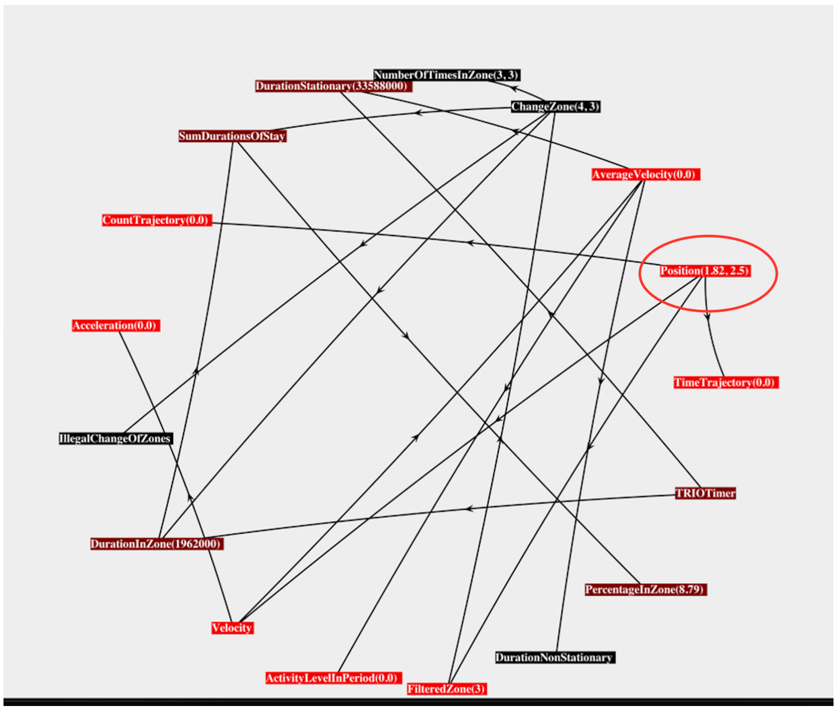


Figure 5.4: TRIO logical sensor dependencies original implementation

vertical line illustrates the current time of day.

It is worth noting that the time factor has been modified for testing purposes. The simulator uses something we call ’time compression’, which means that the time in the simulator is going much faster than real time. More specifically, the simulator has a time compression factor of 600, meaning that the time in the simulator passes 600 times faster than in reality. This time factor can be modified, and is there to enable us to test the system faster.

To be able to verify that our expansion of the system did not compromise it, we have to test it with the simulator and using risk functions. The simulator, as it were originally, did not handle more than one active POD at a time. In the original implementation we had only one visualized value of both number of times in zone and percentage in zone as seen in Figure 5.3. We quickly realized that to be able to test and verify the application, we first had to improve the simulator so that it showed the data for both number of times in zone and percentage in a zone for each POD in each zone. This way we could begin to verify that our improvement worked by using the simulator to see the data the logical sensors are producing. Then, test the risk functions using the new POD logical sensors to see that they are still working as expected.

Consequently, we modified the simulator to visualize the duration in zone and number of times in zone counters, in Figure 5.6a we see an overview of the application after the improvement. As we can see, there are now more logical sensors illustrated in the right part of the simulator, which illustrates what logical

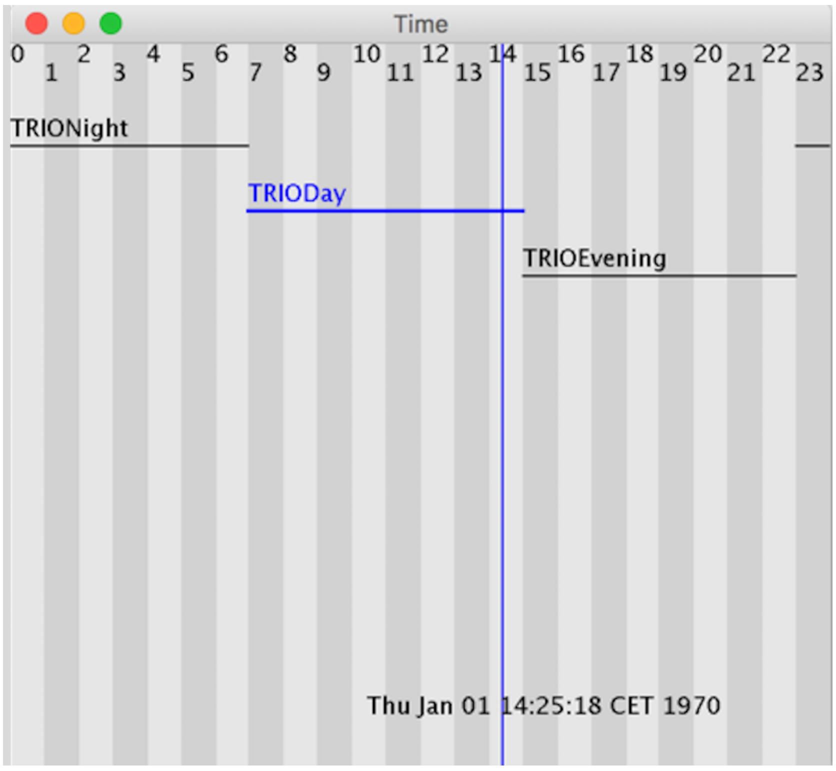


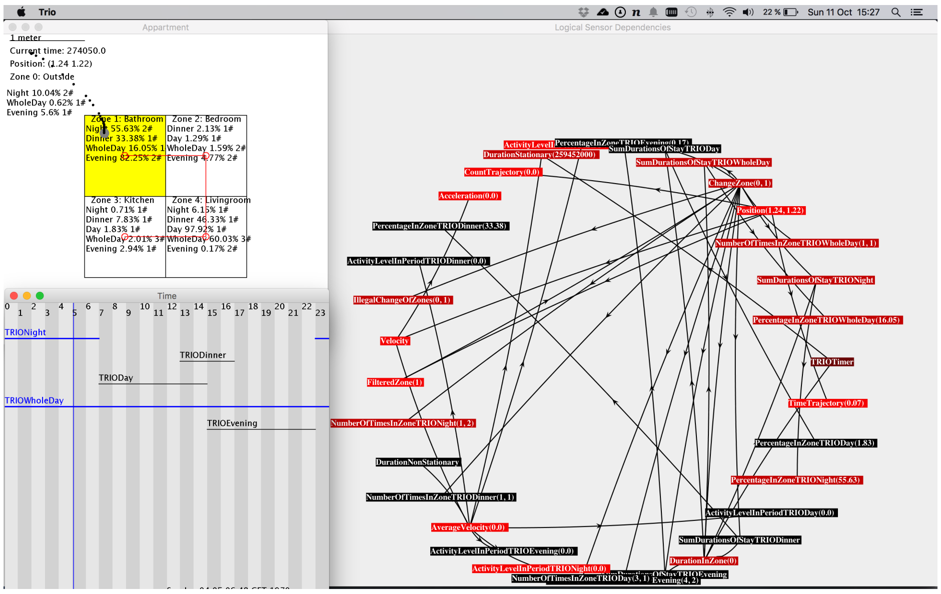
Figure 5.5: TRIO POD overview original implementation

sensors are in TRIO. We have now implemented all POD specific sensors for each POD as discussed in Chapter 5.1, therefore there are now more logical sensors than before.

