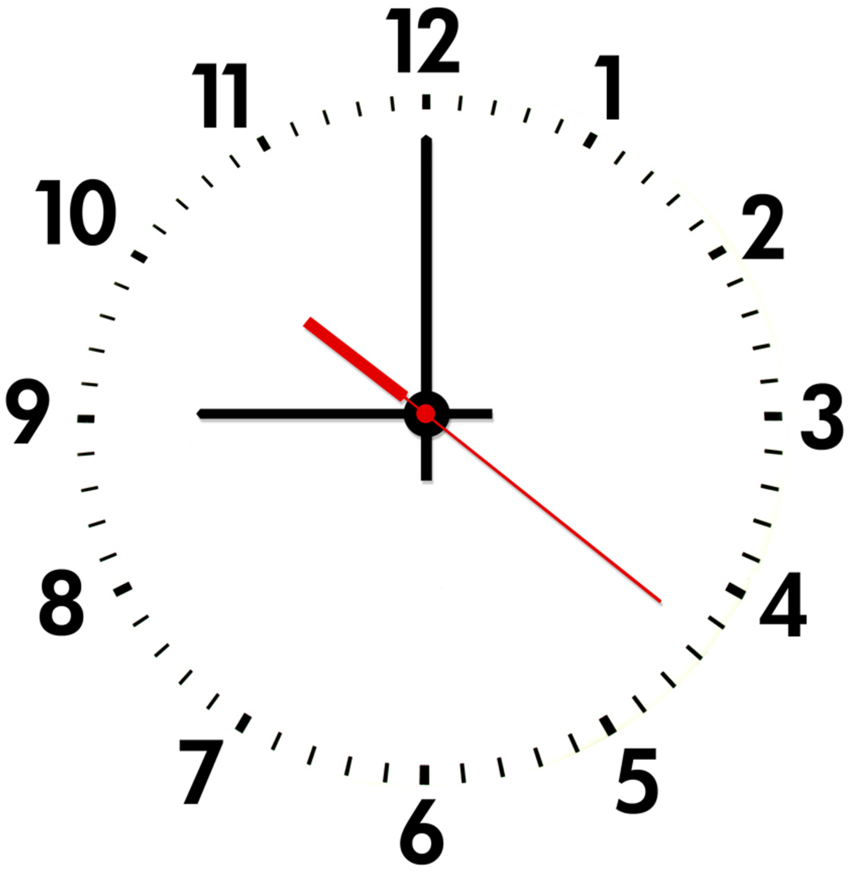
Easy Clocking

- A system for automatically clocking in and out employees



Software 7 - 2009

Aalborg University
Department of Computer Science
Selma Lagerlöfs Vej 300
9220 Aalborg
Telephone:(45)96358080
http://www.cs.aau.dk

Title:
Easy Clocking
- A system for automatically clocking in and out employees.

Theme:
Internet Development

Project timeframe:
SW7, 2nd September - 18th December, 2009

Project group:
sw701b

Group members:
Christian Frost
Casper Svenning Jensen
Kasper Søe Luckow

Supervisor:
Lone Leth Thomsen

Abstract:

Easy Clocking is a system capable of automatically clocking in and out employees based on presence at workstations.

Embedded devices are configured and software is developed to detect the presence of butchers at workstations using a localisation technology. We analyse and compare a number of different localisation technologies from which we choose Bluetooth. Its applicability is evaluated by conducting experiments.

A web application is developed in order to view registered location information and to solve conflicts, such as if the butcher is detected at multiple workstations at the same time.

Work has been done with emphasis on usability and flexibility of the web application, making Easy clocking effective to work with and allows for customisation according to the needs of the company deploying it, respectively.

Copies: 6

Total pages: 156

Of this Appendices: 26

Paper finished: 14th of December 2009

Signatures

Christian Frost

Casper Svenning Jensen

Kasper Søe Luckow

Preface

This report is written by three software-engineering students attending the 7th semester at Aalborg University as a part of their semester project. The project was commenced on the 2nd of September 2009, and finished on December 18th 2009.

