*DRAFT*

Master thesis report for the MSc Embedded Systems

TU Delft – Interactive Intelligence

User valued smart reminders: Finding Appropriate Moments for Support in Socially Adaptive Electronic Partners

Remy Kabel

4132165

Supervised by:  
Dr. M. Birna van Riemsdijk  
Dr. Myrthe Tielman

03-11-2018

# Abstract

This project will focus on finding what defines an appropriate moment in regards to providing support through a Social Adaptive Electronic Partner (SAEP). It paves the way to ultimately answering the question “Given a user’s daily activity, what is considered an appropriate time for support feed-back, taking into consideration the user’s norms and values, to achieve a certain goal?”. **TODO**

# Table of common terms

|  |  |
| --- | --- |
| **Term** | **Description** |
| ADL | Activities of daily living |
| SAEP | Socially Adaptive Electronic Partner |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Table of Contents

[Abstract 2](#_Toc529108264)

[Table of common terms 3](#_Toc529108265)

[Table of Contents 4](#_Toc529108266)

[1 Introduction 5](#_Toc529108267)

[1.1 Approach 6](#_Toc529108268)

[2 Background and Related Work 7](#_Toc529108269)

[2.1 Background 7](#_Toc529108270)

[2.2 Existing implementations 7](#_Toc529108271)

[2.3 Related work 7](#_Toc529108272)

[3 Research Approach 9](#_Toc529108273)

[3.1 Research questions 9](#_Toc529108274)

[3.2 Roadmap 10](#_Toc529108275)

[4 Literature Study 12](#_Toc529108276)

[4.1 Model requirements 12](#_Toc529108277)

[4.2 Existing implementations 12](#_Toc529108278)

[4.3 Comparison 17](#_Toc529108279)

[4.4 Feasibility 18](#_Toc529108280)

[4.5 Conclusions 19](#_Toc529108281)

[5 Concept design 20](#_Toc529108282)

[5.1 Activity prediction 20](#_Toc529108283)

[5.1.1 Expectation Maximization 20](#_Toc529108284)

[5.1.2 Apriori algorithm 21](#_Toc529108285)

[5.2 Value based design 23](#_Toc529108286)

[5.3 Concept description 23](#_Toc529108287)

[6 Implementation 24](#_Toc529108288)

[6.1 Platform 24](#_Toc529108289)

[6.1.1 Internet of Things 24](#_Toc529108290)

[6.1.2 Programming language 25](#_Toc529108291)

[6.1.3 Set-up 25](#_Toc529108292)

[7 System architecture 26](#_Toc529108293)

[8 References 27](#_Toc529108294)

# Introduction

The use of technology to support the daily lives of people is an ever-prevalent topic. Through applications in smart homes, wearables, virtual coaches and many others, we can improve our health, efficiency and be more connected. Conversely, the abundance of apps and notifications causes us to grow immune to the constant stream of information that is presented to us in a daily basis [1]. Especially the elderly or people with a mental impairment could benefit from an effective support agent [2]–[7]. In order to create a truly effective support agent, it is crucial to not only generate feedback in relation to the user’s actions but to provide this feedback at an appropriate time.

But what actually is an appropriate time? The appropriate time for feedback is inherently linked to the nature of the user’s action. To illustrate this, consider the following example throughout this report.

An elderly gentleman, Peter, often forgets to close the garden doors before leaving the house or going to sleep.

In this example, timely notification is of the essence. Preferably, notification just before sleeping or leaving the house is desired. Generally, these are quite predictable activities. In the current technological landscape, a simple scheduled notification would be the likely solution. Possibly a geofence[[1]](#footnote-1) may be used to trigger a notification when leaving the house, but this will be post factum.

Identifying such an appropriate time for support feedback for a specific scenario is not difficult. The difficulty of this lies in the generalization. While the above examples can be implemented relatively easy at design time, diversions from normal behavior are not handled. Existing technologies are often made by hardwiring norms and as such are very rigid and unable to adapt to evolving norms [8]. Furthermore, dealing with different problems, such as remembering to turn on the alarm system before leaving work, would require a completely different implementation. Nonetheless, generalization requires analysis of goals and the values underlying the user’s daily activities.