In Figure 5.6b, we can see that there is now more counters for percentage and number of times in zone for each zone. There is now a percentage counter and a number of times in zone counter for each of the PODs in each zone.

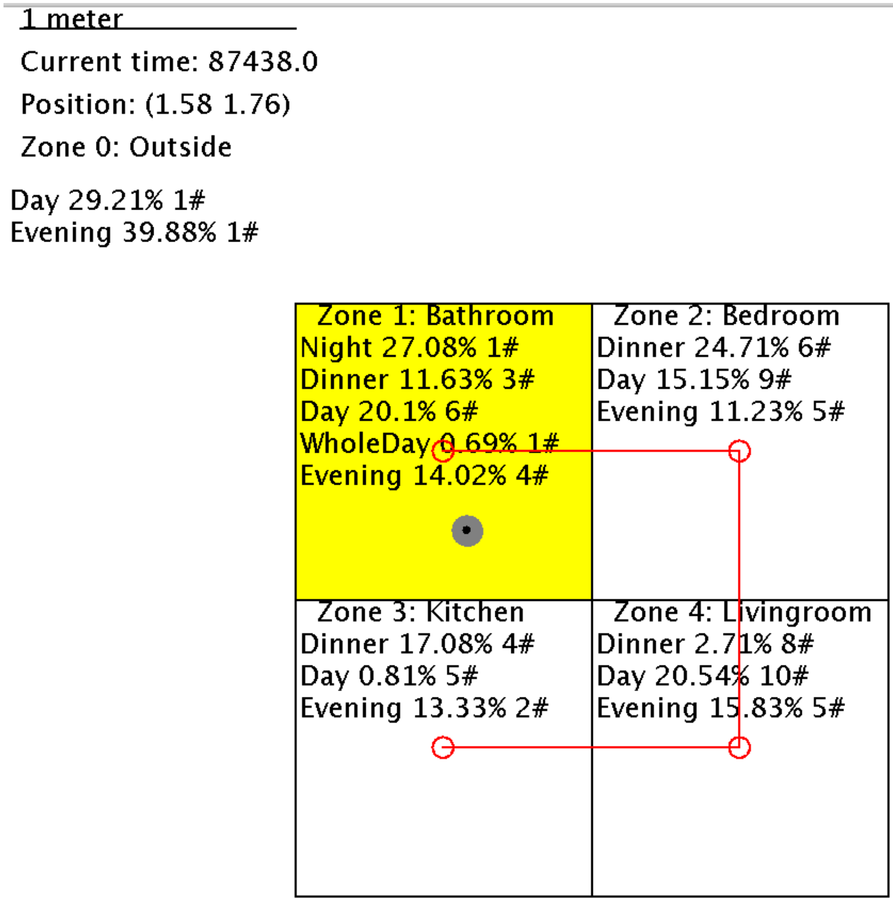
The visualization of how all the sensors are connected in Figure 5.7, is now a bit more complicated and difficult to follow. The reason for this, is that there are so many new logical sensors that the visualization gets over-flooded. There are different ways this can be handled. E.g., we could only visualize the active logical sensors. The active logical sensors are the logical sensors that is in use at the moment by either a zone or a POD, and all the inactive logical sensors could be filtered out. This would mean that the visualization window would have to be graphically re-rendered each time a zone or a POD becomes active or inactive.

Another possible solution is to apply layered categories, meaning that all logical sensors belonging to a category would be in its own visual container. The categories could for instance be PODs or zones, or some other division. Maybe the way of visualizing the connection between logical sensors should all just be rethought and done in another way. This was however, too big of an assignment to include in this master thesis, so any redesign of the visualization of the logical sensors is considered to be out of scope for this master thesis. It may even be an idea for a new thesis where the goal is to discover and implement a good way to



Trio
Appartment
Logical Sensor Dependencies
Time
1 meter
Current time: 274050.0
Position: (1.24 1.22)
Zone 0: Outside
Night 10.04% 2#
WholeDay 0.62% 1#
Evening 5.6% 1#
Zone 1: Bathroom
Zone 2: Bedroom
Zone 3: Kitchen
Zone 4: Livingroom
TRIONight
TRIODinner
TRIODay
TRIOWholeDay
TRIOEvening
CountTrajectory(0.0)
Acceleration(0.0)
PercentageInZoneTRIODinner(33.38)
ActivityLevelInPeriodTRIODinner(0.0)
IllegalChangeOfZones(0, 1)
Velocity
FilteredZone(1)
NumberOfTimesInZoneTRIONight(1, 2)
DurationNonStationary
NumberOfTimesInZoneTRIODinner(1, 1)
AverageVelocity(0.0)
ActivityLevelInPeriodTRIOEvening(0.0)
ActivityLevelInPeriodTRIONight(0.0)
ChangeZone(0, 1)
Position(1.24, 1.22)
NumberOfTimesInZoneTRIOWholeDay(1, 1)
SumDurationsOfStayTRIONight
PercentageInZoneTRIOWholeDay(16.05)
TRIOTimer
TimeTrajectory(0.07)
PercentageInZoneTRIODay(1.83)
PercentageInZoneTRIONight(55.63)
ActivityLevelInPeriodTRIODay(0.0)
SumDurationsOfStayTRIODinner
DurationInZone(0)
SumDurationsOfStayTRIOWholeDay
DurationStationary(259452000)
SumDurationsOfStayTRIODay

(a) TRIO application overview new implementation



1 meter

Current time: 87438.0

Position: (1.58 1.76)

Zone 0: Outside

Day 29.21% 1#
Evening 39.88% 1#

Zone 1: Bathroom	Zone 2: Bedroom
Night 27.08% 1# Dinner 11.63% 3# Day 20.1% 6# WholeDay 0.69% 1# Evening 14.02% 4#	Dinner 24.71% 6# Day 15.15% 9# Evening 11.23% 5#
Zone 3: Kitchen	**Zone 4: Livingroom**
Dinner 17.08% 4# Day 0.81% 5# Evening 13.33% 2#	Dinner 2.71% 8# Day 20.54% 10# Evening 15.83% 5#

(b) TRIO simulator new implementation

Figure 5.6: TRIO overview after implementation

visualize the connection between all the logical sensors in TRIO, as it is fair to assume that the number of logical sensors in TRIO will only increase with time.