During the project of Easy Clocking, we have received help from a number of people which we would like to thank. First of all, our supervisor Lone Leth Thomsen, who has been helpful with advice regarding the project and the content of the report. Rico Wind, who helped us understanding how StreamSpin could be modified to accommodate our needs. Rene Hansen who gave advice on collecting and analysing location data. Finally, Andreas Weisberg and Morten Bested who functioned as test users in our usability test.

Because we are software engineering students, this report will concentrate on subjects related to computer science. Therefore, we assume that the reader has equivalent knowledge in the field of computer science, as that of a 7th semester software engineering student.

Two types of source references are used throughout the report. One is a reference placed after a period which refers to the given section. The other type of reference is placed before a period which refers to the particular sentence or word. The sources of the references used throughout this report, can be found in the bibliography at the end of the report.

A CD-ROM is provided including the Easy Clocking source code.

Aalborg, December 2009
- sw701b

Contents

1 Introduction

Today, employees in both private companies and public institutions use clocking as a central part of their work day. Clocking is the practice of registering when an employee begins and ends a given assignment in order to determine the time spent on it. This information can be used to calculate salary or, as used in the Danish home care[FOA, 2008], to monitor employees and enforce time restrictions on assignments.

Clocking has been shown to add overhead for employees. As an example, the policy of having home carers do clocking on their PDA each time they visit an elderly has been criticised.[FOA, 2008] Additionally, the manual process opens for mistakes potentially creating inconsistencies in the registered data.

We propose a system, named Easy Clocking, capable of detecting employees and registering their location with a minimum amount of interaction from the employees. Hence, the purpose of Easy Clocking is to automate the clocking procedure in order to remove the overhead of manual clocking while also reducing errors in registering start and end times.

Throughout the report, we use a scenario of a Danish butchery in order to relate Easy Clocking to a concrete company, even though Easy Clocking could be applied to other companies as well.

We tried to contact a Danish butchery in order to construct a scenario, create personas and get feedback on Easy Clocking. We were unable to successfully establish such a relationship. The mentioned areas are therefore based on our knowledge.

In the following sections, we state the learning goals of the project, describe the scenario, describe how Easy Clocking is applied to it, and give a summary of the actors using Easy Clocking. Finally, an overview of the structure of the remaining report is given.

1.1 Learning Goals

Besides giving a solution to the clocking problem, the project aimed at fulfilling the goals from the study regulation for a 7^{th} semester software engineering project. Furthermore, we chose additional goals to gain knowledge in different aspects of software engineering. These goals were made on the basis of what we found interesting from untried theoretical aspects and new aspects which were related to the project. The goals from the study regulation and our own goals are listed in the following:

- Demonstrate knowledge and understanding of Internet, Internet technologies and Internet services.
- Demonstrate skills in developing an Internet application.
- Demonstrate skills in developing a user interface for Human-Computer Interaction (HCI).
- Learn how to use a framework when developing an Internet application and additionally learn a new programming language appropriate for this type of development.
- Know the advantages and disadvantages of different localisation technologies and techniques.
- Apply the theoretical knowledge of a development method introduced to us.
- Create a product that helps a customer to solve a specific task.
- Become acquainted with embedded software.
- Gain knowledge in how to make software more flexible through plug-ins.

1.2 Scenario Without Easy Clocking

This section describes our understanding of the work processes related to clocking at the butchery, before deploying Easy Clocking. This helped us to understand the context Easy Clocking needed to fit into.

Danish butcheries use piecework and hourly wages to determine the salary of their butchers. This means they get paid based on the amount of assignments they accomplish, and the clocking information is used to calculate their hourly wages.

Managers determine which products need to be produced for a given day, and create lists of assignments for the individual butchers. These are placed at notice boards at the start of a work day such that the butchers know what they need to do.

When a butcher receives his tasks, he walks to a nearby clocking machine. All butchers carry a laminated card with a unique barcode identifying themselves. When they clock in, they use their personal card by placing it in front of the clocking machine which scans the barcode. Next, the butcher locates and scans the card corresponding to his assignment. The card is placed in a card rack next to the machine, containing a card for each workstation.

When the butcher has completed his task or wants to take a break, he walks to a nearby clocking machine and scans his personal card to clock out. Since many of the assignments performed at the butchery involve continuous work, breaks must be frequently held. Hence, with the amount of breaks along with the number of different assignments a butcher needs to perform during a work day, the butcher is required to clock in and out multiple times a day.