## Approach

The problem of finding boils down to a few steps; each worth further analysis in their own right. Working our way back, the first question that arises is what defines the goal. The goal is defined by the users and can be anything such as: *“I want my garden doors to be closed when I go to sleep or leave the house”.* Assuming we know the user’s activities of daily living (ADL), and optionally the status of the garden doors at any moment, the first step is analyzing which prerequisites there are to attaining that goal. Usually, a goal is not an independent action taken, but rather the consequence of a series of actions. As such, knowledge is required on how a goal be deconstructed into a number of distinct prerequisites.

In order to analyze arriving at this goal, some sort of model needs to be created from the user’s ADL. Once this model has been created, we can use it to analyze the limits of the possible moments for support. More directly, the prerequisites will indicate a number of actions that will have to have been completed, but also some actions may not have been completed. For example, a user will first have to arrive home, but should have received the support feedback before leaving once again, when the user will need their keys. However, Finding the most suitable moment for support is dependent on more than just this.

Finding the most “appropriate” time for the support feedback boils down to finding a moment which is both maximally effective and minimally invasive. Depending on the chosen solution, a number of other values are negatively affected. For example, sounding an alarm in the middle of a person’s sleep may be very effective, but it sure is annoying. The problem is, however, that it’s difficult to quantify invasiveness.

Summarizing, the required steps are:

* Definition of the goal and its prerequisites
* Analysis and modelling of the user’s ADL
* Analysis of effectivity
* Analysis of invasiveness

*(This should, however, be analyzed with respect to the consequences of not remembering.) In case Peter forgets before sleeping, he will either wake up with a sense of insecurity, or if he wakes up at night, he will have to get out of bed and properly interrupt his sleep. If he forgets and leaves the house, the only solutions may be to return home, to ask a friend, or to leave it be. In all cases, his value of security will be diminished, let alone if a break-in were to actually happen.* ***Dit moet nog ergens***

# Background and Related Work

There are plentiful existing implementations, related papers and interesting concepts. This chapter revolves around those existing and past works, in service of finding an approach to the aforementioned problems.

## Background

**Is dit echt nog nodig?**

## Existing implementations

More and more apps are taking advantage of the increased use of smart devices and services in order to get a more accurate picture of the user’s ADL. The following examples are finished

Olisto/IFTTT [9], [10] Can combine date, location and smart device information to, for example, give reminders when leaving home and a specific power consumption is still high (i.e. the TV is still on) and subsequently turn it off.

Maps/Waze [11]–[13] Combines real-time traffic information and address in calendar events to provide timely departure reminders.

Timeful [14] Combines user activity, calendar and to-do items to estimate duration of to-do items, plan them in and generate reminders at off-peak times.

While very promising implementations, most apps predominantly rely on design time logic. Exceptions to this usually create a predictive model and verify this with the user in order to strengthen the model [14], [15].

## Related work

There have been various approaches as to how and when to provide feedback to the user. Generally, the preferred method of feedback is “smart reminders” [16]. Similar to the implementations, papers frequently focus on finding novel ways of combining information from smart devices into producing reminders, following norms provided at design time. Examples include combinations of location and time [17]–[19], events based on smart devices [3], [20], [21], or a combination of numerous sources of information [22]–[24].

The more innovative ideas add an extra logic layer on top of the data of the user’s ADL. Analyzing the user’s values is an intrinsic part of establishing a model. The concept of a Socially Adaptive Electronic Partner (SAEP) has been previously introduced by van Riemsdijk [8]. It follows the ideology that technology should adapt to the user and not vice versa. As such, its logic incorporates the norms and values of the social context. Subsequent work has been done expanding on this, including temporal logic and analyzing actions and habits. [25]–[27]. A simple but tedious approach is to ask for user feedback whenever values are needed. Instead, Zhou et al. [28] use a fuzzy linguistic approach to determine value levels.

Rather than specifying norms at design time, they are constructed based on the ADL. Several approaches are proposed. Chaminda et al. [29] suggest coupling complex activities that have a strong relationship among initiation and conclusion, such as closing the tap after opening it. Other papers [2], [30] support this analysis of temporal relationships between activities, in order to generate a set of norms for the support agent. Other context-aware approaches vary greatly. For example, Vurgun et al. [31] apply a dynamic Bayesian statistical approach. Giorgini et al. [32] use label propagation algorithms to break down goals and identify all prior actions necessary to achieve the goal.