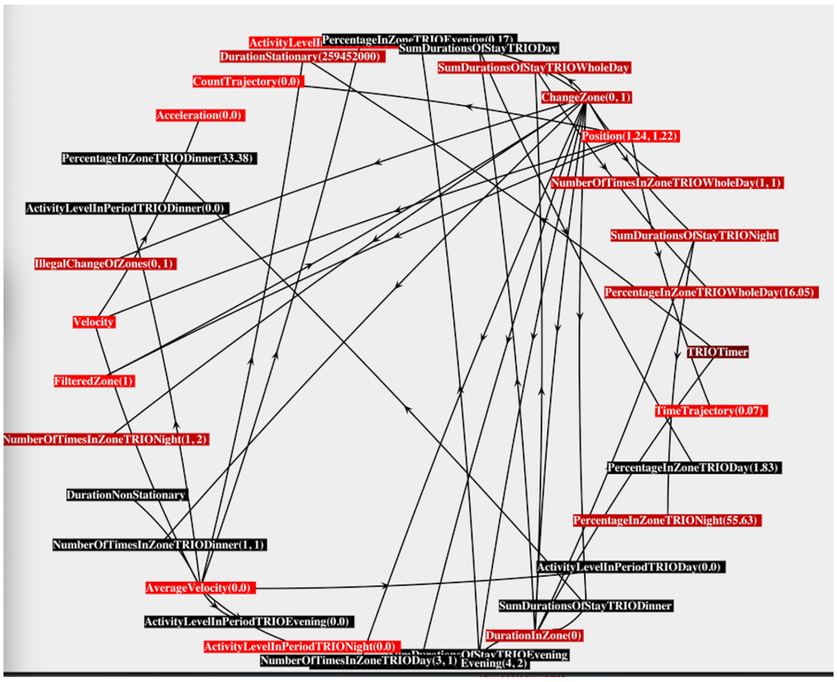

ActivityLevelI
PercentageInZoneTRIOEvening(0.17)
SumDurationsOfStayTRIODay
DurationStationary(259452000)
SumDurationsOfStayTRIOWholeDay
CountTrajectory(0.0)
ChangeZone(0, 1)
Acceleration(0.0)
Position(1.24, 1.22)
PercentageInZoneTRIODinner(33.38)
NumberOfTimesInZoneTRIOWholeDay(1, 1)
ActivityLevelInPeriodTRIODinner(0.0)
SumDurationsOfStayTRIONight
IllegalChangeOfZones(0, 1)
PercentageInZoneTRIOWholeDay(16.05)
Velocity
TRIOTimer
FilteredZone(1)
TimeTrajectory(0.07)
NumberOfTimesInZoneTRIONight(1, 2)
PercentageInZoneTRIODay(1.83)
DurationNonStationary
PercentageInZoneTRIONight(55.63)
NumberOfTimesInZoneTRIODinner(1, 1)
ActivityLevelInPeriodTRIODay(0.0)
AverageVelocity(0.0)
SumDurationsOfStayTRIODinner
ActivityLevelInPeriodTRIOEvening(0.0)
DurationInZone(0)
ActivityLevelInPeriodTRIONight(0.0)
OfStayTRIOEvening
NumberOfTimesInZoneTRIODay(3, 1)
Evening(4, 2)

Figure 5.7: TRIO logical sensors dependencies new implementation

In the POD overview in Figure 5.8 we can see that we now have more PODs, like the new POD TRIOWholeDay, which lasts for 24 hours, and TRIODinner, which overlaps TRIODay and TRIOEvening. We chose these PODs to make sure that TRIO handles all kinds of new PODs as these PODs covers all edge cases, such as nested and overlapping PODs.

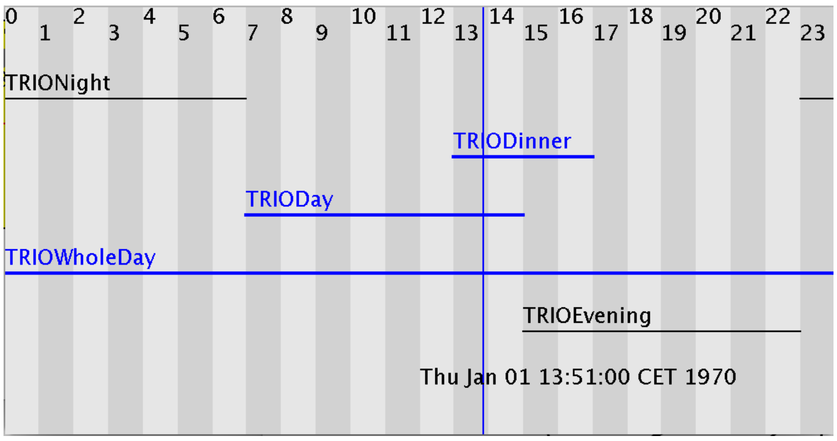


Figure 5.8: TRIO PODs new implementation

5.2.1 Verification scenario

After improving the simulator so that it handles the new POD implementation, we need to verify that the risk functions are still working as intended. This is important because the core functionality of the TRIO system is to calculate risk.

We do this by testing one of the old risk functions, first on the old implementation, then on the new implementation, to verify that our implementation is working like the old implementation. The risk function we decided to test is the function that sounds an alarm if the person is in the bedroom for too long during the POD TRIODay.

Test scenario original implementation First we enable this risk function, then we wait until the TRIO system is in the POD TRIODay. We can see that TRIO is in the POD TRIODay when the vertical line in Figure 5.5, is within the POD TRIODay that is illustrated by a horizontal line under the name TRIODay. Then, we move into the bedroom, by moving our cursor[3] from Zone 3: Kitchen zone to Zone 2: Bedroom, which is the top right zone as we can see in Figure 5.3. After some time[4] the risk function is in alert mode as we can see in Figure 5.9. When a risk function is in alert mode, it is illustrated by a red background color of the explanation text in the top of the risk function, as seen in Figure 5.9, it is also producing an alarm sound.

Test scenario new implementation We now have a base line for the intended behavior, we then continue with replicating the scenario on our new implementation. We activate the risk function, then we move into the bedroom zone during the POD TRIODay. Then, after some time, we can see that the risk function is working as expected and producing an alarm in the same way as it did for the old implementation. The risk function from the new implementation is illustrated in Figure 5.10.

The risk function from the new implementation looks almost the same, but there is a small difference to the old implementation: we used the CurrentStayInPeriod sensor for gathering the duration we were in the bedroom zone. In the new implementation, we no longer have this logical sensor, because all PODs have their own CurrentStayIn logical sensor. Therefore, we are using the CurrentStayInTRIODay instead, which leads to the same result.

This verification scenario verifies that our new implementation has not broken the functionality from the original implementation.

Test scenario new functionality on the new implementation Now we have to do another experiment using one of the new PODs to make sure that our improvement adds value to the system and is working as designed.