Finally, some workstations are not completely stationary and require the butcher to temporarily leave e.g. for passing intermediate products for further processing.

1.3 Scenario With Easy Clocking

We have used the *application and problem domain* work product to formulate and understand the usage and structure of Easy Clocking. The application domain reflects the work process after Easy Clocking has been deployed, and the problem domain describes the objects in Easy Clocking[Design, 2000].

Before analysing the problem and application domain, we found role models for Easy Clocking. These gave us insight into aspects of this type of product, such as applicable technologies and different deployment purposes. A description of these can be found in Appendix G.

Figure 1.1 depicts the elements of Easy Clocking applied to the scenario.

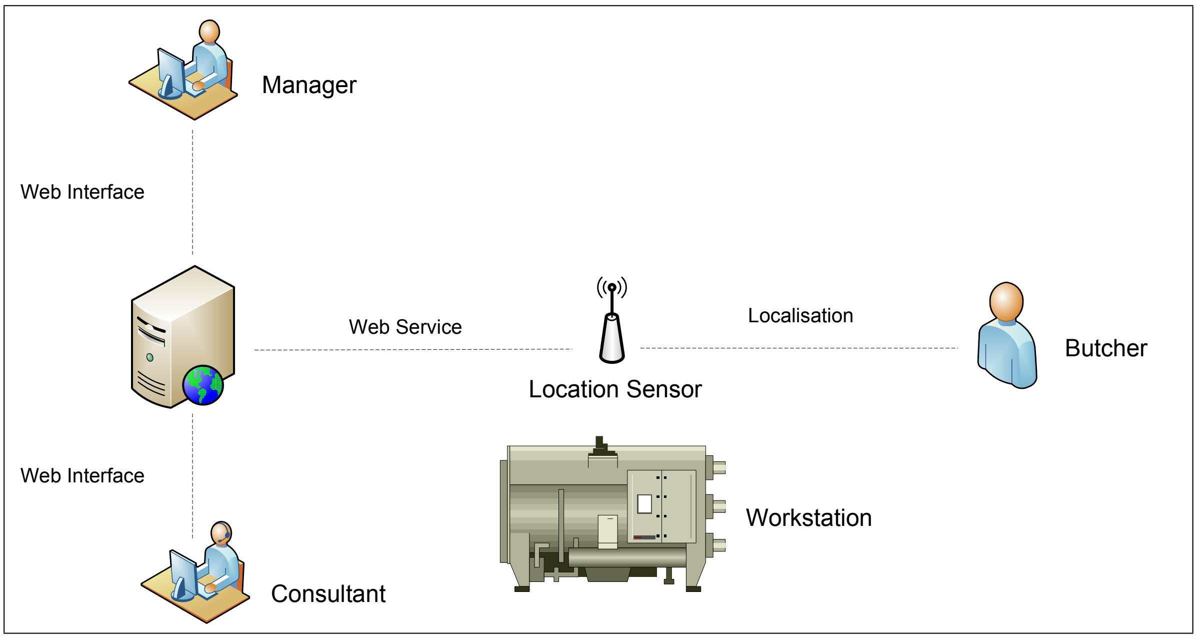


Figure 1.1: Easy Clocking applied to a butchery with management using a web application receiving location data from location sensors.

As shown, managers and Easy Clocking consultants interact with the web application. The butchers are detected by location sensors when working at workstations. From this, clocking information is transferred to the web application.

1.3.1 Application Domain

The following describes a typical work day for a manager, butcher and Easy Clocking consultant using Easy Clocking.

The manager plan the current day's work through a web application by assigning tasks at specific workstations to the butchers. After the assignments have been determined, the manager reviews the previous day's assignments which are cross checked with clocking information from the location sensors. Possible inconsistencies, called conflicts, between the schedule and clocking information are resolved.

The butchers receive their work assignments from printed lists of assignments pinned to information boards. They proceed by going to their assigned workstation and work there until they have completed their task. The butcher's presence at the workstations is registered by the location sensors.

Easy Clocking consultants deploy location sensors for new workstations and use the web application for configuring the individual location sensors in terms of various parameters.