Another approach for this makes use of Behavior Change Support Systems (BCSS) [33] by applying principles of Human Computer Interaction (HCI) [34]. This practice is used increasingly in health focused applications to make sense of the abundance of data. Examples of applications [35], [36] share large similarities with the analysis of the user’s norms and values.

# Research Approach

As previously mentioned, there are several steps in finding an appropriate time for supportive feedback. However, time is limited and several aspects have already been researched plenty. As such, let us limit the focus of the thesis research.

The first two steps, goal definition and ADL analysis are all linked to activity recognition and analysis to provide smart reminders. As discussed in the previous paragraph, several models covering exactly this already exist. Each having their own properties, advantages and disadvantages.

The effectiveness and invasiveness are both quite difficult to quantify. However, they can be combined into user values. These, in return, are more quantifiable. To illustrate this, let us revisit the example of the elderly man, Peter. The goal, taking his medicine in time, drastically promotes his value of health. The moment the supportive feedback is provided, however, may demote that value or another. For example, if it causes him to wake up from his sleep, it will demote his value of health, or if it interrupts him during a phone call it may demote his value of social contact.

## Research questions

Combining the previous matters and these realizations, the focus of this thesis will be combining the concepts of a SAEP and expanding on the existing research as discussed before. The overall research question is:

How can existing smart reminder systems be extended to incorporate user values to provide appropriately timed supportive feedback and thereby increase the user values.

The expected outcome of this question is a model which provides timed feedback based on the user’s ADL and value input.

Subsequently this leads to a number of sub-questions that need to be answered before this:

R1: What are the requirements for the smart reminder system model?

R2: Which existing models and systems exist for smart reminder systems and how do they compare.

These two questions should provide a good overview on the abilities of the existing systems and the amount of work required to extend them to incorporate user values. Of course, for this we need to be able to actually find out about the user values.

R3: What are possible ways of analyzing and quantifying the values of the user?

R4: How can the model be extended to incorporate user values?

Ultimately, all knowledge can be combined into a model which can be used to approximate the most “appropriate time” for support feedback. This model can subsequently be implemented in a piece of software in order for the model to be dynamically generated depending on new input regarding the ADL, goals, norms and values. Once such an implementation has been made, the model can be tweaked according to findings and should be tested. This brings us to the final sub-question:

R5: Does the use of the extended model improve support for user values?

This will require prior planning of possible testing methods and clearly defined testing scenarios.

## Roadmap

#### Literature study

An extension of the preliminary research, focusing on papers related to answering the research questions. Specifically, this revolves around analyzing and comparing past papers and reports to see possible ways of doing activity prediction, analyzing user goals and values, and ultimately combining them. All concepts should be compared on a number of key points, to quickly establish the most valuable papers.

Subsequently, all sufficiently valuable and interesting concept should be checked for feasibility and, if necessary, reproducibility.

#### Concept design, implementation & architecture

**Of noemen we het concept design, methodology & implementation**

Based on the findings of the feasibility analysis, an initial concept can be designed. Accordingly, all required sub-components are analyzed and possible options for their implementations are discussed, leading to a final, defined design for the system architecture.

#### Experimentation and evaluation

Once the complete methodology has been established, the implementation can be tested upon actual data. This requires three things: a completed implementation, a suitable dataset and a method of evaluation.

# Literature Study

Within the literature study we aim to answer the first three research sub-questions. **In final design doe ergens wolkje met de RQ’s** These are necessary before a model can be created in which the answers to these questions can be combined into a concept design aimed at answering the fourth sub-question.

## Model requirements

Blah

## Existing implementations

In this section, the concepts previously mentioned in 2.2 and 2.3, as well as several others, are analyzed and compared to the aforementioned requirements. The papers mentioned all focus on one or more of the following aspects: activity prediction, smart reminders, goal reasoning or user values. Ultimately, an implementation is desired which combines all four of those aspects, or at least several of them.