[3]We are now using the mouse as input data for the person's location

[4]It is hard to tell exactly how long we have been waiting because of the time compression factor

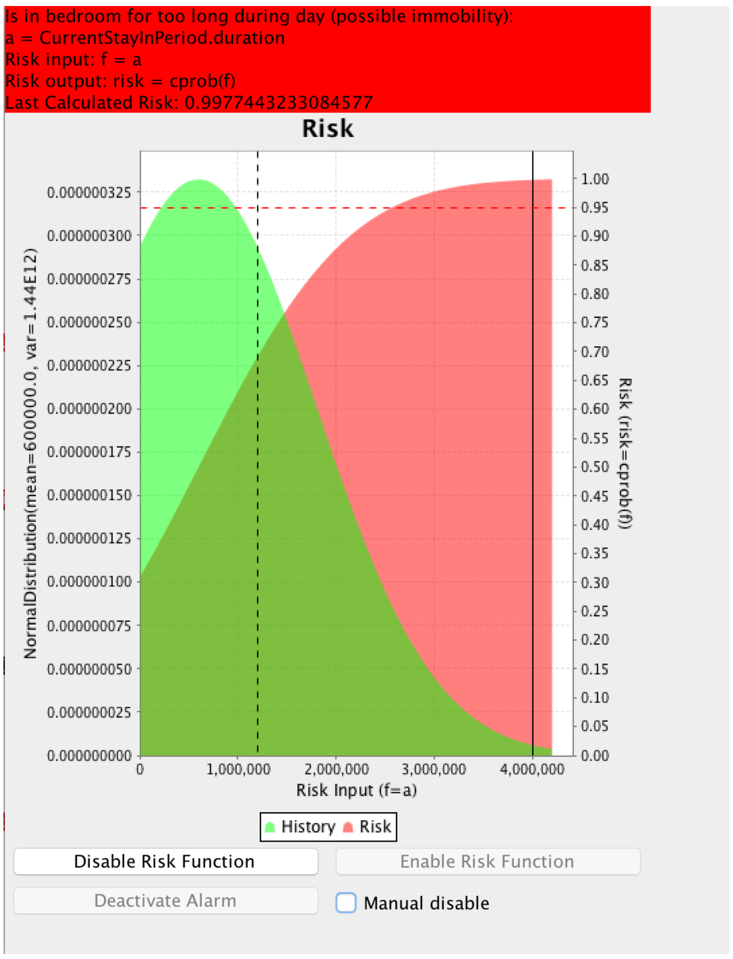

Is in bedroom for too long during day (possible immobility):
a = CurrentStayInPeriod.duration
Risk input: f = a
Risk output: risk = cprob(f)
Last Calculated Risk: 0.9977443233084577

Figure 5.9: TRIO risk function too long in bedroom during TRIODay warning old implementation

We do this by testing another risk function using one of the new PODs. We decided to use a risk function that measures if the monitored person is residing in the bathroom for too long during TRIODinner. TRIODinner is one of the new PODs we have added, this POD is within TRIOWholeDay and it overlaps with the PODs TRIODay and TRIONight as we can see in Figure 5.8. First we activate the risk function, then we move the cursor into Zone 1: Bathroom during the POD TRIODinner as illustrated in Figure 5.6b. After some time, the risk function sounds an alarm as we can see in Figure 5.11. This indicates that the new PODs are also working as expected with the risk function.

5.2.2 Performance, Scalability and expandability of the POD Implementation

When we are implementing new functionality into a system, it is important that this functionality does not harm the system we are trying to improve. A big concern with our implemented extension, is that we do not want to add

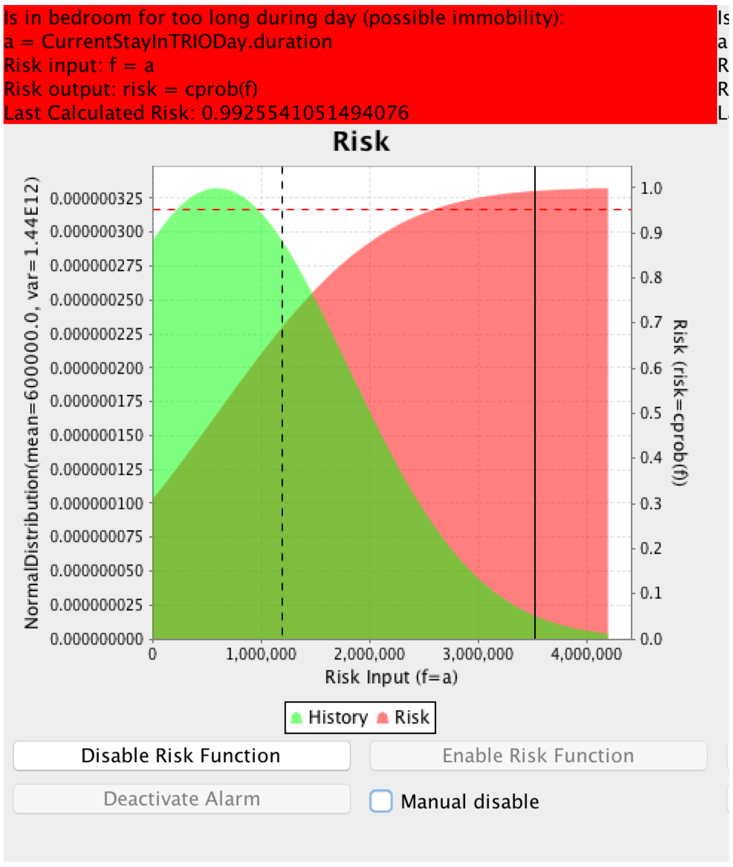

Is in bedroom for too long during day (possible immobility):
a = CurrentStayInTRIODay.duration
Risk input: f = a
Risk output: risk = cprob(f)
Last Calculated Risk: 0.9925541051494076

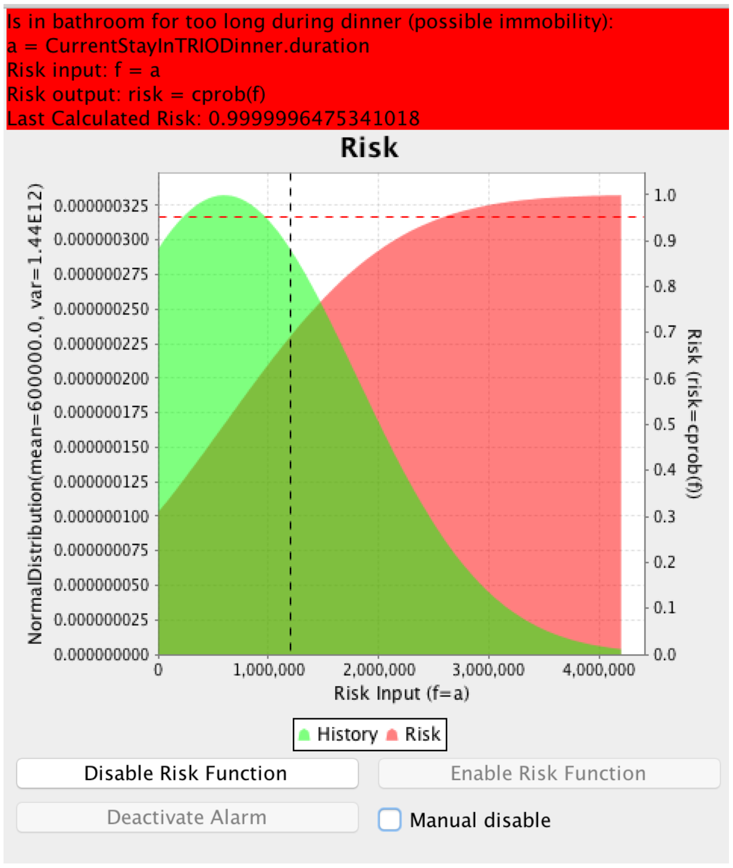
Figure 5.10: TRIO risk function too long in bedroom during TRIODay warning new implementation

functionality that may hinder the intended use of TRIO.