1.3.2 Problem Domain

The problem domain comprises various components as shown in Figure 1.2. The following is a description of these, their relationships and responsibilities.

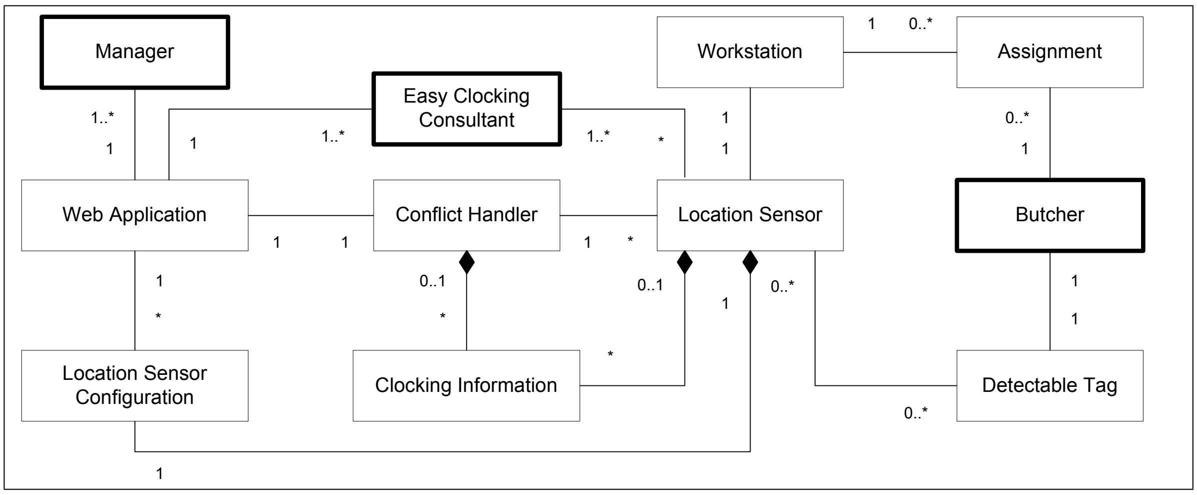


Figure 1.2: Class diagram showing the different entities in our problem domain and their relations.

A manager uses the web application to administrate assignments for the butchers and to interact with the conflict handler in order to resolve conflicts. The Easy Clocking consultants use the web application to add new location sensors or change their configurations. The configurations consist of parameters defining the behaviour of the corresponding location sensor and are propagated to them.

Butchers are assigned to work at workstations during a work day. A location sensor is placed at each workstation and detects nearby detectable tags carried by the butchers. The sensors measure the distance to the tags and register when they enter or leave a given area around the workstation. This clocking information, is stored by the location sensors until they are able to pass it to the conflict handler.

The conflict handler processes the clocking information and stores it for later use. It is the responsibility of the conflict handler to keep track of and aid in solving conflicts in the gathered information. The possible conflicts are described in the following.

Conflicts

As described previously, the butcher is initially assigned tasks that must be conducted during the work day. This procedure gives rise to potential conflicts which are described in the following:

- It may happen that the butcher does not show up at all the workstations corresponding to his assignments. This conflict can only be determined by the end of the work day and resolving it automatically by the system is difficult due to the many reasons for not showing up such as illness and machine breakdown. The system can notify a manager about the conflict who then must correct the conflicting information manually.
- In contrast, it may be the case that the butcher shows up at other workstations than originally assigned by the manager. In this case, the conflict can also be resolved manually by the manager. In addition, if it is the case that the assignments change during the work day, the manager should be able to accommodate this by changing the assignments in the system.
- Finally, a conflict can arise if the butcher is detected the same time at multiple workstations where he was supposed to work during a work day. The range in which the butchers has been detected at the workstations can be used to infer the correct workstation. E.g. if one workstation detects the butcher at 1 meter and another at 8 meters then he likely had been working at the former. If this can not be inferred, the manager must solve the conflict by talking to the butcher.

Some of the above conflicts can be resolved automatically by using a number of assumptions. For instance, if a butcher was not assigned to work at a specific workstation, then we assume those measurements were made when he walked past the station or talked to a person working there.