#### AHCS/TAFETA [24], [4]:

These concepts attempt to design a context-aware application which analyses data from various sensors within the user’s house. AHCS makes use of the CASanDRA framework [37] in order to create awareness of the user’s context. The CASanDRA framework is a middleware which provides easily consumable context information and accepts different information inputs which are fused together. The concepts use either the middleware or their own AI to analyze the collected information and compare this with a number of predefined rules to provide detailed information on the user to the caregiver and provide reminders when rules are broken.

Special properties:

* Context analysis independent from reminder system
* Levels and types of alerting

#### CogKnow [22]:

This concept actually touches upon user values, but instead uses them to define the required support. A distinct number of support scenarios are handled and rulesets are defined accordingly. Predominantly the user context is considered rather than anything else. The rulesets are aimed at avoiding interruptions of important activities, but don’t do any further analysis.

#### Gate reminder [20]:

This concept centralizes around providing reminders at the moment a user leaves their house. Knowledge about possibly forgotten items is obtained through the use of RFID tags, focusing on a zero user workload interaction. A crucial part in its working is that it is focused on Korean household, where shoes are generally left at the front door, so there is a clearly defined time slot in which all tags can be analyzed. Focus on the study was mostly the actual prototype rather than any smart algorithm.

*Special properties:*

* Physical prototype
* Transparent interaction
* Object detection

#### Goal models [32]:

This concept does not directly involve itself with reminders, but rather with linking certain activities to achieving certain goals. These activities may have complex relations with one another and may promote or demote a goal. As such, this can be similarly applied to activities aiming to achieve a certain goal where the promotions and demotions are linked to the user values.

Special properties:

* Linking activities to goals
* Not related to reminders

#### HeadacheCoach [35]:

While not directly a reminder system, HeadacheCoach does propose a possibly usable system. It uses user and environmental context analysis to identify possible triggers for a headache and consequently provides possible solution. A similar approach may be used to identify moments of lower cognitive ability in order to preempt a reminder being necessary at all.

#### MagHive [38]:

This honeycomb shaped magnetic smart surface is attached to the wall and allows devices and other objects to be placed on them. Aside from the useful functionalities such as wireless phone charging, it uses NFC and QI technologies to detect the presence and identity of the objects. As such it is able to remind the user when he or she forgets to take or put back an item.

Special properties:

* Actual product
* Provides a great base for further development

#### MLCARS [39]:

This dissertation discusses a concept which uses machine learning to analyze shopping items and where they were bought (or cleared off the to-do list) to predict similar available items or similar stores. This data is collected among all users and combined with information from companies and stores and ultimately stored in a database which is continuously updated. Combining this with the data of the user’s shopping list as well as their location allows to provide appropriately timed reminders not to forget items from their shopping list. These reminders are not just when near their usual supermarket (like is already possible with location-based reminders) but also when close to any store that is expected to have the desired item.

Special properties:

* Activity clustering
* Prediction of next activity without machine learning

#### Olisto/IFTTT/CAMP/CybreMinder [9], [10], [31], [40]:

These apps and concepts allow setting reminders based on various aspects of user and environment contexts. Once the current situation satisfies all conditions in all contexts, the user is automatically notified. Information is retrieved from the user’s (IoT) devices and (online) services. No form of pattern recognition or prediction is done, however.

Special properties:

* Existing (possibly discontinued) apps

#### Smart reminder system [29]:

This concept creates a smart reminder system through three major components: activity recognition, location recognition and prediction. The activity recognition is done through the use of analysis of the hand movements over time and applying machine learning algorithms and fuzzy logic to map this to activities. Location recognition is done through image recognition by camera and neural networks. These two are then combined to analyze coupled activities, two activities that are strongly related. Alongside, predictions are made regarding pending and forgotten activities. As such reminders can be produced when likely to be forgotten activities should occur.

Special properties:

* Specific setup

#### Attelia [1]:

Attelia is a middleware concept which intercepts any notifications. It analyses breakpoints in the user’s mobile interactions and adaptively delivers the notification to minimize interruptions and the user’s attentional overload. As such, it lowers the user’s frustration caused by receiving too many notifications.

Special properties:

* Focuses on mobile screen use to analyze activity

#### Decision maker [41]:

This concept intercepts notifications from all sources and processes them in a “decision maker” prior to actually arriving at the user. Instead, it processes information from sensors and IoT devices within user and environment contexts to decide upon the target device, type of notification and time of notification. This is done using a machine learning approach. Rather than analyzing the actual patterns in decisions on whether to and how to notify, the paper continues by focusing mostly on the speed and accuracy of various machine learning algorithms.