Our main goal for exploring the scalability and expandability repercussions for our extension to TRIO, is to explore if our extension introduces performance problems or not. An extension, is only considered to be an improvement if it does not apply problems regarding the intended use of the system. A possible issue with our extension is performance. Our extension lets us have more than one active POD at a time. In our extension, each POD has a set of logical sensors, meaning that for each POD we add to TRIO, we add a new set of logical sensors. Furthermore, each POD we add within the same period of time will increase the number of simultaneously active logical sensors in TRIO.

An important part of a system extension, is to evaluate the impact the extension has on the system in form of:

- Performance, how does our extension affect the performance of the system, exploring if our implementation introduce any performance issues.
- Scalability, a system such as TRIO is supposed to be used in a bigger scale in the future with many different logical sensors, we have to consider if our extension limits these possibilities.



Is in bathroom for too long during dinner (possible immobility):
a = CurrentStayInTRIODinner.duration
Risk input: f = a
Risk output: risk = cprob(f)
Last Calculated Risk: 0.9999996475341018

Figure 5.11: TRIO risk function too long in bathroom during TRIODinner warning new implementation

- Further development, we explore how our expansion affects the further development of TRIO.

This evaluation is important, because it gives us the scientific data needed to conclude if our extension of TRIO is an improvement or not. Firstly, we explore how our extension impacts the performance of TRIO. Then, we discuss how our extension impacts the scalability and further development of TRIO.

To get an overview of which parts of TRIO that can be affected by this extension, we view each part of TRIO separately, using the following simplified explanation of TRIO. TRIO can be divided into three parts as illustrated in Figure 5.12; the simulator, the TRIO system backend and Esper. The TRIO backend creates the logical sensors for Esper, and the results from Esper is visualized in the simulator by the TRIO backend. Our extension has not made any changes to Esper, however we have made some changes to both TRIO backend and the simulator, therefore will these parts be explored further.

As mentioned earlier in this section, our extension of TRIO may lead to an increased number of simultaneously active logical sensors. Each POD we add to the system generates logical sensors which are interpreted as events in Esper. When the number of simultaneously active logical sensors increases, the events

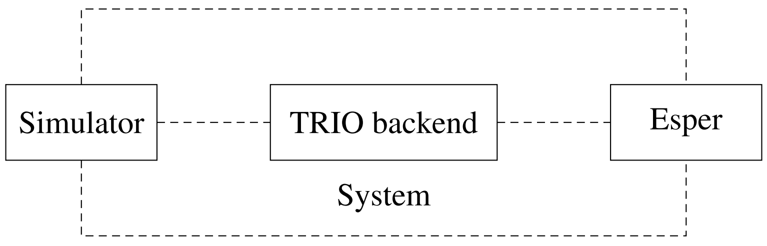


Figure 5.12: System definition

per second increases as well, this may lead to performance issues. We know from the Esper FAQ site, that the number of simulations events per second Esper can handle single threaded, is likely between 10 K and 200 K. This is a highly flexible number, therefore we explore how many events per second TRIO is able to handle.

We perform experiments to measure the performance impact of our extension. To produce load we are adding simultaneous PODs, for each POD we add, logical sensors are generated which leads to more events that TRIO must handle. The idea is to stress TRIO to the point where it breaks, we therefore add PODs until TRIO is no longer functional.

Limitations The experiments we are conducting have little accuracy, as millisecond accuracy has little impact when determining if our extension is an improvement or not. Therefore, temperature and other outside influences will be disregarded. However, it is worth mentioning that the system will most likely be running in room temperature, as this is a system intended for use in a persons home.

All experiments are being executed from a MacBook Pro 11,3 with the following specifications; Intel Core i7 processor speed: 2,6 GHz with 4 cores and 16 GB of memory. System version: OS X 10.11.6 (15G31). The results may be different on a machine with different hardware, but to do advanced performance analysis is out of the scope for this thesis.

The number of zones in TRIO is something that can be changed in the future, however this will not be changed in our tests. As additional zones will not have much impact on the performance of TRIO.

To test the performance of the system, we will change the number of simultaneously active PODs to increase or decrease the workload on the TRIO system. The system extension is implemented in a way that should only make it possible for PODs to be a performance issue if there are multiple PODs within the same time frame. This is due to the fact that PODs are context dependent, which means that the logical sensors belonging to each POD is set to only be active when the context of that POD is active. For instance, the logical sensors regarding POD *TRIODay* is only active when the POD is active, i.e. from 8 a.m to 16 pm. This means that adding POD's spread out over time should not have an effect on system performance.

Performance evaluation

We start by explaining what performance evaluation is and why it is important, then we will move on to the performance evaluation itself.

Firstly, we must think about the changes done in our implementation that may affect the performance of TRIO. As mentioned above, our implementation enables TRIO to have more than one active POD at a time. For each active POD, we have duplicated all the logical sensors that create data for each POD. However, some of the logical sensors in TRIO are common or independent for all PODs, more specifically there are 20 logical sensors active in TRIO even if we remove all PODs. For each concurrent POD, we have an additional 23 logical sensors active. With the expected use of TRIO at this time, we expect no need for more than three concurrently active PODs, as we see no use for more. Three concurrently active PODs, leads to a total of $20 + (3 * 23) = 89$ concurrently active logical sensors. The number of concurrently active PODs can however increase in the future, but we do not expect this number to increase drastically. Therefore, it is important that TRIO can handle a minimum of three concurrent PODs. However, to support possible future needs, we prefer that TRIO can handle at least six concurrent PODs.

To evaluate how TRIO performs after our extension, we test how a various number of simultaneous PODs affect ther performance of TRIO. By performance, we mean how the system acts and behaves under load, specifically we want to know if TRIO works as it is supposed to under heavy load. Therefore, we need a baseline for what we consider to be normal behavior of TRIO.

We define our baseline by performing the following actions; We activate the risk function that checks if the monitored person has been in the bathroom for more than 20 minutes during the POD *TRIODay*. Then, we move the cursor (which serves as the monitored person's position), into the zone *bedroom* during the POD *TRIODay*. We roughly measure the elapsed time from when we moved the cursor into the *bedroom* zone and until the risk function sounds an alarm. This time estimate is our baseline for correct behavior of the aforementioned risk function in TRIO. Because TRIO's ultimate goal is to calculate risk, we find it natural to use risk function as a part of our performance experiment.

This baseline experiment is done with only the three original PODs active in TRIO; *TRIODay*, *TRIOEvening* and *TRIONight*. To measure the performance of TRIO, we redo the experiment with an increasing number of PODs within the same timespan as *TRIODay*.

It is worth mentioning that during the implementation of our extension, we did some preliminary tests to verify basic functionality of the system. This was done to check that there were no obvious faults with TRIO's behavior as a consequence of our implementation. We did not notice any performance issues in this phase.

We consider an experiment to be successful if the following two criteria are fulfilled.

1. The risk function is working properly, i.e. we are able to activate the risk function, and the risk function sounds an alarm within the appropriate time.

2. The responsiveness of TRIO is not too bad, i.e. the responsiveness does not make TRIO hard to use.

If these criteria are not fulfilled we consider the experiment to be unsuccessful. We test TRIO with an increasing number of simultaneous PODs until the test fail, thus determining the performance limits of TRIO on our test computer. This simplified test, gives us an impression of the performance of TRIO after the implementation of our extension. When we know the performance limitations of TRIO, we can discuss how and if this limits the expandability of TRIO.

Experiments

The measurement metrics for the experiments we are conducting is the number of concurrent logical sensors we have active. We produce concurrent logical sensors by adding PODs within the same time frame as the POD *TRIODay*. For all experiments, overall system performance is evaluated, as it is important that the system has free processing time to e.g. send a message to health personnel.

It is worth mentioning that the test machine is a quad core machine that supports hyper threading, which means that 100% CPU utilization is actually 12% of our machines entire CPU capacity. A summary description of all experiments is given in Table 5.1.

Experiment 1 In our first experiment, we test how much load the original implementation produces. Which means that the only PODs running are *TRIODay*, *TRIOEvening* and *TRIONight*.

TRIO responsiveness is smooth and fast, the risk function is activated almost immediately and sounds an alarm after about five seconds. Overall system response is good. We can see a CPU load of about 50%, and a average RAM usage of 280MB.

Experiment 2 In this experiment we want to test our extension against the original implementation to see if our extension is using more resources. Therefore, we have the same PODs as in the previous experiment.

Both system performance and TRIO acts and feels the same as in the previous experiment. The risk function is activated as fast as in the previous experiment, and sounds an alarm after about five seconds. We can see a CPU usage of about 50%, and RAM usage is about 280MB.

Experiment 3 In this experiment we wish to verify if TRIO can handle as many concurrent PODs as needed. We believe that TRIO will need to be able to handle at least six concurrent PODs. However to be sure, we are adding ten concurrent logical sensors within the same timeframe. More specifically; in the same timeframe as *TRIODay*.

The overall system responsiveness is unaffected, but the responsiveness in TRIO is a bit slower when the program starts. However, after about five seconds

TRIO recovers and the responsiveness improves. The risk function now uses about a half second to activate, but once it is activated it sounds an alarm after about five seconds. The CPU usage has increased to about 70%, this is an 20% increase from the previous example. Therefore, it is clear that running as many as ten concurrent PODs consumes a lot of CPU runtime on our test machine. The RAM usage is about 280MB, i.e. the same as in previous experiments. This experiment gives an indication that our implementation has not introduced any big performance issues, as TRIO is able to handle more concurrent PODs than we believe it will need to. If we are in need of even more concurrent PODs, the simulator would have to be rewritten. With ten concurrent PODs as well as the PODs *TRIOEvening* and *TRIONight*, the simulator overview is over-flooded with information. Information about current stay in duration and number of times in zone, for all 12 PODs. Which means that we need to make modifications to the simulator, before having to improve performance.

Experiment 4 In this experiment we wish to see how TRIO behaves with 100 concurrently active PODs. We wish to explore if we are still able to activate a risk function and if it works as it is supposed to, or if this amount of concurrent PODs will cause TRIO to crash.

The overall system responsiveness is still unaffected, but the responsiveness of TRIO is now noticeably slower. TRIO uses a couple of seconds to start, and when we move the cursor around in the simulator view, the movements are choppy and slow. It takes a couple of seconds to activate the risk function, and it sounds an alarm after about five seconds. All though the simulator is slow, and TRIO takes longer to start, it still works. However, the simulator is completely over-flooded with information. Data recorded for duration and number of times in a zone is now not visible for most of the PODs. The CPU usage has increased from the previous experiment, it peaks to 150% CPU when TRIO is starting, but after some seconds it goes down to about 100% CPU usage. The RAM usage has increased as well, TRIO is now consuming about 450MB of RAM.

Experiment 5 In the previous example, the simulator was slow, but the TRIO system did work as intended. In this experiment, we wish to test if 500 concurrent PODs will cause TRIO to malfunction.

The overall system performance is still unaffected, but the responsiveness of TRIO is terrible. TRIO uses about one minute to start, and it takes about 30 seconds to activate the risk function. The simulator is very choppy and slow to react, and TRIO freezes for some seconds when we are moving the curser around. The risk function still gets activated and it does sounds an alarm after about five seconds, but TRIO is too slow and unstable to be usable at this point. The CPU usage peaks at about 250% at the startup of TRIO, but after some seconds it goes down to about 100%. This indicate that most of the load in TRIO is generated in only one thread. This may be improved if the need for such an overwhelming number of PODs are needed. The RAM usage has increased quite a bit from the previous examples, TRIO is now consuming about 1.5GB of RAM.

Table 5.1: Performance evaluation experiments summary

Experiment summary

Experiments	**Results**	**CPU**	**RAM**
Experiment 1, original version of TRIO, active PODs *TRIODay*, *TRIOEvening* and *TRIONight*.	TRIO startup time: Instantly. Simulator behavior: Fast and smooth. Risk function activation time: Immediately. Alarm sounding after: Approximately 5 seconds.	About 50%	About 280MB
Experiment 2, extended version of TRIO, active PODs *TRIODay*, *TRIOEvening* and *TRIONight*.	TRIO startup time: Instantly. Simulator behavior: Fast and smooth. Risk function activation time: Immediately. Alarm sounding after: Approximately 5 seconds.	About 50%	About 280MB
Experiment 3, extended version of TRIO, active PODs *TRIODay1-10*, *TRIOEvening* and *TRIONight*.	TRIO startup time: Few seconds. Simulator behavior: A bit slower. Risk function activation time: Approximately half a second. Alarm sounding after: Approximately 5 seconds.	About 70%	About 280MB
Experiment 4, extended version of TRIO, active PODs *TRIODay1-100*, *TRIOEvening* and *TRIONight*.	TRIO startup time: About eight seconds. Simulator behavior: Noticeably slower. Risk function activation time: Approximately a couple seconds. Alarm sounding after: Approximately 5 seconds.	About 100%	About 450MB
Experiment 5, extended version of TRIO, active PODs *TRIODay1-500*, *TRIOEvening* and *TRIONight*.	TRIO startup time: About a minute. Simulator behavior: noticeably slower. Risk function activation time: About 30 seconds. Alarm sounding after: Approximately 5 seconds.	About 100%	About 1.5GB

Experiment Summary With an extension such as ours, which changes some of the fundamentals of a system, there is always a risk that it will have an impact on scalability. Our implementation changes TRIO from having a set of logical sensors shared between all PODs, to having them replicated for each POD.

There are more ways such an extension may affect scalability. If our extension impacts performance in a way that limits the use of the system, this directly impacts the scalability of the system. For instance, if the system responsiveness is very slow, this will limit the scalability of the system. If we are not able to have as many simultaneous logical sensors as we need, because of performance issues, we have limited the scalability of the system.