Special properties:

* Machine learning
* Habit analysis

#### Fuzzy linguistics [28]:

This concept uses fuzzy logic and linguistic variables to analyze the urgency of the reminder and the level of annoyance created by the interruption of the current activity. Resulting from this is a reminder level which determines whether or not the reminder is delayed and/or how the reminder is presented. The urgencies and other levels are all given at design time, however, and are averaged over all users tested prior.

#### PAIR [2]:

This is a relatively older paper which describes one of the first, more advance planners. It takes into consideration several rules as prescribed by the user or caregiver and lays them alongside the activities of the user to provide appropriate reminders. However, no dynamic analysis is done, only design time rules are analyzed.

#### CIA [16]:

Although this paper clearly states “smart reminder”, it doesn’t actually do much in regards to reminding. Instead, it uses image recognition to identify people. After this identification it combines information previously gathered through various systems to display information regarding this person and possible events and reminders tied to them.

Special properties:

* Linking information
* Not directly related to reminders

#### Long term evaluation of smart homes [42]:

Another one not related to reminders per se. This dissertation reviews the users values over long time use of smart home appliances. Their conclusions span generally across all types of smart home appliances. In order for the appliances to provide usefulness it is important that the values of accessibility and trust are upheld. Any appliance which does promote accessibility immediately diminishes any usefulness for the user. Trust generally boils down to the reliability of the provided functionality. If the product still has function impairing bugs, users are likely not to use the product. Even if the producer manages to fix the flaws, the lost trust takes vast time to recover. Another drawn conclusion is that whatever solution implemented, users are initially curious and excited and are willing to try most ideas, but ultimately go back to their routine behavior. As such, the smart appliance should blend into this rather than interrupting it.

#### TEREDA [30]:

Another concept not directly related to reminders. It works by gathering simple data from many sensors around the house and feeding that into the middleware. From this, distributions for the start time and duration are analyzed and used to help recognize activities and cluster them by starting time. For example, there might be 4 clusters of starting times in which the user may generally start to watch TV (with corresponding durations). Each of these clusters may have different subsequent activities, each with different likelihoods. As such, this temporal analysis may be used to predict the likely following activity.

Special properties:

* Activity clustering
* Prediction of next activity

#### What should I do/Action Hierarchies [27], [43]:

These two papers, while again not a directly related to reminders, do portray several underlaying concepts. The first paper presents a framework which represents hierarchical relationships among actions. This is formalized in the second paper. Secondly, this framework shows how the relationships tie in with promotion and demotion of values. Lastly, a method is shown on how to infer norms from values rather than vice versa. However, this remains a very theoretical paper.

Special properties:

* Values → Norms
* Not directly related to reminders
* Action hierarchy

## Comparison

Below is a comparison of the previously described implementations per the requirements stated in 4.1.

| Concept | RP | RI | SS | Tim | Loc | Act | Env | Dyn | UV | IA | Ref. |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| AHCS | x |  |  | x | x | x | x |  |  |  | [24] |
| CAMP | x |  |  |  |  |  |  |  |  |  | [31] |
| CogKnow | x |  | x | x | x | x |  |  | x\* | x | [22] |
| CybreMinder | x |  |  | x | x | x |  |  |  |  | [40] |
| Gate reminder | x |  | x |  |  |  | x |  |  |  | [20] |
| IFTTT | x |  |  | x | x | x | x |  |  |  | [10] |
| MagHive | x |  | x |  |  |  | x | x |  |  | [38] |
| MLCARS | x |  |  |  | x |  |  | x |  |  | [39] |
| Olisto | x |  |  | x | x | x | x |  |  |  | [9] |
| SRS | x |  | x |  | x | x |  | x |  |  | [29] |
| TAFETA | x |  |  | x | x | x | x |  |  |  | [4] |
| Attelia |  | x |  |  |  |  |  |  |  | x | [1] |
| Decision maker |  | x |  | x | x | x | x | x |  |  | [41] |
| Fuzzy lingustics |  | x |  |  |  | x |  |  |  | x | [28] |
| PAIR |  | x |  |  |  | x |  |  |  |  | [2] |
| CIA |  |  |  |  |  |  |  |  |  |  | [16] |
| Goal models |  |  |  |  |  | x |  | x |  |  | [32] |
| HeadacheCoach |  |  |  | x | x | x | x |  |  |  | [35] |
| LTE SH |  |  |  |  |  |  |  |  | x |  | [42] |
| TEREDA |  |  |  | x |  | x |  | x |  |  | [30] |
| WSID/AH |  |  |  | x | x | x | x | x | x |  | [27], [43] |