With these simple experiments, we have illustrated that our extension of TRIO regarding POD is performing quite well. The number of PODs the system would be able to handle to pass this test, was at least six. To be sure, we tested with ten concurrently active PODs. Although TRIO consumes more resources, it is working as it is supposed to. The TRIO system did even perform its key objective, which is to calculate risk, with 100 and 500 concurrent PODs. Although the overall performance of TRIO was slower, we still consider the test with 100 concurrently active PODs a success performance wise. With 500 concurrently active PODs, the TRIO systems overall performance was too bad to be called a success. To use TRIO with more than 12-15 PODs the simulator must be changed, as the simulator view gets flooded with information, as the number of PODs increases. If an higher amount than 100 concurrent PODs should be supported, TRIO should be rewritten to run on more threads to utilize more than one core of the machine it is running on. Another possibility is to use a machine with a stronger CPU than our test machine. We therefore believe there is no performance issues with our extension of TRIO.

We also have to explore if our solution has decreased the expandability of the TRIO system. Expandability is the possibility for further development and expansion of TRIO. To determine if our extension has decreased the expandability of TRIO, we consider which parts of the TRIO system our extension as altered. We have altered the simulator in TRIO, and logical sensors regarding POD.

The simulator change is relatively small, it lets TRIO visualize both current stay in duration in zone, and number of times in zone which is produced for each POD. This change does not interfere with any other parts of TRIO and is therefore considered isolated and irrelevant for further development of TRIO.

The POD change is however bigger, we have altered the way logical sensors that is relevant for PODs gets added to TRIO, because of this we must ask ourselves the following: Is there any logical sensors that we were able to add to TRIO before the expansion, but not after?

The implementation of the POD extension does not limit the possibilities for adding any sensors as we could have prior to the extension. This is because we have not altered to fundamentals for adding logical sensors to the TRIO system, we have altered the way TRIO creates logical sensors that are meant for use with a POD. We create the same logical sensors as before, we just create them for each POD.

We therefore view our extension of TRIO as an improvement.

5.3 Summary

In this chapter we have learned about the context period of day, how PODs work, and the current weaknesses of the implementation. We have discussed and analyzed the current implementation and how it can be improved, by exploring the weaknesses of the current implementation and trying to find a solution to the how to improve this weakness. Then, we explored several possible implementation alternatives and implemented one of them. After the implementation, we did some experiments to check if TRIO is working as expected after the implementation of this extension. These verification experiments did not give any indication that our implementation has broken the fundamental functions of TRIO. We have explored whether this implementation is an improvement on TRIO or not. This implementation gives us the possibility to add PODs to TRIO independent of existing PODs. Which lets us add PODs, like e.g. TRIOWholeDay which lets TRIO detect risk for behavior during the whole day, or other specific times with other PODs which leads to more accurate risk assessment.

In the next chapter, we conclude this thesis, wrapping up what we have done and discovered during this thesis, as well as future work and a critical assessment of the thesis.

Chapter 6

Conclusion

In this masters thesis, we have explored the usability of Novelda's safety radar for use in the medical care domain. We have explored the ambient assisted living system, TRIO, where we discussed and improved some of the functionality. First we present our contribution, then we move on to a critical assessment of the thesis. Then, we present topics and ideas for future work. The goal of this master thesis was to explore the usability of Novelda's radar in the medical care area, as well as exploring and improving the ambient assisted living system TRIO.

6.1 Contributions

Our main contributions for these goals are: Explore the TRIO system and Novelda's safety radar and how these can be used in a medical care environment. We have looked in to existing work, to understand what an ambient intelligent system is. Which criteria such a system will have to fulfill, and why TRIO is considered to be an ambient assisted intelligent system. We have explored what we can improve in TRIO to make it a better ambient intelligent living system. During this process we have explained core parts of TRIO in detail, focusing on the concept of contexts and risk functions. The main goal of an ambient intelligent living system is to determine risk, therefore we want to make improvements on TRIO so that it can better calculate risk.

Our *first* contribution is to explore and test the Novelda safety radar. We have tested and explored the Novelda radar in a laboratory environment at UiO. Our testing concluded that the triangulation scripts supplied by Novelda for tracking a person, was at this time not good enough to be used in the development of new logical sensors and risk functions in TRIO. Our test results and findings have been brought to the attention of Novelda, so that they are able to improve their triangulation scripts.

Our *second* contribution has been to discover a weakness in the current implementation of PODs in the TRIO system. We have discussed and explained this weakness, making it clear that the current implementation leads to inferior risk assessment accuracy. We have designed and implemented a solution for this weakness which improves the current implementation. This improvement has also been verified and tested with verification scenarios, which help determine that our implementation has not removed any functionality from the system. Then, we verified that the new functionality we have implemented is working as intended giving the TRIO system more accurate risk assessment.

Our *third* contribution has been to discover a weakness of the current implementation of zones in the TRIO system. We have discussed and explained the nature and consequence of this weakness and why this weakness leads to inferior risk assessment. We have explored different solutions to this problem and decided on an approach. We have made a design for the approach we chose which should further improve the risk assessment of TRIO.

6.2 Critical assessment

If we could start the process over, we would have shifted our focus from the original assignment sooner when it became clear that our original assignment was not doable. Our original assignment was to use Novelda's safety radar together with the TRIO system to create new logical sensors. As it were, we used a lot of time looking into creating new logical sensors and using the radar for input data. We did many experiments with the radar to deduce if we could use the radar as an input source for our positioning data, which proved not to be so important to the nature of our improvement.

If we could do it all over again, we would have used less time on testing the radar, and trying to find ideas for some new kind of logical sensor for the TRIO. I would have spent more time on the context expansion, then I would probably have had time to finish the zone implementation as well.

Given more time, I would also have done more verification scenarios to check if the improvement of POD is working as expected and that it does not have any unforeseen downsides.

6.3 Future work

Given more time there are several aspects of the TRIO application we would have liked to have spent more time on:

We would have liked to implement the improvement idea for the zone implementation, as we feel that this would have given even more value to the TRIO system.

It would also have been interesting to work with triangulation in three dimensions, expanding TRIO to work in the third dimension. This would be very useful, as we would not only be able to track where the person is, but also what the person is doing. For instance, we would be able to see if the monitored person was standing or lying. If we had completed our zone improvement, TRIO would know if a person was lying/standing on the floor or a sofa. With triangulation in three dimensions the system would be better equipped to detect falls. This way the TRIO system would be even better at calculating correct risk.

We would also have liked to implement the support in TRIO for heart rate measurements, so that the system would not only monitor a persons movements, but also used the radar for heart beat measurements. First of all the monitoring of heartbeat in bed, but this could also be expanded to respiratory monitoring.

It would also be interesting to explore what the usability for other shapes of zones would be. For instance, round zones or zones that where not restricted by four corners. E.g., if we could create an L-shaped zone, this could be useful for having a zone that was an L-shaped sofa. This would also let the system better handle different shapes of room zones.