\* Only at design time

**Legend:**

RP: Reminder Producing

RI: Reminder Intercepting

SS: Specific setup

Tim: Time

Loc: Location

Act: Activity

Env: Environment

Dyn: Dynamic

UV: User Values

IA: Interrupt Analysis

The first fifteen discussed papers and concepts are all smart reminder concepts, whereas the last few describe related concepts such as activity recognition, goal reasoning user values and temporal relations. Overall, the smart reminder concepts can be sub-divided into several groups.

Firstly, the most prominent are those that produce reminders opposed to those that take existing or planned reminders and intercept and process them in some way before actually presenting them to the user.

Secondly, there is a number of concepts which require a specific set-up of hardware opposed to more general, theoretic or software based concepts. These concepts are quite apt and able for those scenarios, but quickly fall short when applied to other scenarios or when generalizing the solution.

The majority of the concepts use (or can use) information about the user or their environment to some extent. Frequently, aside from time, other variables such as location, activity or even weather are used as triggers or conditions for reminders. However, most of these solutions use this information at design time. There are only a few which take it further and use machine learning or other methods in order to create a dynamic system and, for example, predict the subsequent activity and use this information to improve the reminder system.

Lastly, user values are not something generally linked to timing smart reminders. To less or more extent, however, they are being used at design time to shape the model.

So, what is useful? There is no existing implementation that can immediately be extended with user values. However, there are several implementations that contain interesting ideas that can be combined. Most notably [4], [9], [24], [28], [30], [32], [43].

## Feasibility

If the desire is to combine the concept of the aforementioned papers, just like with any store-bought product, it is important to check whether they actually deliver. Additionally, the question is to what extent these concepts can be used to create a value based smart reminder system.

Firstly, the data collection concepts. Thank to close ties with the company behind Olisto [9], access is granted to all services and code behind. As such, a simple middleware can easily be built and integrated into their existing infrastructure. Using their information provides direct insight into events (such as device alarms) and states of devices (such as door open or closed) and services (such as weather). This is already an up and running platform, so lots of data is readily available.

Aside from gathering and analyzing data ourselves, there are numerous existing data sets. Two of which, are directly provided [44], [45] and probably more are available. These datasets have a range of activities and other data recorded over time. Note that in [41], the first dataset was used, but synthetically enhanced to add several properties such as the user activity other than call information and mobile phone usage. The second dataset has a limited but clear number of activities which are recognized and as such more readily usable. These, and similar, datasets can be used both for design and for testing. The range of activities recorded in these datasets and platforms limits the applicable scenarios. As such the data source should be chosen before the initial designs are done.

Secondly, how to analyze the incoming data. The part of the program analyzing this, or the middleware, is dependent on the type of incoming data. If fully detailed activity information is incoming, the middleware is not necessary since the any further analysis or manipulation can be directly performed on the data. However, when taking information from sensors, such a middleware has to be used to filter any interesting information.

The first solution is writing such a middleware from scratch. This is the most labor intense solution. However, if the other middleware are not easily implementable or require extensive rewrites, starting from scratch may actually require less work. In [4], they did just that; they designed their own middleware. However, it cannot be used since it remains exactly that, a design. In [24], however, they used an existing middleware [37]. In combination with a context toolkit [46], also used in CybreMinder [40]. The CASanDRA framework [37], however shows great promise since it’s actively used. However, up to this moment, 03-08, I have not been able to find the actual implementation. As such I have contacted the authors of the original paper and those of papers which used/referenced it.

Lastly, the most important aspect is the actual analysis. A wonderful starting point from within this research group is that of Tielman [43]. Combining its ideas of action hierarchies and values with that of goal reasoning [32] and possibly that of temporal analysis [30] could lead to very interesting results. The goal reasoning will allow for analyzing the possibilities of the moments of reminding (i.e. before it’s too late). The temporal analysis will allow for better predictions. The authors have been contacted for the actual code behind. However, the description is clear enough to incorporate it without it as well.

## Conclusions

# Concept design

The initial design is based on combining the ideas of two papers, [30], [43]. Consecutively this model is combined with knowledge of user values to extend the model and allow for statistical analysis in order to predict the most ideal moment for notification. First, the individual concepts are explained and consecutively the combined design.

## Activity prediction

Activity prediction is done based on the TEREDA paper by Nazerfet et al. [30]. It focuses on two concepts to create a model for activity prediction; the Expectation Maximization [47] and Apriori [48], [49] algorithms.

### Expectation Maximization

Expectation Maximization (EM) is a clustering algorithm which works iteratively to find maximum likelihood parameters of a statistical model. It is used when such parameters cannot be solved through equations directly. The reason for this may be missing data points, latent variables, or further, still unobserved, data points are to be assumed.

Within clustering there is a division between two types: hard and soft (or fuzzy) clustering. In hard clustering, an element either belongs to a cluster or it does not. In soft clustering, on the other hand, elements can belong to multiple clusters but with different degrees of belief, or confidence. In order to statistically analyze soft clustering, mixture models can be used.

Mixture models are a probabilistically sound way of analyzing soft clustering cases. With this method, each cluster is described as a generative model[[2]](#footnote-2), such as a Gaussian or multinomial. However, the parameters of the model are unknown (for example the mean and covariance in the case of a Gaussian model).

If the source cluster of each observation is known, the estimation of these parameters is trivially done through a simple calculation. However, even when not knowing the source, as is the case in a clustering problem, the EM-algorithm will guess the cluster each point likely belongs to. This is done by using the Baysal formulae, those of conditional probability. However, in order to use these formulae, the parameters of the models need to be known. This leads to a “chicken and egg” problem. The algorithm works on any n-dimensional dataset by first performing a random estimate (expectation) to the initial parameters and iteratively improving (maximizing) them.

### Apriori algorithm

The Apriori algorithm is a machine learning algorithm used to find patterns in large data sets. Specifically, the patterns of frequent item sets. At its core it attempt to identify frequent item sets in order to generate association rules used to describe general trends in the data. The algorithm finds its roots in analyzing and predicting store transactions to find products frequently bought together.

Every transaction, or customer purchase if looking at the example of a store, is logged in a database. Consequently, a breadth-first search is done to find all items having been purchased at least a percentage of times; the threshold or support. These individual items are extended to larger and larger item sets, given those item sets appear sufficiently often in the database. Using these frequent item data sets, association rules can be generated. The association rules can be described using numerous measures. Among others, there are confidence, lift and conviction [50].

Firstly, the confidence of an association rule indicating X leads to Y, or , is the indication of how often the rule has found to be true. The previously defined support, the indication of how often an item set appears in the data set, can be described as:

Where is a transaction within the database of all transactions . As a result, the confidence of the rule is the proportion of transactions that contain set X, that also contain set Y:

Where is the union of the items in the two sets. Rewritten in probabilities, the support can be seen as simply the probability of an event , where is a transaction containing item set X. However, since regards the items in a set, it can rather be written as . Linking to Bayesian formulae, the confidence can be seen as an estimate of the conditional probability . The drawback of the confidence measure is that it only takes the popularity of itemset X into account.

The lift measure takes both item sets into consideration and compares their dependence to each other to that expected if they were independent of each other. It is defined as:

A lift of 1 would indicate that occurrences of X and Y are independent of each other and thus no rule can be drawn. The higher the value is above 1, the larger the degree in which the occurrence of Y is dependent on that of X and as such is potentially more useful for prediction. Note that a lift below 1 actually indicates that X and Y have a negative impact on each other.

Lastly, the conviction of a rule is an indication of the frequency of an incorrect prediction. It is defined as:

For example, a conviction value of 1.2 indicates that an incorrect prediction occurs 20% more often than if the association was simply by random chance.