Appendix A

Experiment overview

Filename	Description
Ex.1,1-5	Object put 2m center in front of radar after clutter-map is created start live feed then move the object and record 10 sec distance between radars 1.07. Rangeweight 0.2.
Ex.2,1-5	Object put 2m center in front of radar after clutter-map is created, before live feed then live feed and record 10 sec distance between radars 1.07. Rangeweight 0.2.
Ex.3,1-5	Person standing 2m center in front of radar after clutter-map is created and before live feed then record 10 sec distance between radars 1.07m. Rangeweight 0.2.
Ex.4,1-5	Empty clutter-map, record with no target 10 sec distance between radars 1.07m. Rangeweight 0.2.
Ex.5,1-5	A chair with a bag 2m center front of radar before clutter-map is created. Record 10 sec with no target distance between radars 1.07m. Rangeweight 0.2.
Ex.6,1-5	Person standing 2m center in front of radar after clutter-map is created before live feed then record 10 sec distance between radars 2m. Rangeweight 0.2.
Ex.7,1-5	Person standing 2m center in front of the radar standing sideways before the clutter-map is created before live feed then record 10 sec distance between radars 2m . Rangeweight 0.2.
Ex.8,1-3	Person standing 2m center in front of the radar. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.9,1-3	Person standing 2m center in front of the radar. Radars tilted horizontal. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.10,1-3	Person standing center in grid nr. 5. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14

Ex.11,1-3	Person standing center in grid nr. 6. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.12,1-3	Person standing center in grid nr. 7. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.13,1-3	Person standing center in grid nr. 8. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.14,1-3	Person standing center in grid nr. 9. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.15,1-3	Person standing center in grid nr. 10. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.16,1-3	Person standing center in grid nr. 11. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.17,1-3	Person standing center in grid nr. 12. Radars tilted vertically. Distance between radars 2m. 10s recording. Rangeweight 0.2. Date 27.10.14
Ex.18,1-3	Person walking from grid 6 through 10, 11, 7 and ending up in 6 counter clockwise. Framestitch 3. Radar distance 2m vertical. Walking slow. Date 28.10.14
Ex.19,1-3	Person walking from grid 5 through 9, 10, 11, 12, 8, 7, 6 and ending up in 5 counter clockwise. Framestitch 3. Radar distance 2m vertical. Walking slow. Date 28.10.14
Ex.20,1-3	Person walking from grid 6 through 10, 11, 7 and ending up in 6 counter clockwise. Framestitch 3. Radar distance 2m vertical. Walking normal. Date 28.10.14
Ex.21,1-3	Person walking from grid 5 through 9, 10, 11, 12, 8, 7, 6 and ending up in 5 counter clockwise. Framestitch 3. Radar distance 2m vertical. Walking normal. Date 28.10.14
Ex.22,1-3	Person walking from grid 6 through 10, 11, 7 and ending up in 6 counter clockwise. Framestitch 3. Radar distance 2m vertical. Walking fast. Date 28.10.14
Ex.23,1-3	Person walking from grid 5 through 9, 10, 11, 12, 8, 7, 6 and ending up in 5 counter clockwise. Framestitch 3. Radar distance 2m vertical. Walking fast. Date 28.10.14
Ex.24,1-31,3	Standing still in zone 5,6,7,8,9,10,11,12. Radar distance 2m horizontal.

Ex.32,1-3	Person walking from grid 6 through 10, 11, 7 and ending up in 6 counter clockwise. Framestitch 3. Radar distance 2m horizontal. Walking slow. Date 29.10.14
Ex.33,1-3	Person walking from grid 6 through 10, 11, 7 and ending up in 6 counter clockwise. Framestitch 3. Radar distance 2m horizontal. Walking normal. Date 29.10.14
Ex.34,1-3	Person walking from grid 6 through 10, 11, 7 and ending up in 6 counter clockwise. Framestitch 3. Radar distance 2m horizontal. Walking fast. Date 29.10.14
Ex35,1-3	Person walking from grid 5 through 9, 10, 11, 12, 8, 7, 6 and ending up in 5 counter clockwise. Framestitch 3. Radar distance 2m horizontal. Walking slow. Date 29.10.14
Ex.36,1-3	Person walking from grid 5 through 9, 10, 11, 12, 8, 7, 6 and ending up in 5 counter clockwise. Framestitch 3. Radar distance 2m horizontal. Walking normal. Date 29.10.14
Ex.37,1-3	Person walking from grid 5 through 9, 10, 11, 12, 8, 7, 6 and ending up in 5 counter clockwise. Framestitch 3. Radar distance 2m horizontal. Walking fast. Date 29.10.14

Bibliography

[1] Esper, http://www.espertech.com/esper/.

[2] Trio: "safety radar for home care" (in norwegian), project description. 2012.

[3] Tove Irene Slaastad Anders Falnes-Dalheim. Færre unge - flere eldre, https://www.ssb.no/befolkning/artikler-og-publikasjoner/faerre-unge-flere-eldre.

[4] Copyright © 2015 Novelda AS. About us, https://xethru.com/about-us.html.

[5] Ben Benfold and Ian Reid. Stable multi-target tracking in real-time surveillance video. In *CVPR*, pages 3457–3464, June 2011.

[6] Andreas Berglihn. Towards improved indoor positioning using xethru radar-sensors. Master's thesis, University of Oslo, 2015.

[7] D. E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young. Computing as a discipline. *Commun. ACM*, 32(1):9–23, January 1989.

[8] Daintith and Wright. context awareness.

[9] Anind K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[10] Oxford Dictionaries. "context". oxford dictionaries.

[11] José Encarnação Emile Aarts. *True Visions, The Emergence of Ambient Intelligence*. Springer Berlin Heidelberg, 2006.

[12] F. Erden, S. Velipasalar, A. Z. Alkar, and A. E. Cetin. Sensors in assisted living: A survey of signal and image processing methods. *IEEE Signal Processing Magazine*, 33(2):36–44, March 2016.

[13] Opher Etzion and Peter Niblett. *Event Processing in Action*. Manning Publications Co., Greenwich, CT, USA, 1st edition, 2010.

[14] Jürgen Nehmer, Martin Becker, Arthur Karshmer, and Rosemarie Lamm. Living assistance systems: An ambient intelligence approach. In *Proceedings of the 28th International Conference on Software Engineering*, ICSE '06, pages 43–50, New York, NY, USA, 2006. ACM.

[15] Novelda AS. *Quick start guide*, November 2012.

[16] Novelda AS. *2D positioning system using the Novelda NVA-R661 radar kits and Matlab*, 0.1 edition, February 2013.

[17] S. Sægrov, A. Eichhorn, J. Emerslund, H. K. Stensland, C. Griwodz, D. Johansen, and P. Halvorsen. Demo: Bagadus an integrated system for soccer analysis. In *Distributed Smart Cameras (ICDSC), 2012 Sixth International Conference on*, pages 1–2, Oct 2012.

[18] B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. Proceedings., Workshop on*, pages 85–90, Dec 1994.

[19] Jarle Søberg. *CommonSens*. PhD thesis, University of Oslo Department of Informatics, 2011.