The process of the Apriori focuses on first finding all possible datasets which have a minimum support and then creating rules based on the confidence. Depending on the implementation, either just the confidence can be used as a baseline for the rule generation, or a number of measures more. Note that there are more measures of interestingness than just those described above, including, but not limited to, collective strength [51] and leverage [52]. In [30], however, none of the measures other than the confidence are used, which will as such be the starting point for this concept.

The main drawback of the Apriori algorithm is that given the bottom up approach, a large number of subsets are required to be generated. As such, the number of database accesses are very high requiring it to be loaded into memory entirely. Furthermore, the time complexity is obviously very high. Consequently, numerous improved algorithms have been suggested. However, its simplicity makes it much easier to implement on any sort of database. This is interesting because whereas the algorithm is initially only interesting for true transactional databases such as those resulting from stores, the Apriori algorithm can be used to find patterns in any sort of data set.

In the case of [30], the Apriori algorithm is used to analyze following activities given the cluster of the current activity, as previously found using the EM algorithm.

## Value based design

WIP

## Concept description

adsdasd

# Implementation

The implementation is a major aspect in this report. In order to test the proposed concept, a number of things have to be done. First, a suitable platform has to be chosen. This platform should not only allow for all desired datasets to be supported, but preferably also allow for connection to a real-life application for field testing. Secondly, the algorithms of the conceptual design have to be implemented in code and linked to one another and to the data sources. Lastly, the implementation should provide some sort of reporting mechanism which allows analysis of the results.

## Platform

What platform to choose isn’t just dependent on what algorithm is chosen, or what libraries are available. More important is to see how the data is obtained. Keeping an open mind as to where data can come from, and not just restricting oneself to using premade datasets, allowing streaming data is important. Why? Because of the rapid rise in Internet of Things devices.

### Internet of Things

The field of activity recognition is a rapidly evolving one. This is mainly due to the exponential rise in Internet of Things (IoT) devices. Currently, there are over 17 billion connected devices in the world. Of these, there are over 7 billion IoT devices (so excluding smartphones, computers and similar) with over 6.5 million new devices being connected every day [53]. This is expected to grow to between 20 and 200 billion within the next five to ten years. The promise of IoT doesn’t end at just connecting the devices to the internet. It is just the first step.

Advances in RF technology and low power computing will bring Internet-connectivity everywhere. Advances in Big Data and machine learning will unlock new business opportunities and models. The possibilities are nearly endless, but they all still lie quite out of reach from the direct consumer. However, specifically for activity recognition, suddenly a lot more data is available than there was 10 years ago. Consequently, more and more papers and implementations such as **<fill in references>** are analyzing activity based on random sensor data.

Whether the activity data or the sensor data is available, in any case a prediction can be made on past events. As long as the event corresponding to the deadline is known before which the notification should have been presented, any form of data should fit within the design. As such, a server based solution, preferably in the cloud, seems most logical.

### Programming language

When it comes to implementing machine learning algorithms, there are several go to languages. The five most used languages [54], in order, are:

* Python
* C/C++
* Java
* R
* JavaScript

While there are many other options, they fall below a 5% mark of prioritization in the field of machine learning. Python takes the clear lead in this field. This is due to the large number of readily available libraries. This dramatically decreases the time required to implement machine learning algorithms in applications. However, regardless of popularity it is shown that professional background is key to choosing a language.

For now ignoring the fact of whether the programmer has any existing proficiencies, it is important to note that there is no best language to use for machine learning and it is important to take the goal into consideration. In this case the goal is to create a server based cloud platform. Whereas the algorithms can still be run on any language, the web part and a possible API[[3]](#footnote-3) interface are likely to be implemented in JavaScript.

### Set-up

Taking the above choices into consideration and looking at the current professional landscape, there is a single, simple way forward.

Node.js

# System architecture

# References

1. A virtual geographic boundary, defined by GPS or RFID technology, that enables software to trigger a response when a mobile device enters or leaves a particular area. [↑](#footnote-ref-1)
2. In machine learning (and other forms of statistical classification) there are two main approaches: generative and discriminative. Given a target Y and an observation X, the generative model is a statistical model of the joint probability distribution. Whereas the discriminative model looks at conditional probability of Y given X=x. [↑](#footnote-ref-2)
3. See section **<fill in later>** [↑](#footnote-ref-3)