It would be preferable if all relevant location data have been collected by the web application before the manager solves conflicts for that given day. If not, a manager could solve conflicts for a work day which later would be in conflict with clocking information later received. In this situation the manager would have to solve the conflicts the specific work day again.

1.4 Actors

As mentioned in the problem and application domain, the actors in the system are: butchers, managers, and Easy Clocking consultants. We will describe each of these using personas. A persona is a precise description of an imaginary person representing a generalisation of an actor. The description contains personal information such as age, education and background, and technical information such as skills and needs in relation to the product.[Cooper, 2004]

Realistic personas can be based on interviews of several users, but in our project, where we had no contact to a real butchery, we will make up the personas.

The purpose of using personas is that the developers get a common understanding of the users of the final system. E.g. when discussing if a certain feature should be implemented or not, personas can aid in making the decision. This way, situations where developers may want to add certain features to the product which are irrelevant for the customers can be avoided. [Rind, 2007]

Following is a summary of our personas. A complete persona for each of the actors can be found in Appendix F

Butcher Michael is not acquainted with many different electronic appliances. He knows how to do some of the most basic tasks on the family computer which include tasks such as using home banking, checking his e-mail and playing some of the built-in card games. At work, he knows how to operate many of the industrial machines in terms of configuring them correctly for the particular task they have to perform.

With respect to the old clocking system, Michael has experienced many annoyances. Specifically, when he is in a rush in the morning, it may occur that he forgets to clock in which results in him having to explain the incident to the secretary and assure that he actually arrived at work on time. Also he is annoyed with the fact that the clocking machines occasionally are located relatively far away from the assignment he is working on. Finally, he is tired of the general way manual clocking works. Cards with barcodes specifying the assignment he is about to do together with a card containing a barcode identifying himself need to be scanned in a specific order. He thinks there is too much room for error, especially because many butchers need to do this multiple times a day.

Manager Bo is not acquainted with working with computers when he is at home. He only knows how to perform simple tasks at the various systems at the butchery. Specifically, he knows how to check his internal e-mail through a web-interface and he has attended various computer courses to aid some of the tasks he has to perform at the butchery.

One of his main concerns is to distribute the assignments that need to be performed the given day. It is a manual process involving looking up which orders the butchery has received. The process of distributing the tasks relies on Bo's knowledge of which certifications the butchers have in order to not assign a task incorrectly. Based on this, he devises a note specifying the task assignments which the butchers can look up in the morning.

Consultant Henrik is a very experienced computer enthusiast. He possesses knowledge in programming and knows how to use many different tools such as editors and word-processing tools. Also, he knows how to set up SOHO networks. His interest in programming is however limited to high-level languages and tools that abstracts the low-level details of development.

1.5 Report Overview

We chose to separate the documentation of our development process and our product, such that the report documents the final product. This choice was made primarily because we used adaptive planning and iterative and incremental development. This meant that the complete design of the system was not entirely made before moving on to implementing parts of the system.

In the following chapters we describe our development method, how it was applied to our project, and list the requirements for Easy Clocking. The remaining report is then structured into the following five parts.

Part 1: Analysis We analyse areas in which we need additional knowledge before creating Easy Clocking. These are localisation technologies, plug-in architectures, and usability.

Part 2: Design We describe the technical platform and architecture of Easy Clocking. In addition, we describe the design of the web application and location sensors.

Part 3: Implementation The implementation of Easy Clocking is documented with detailed descriptions of parts of the implementation with code examples, and screenshots.

Part 4: Conclusion We conclude on our project, discuss problems we have encountered, what we have learned, and mention possible improvements.

2 Development Method

This chapter describes a number of possible development methods applicable to our project. One of the methods is selected, described in more detail, and further we describe how it was applied to our project.

We use a software development method to structure and control the development process throughout the project.[Sommerville, 2001]

On previous semesters, we used more traditional development methods such as the waterfall and the incremental method. Last semester, we were introduced to agile development methods. These agile approaches of software development were appealing to us because they did not seem as strict as the traditional methods. Because of this, we wanted to apply an agile method to the project.

2.1 Selection of Development Method

Below, we describe four different agile methods, introduced in the Software Engineering course (Software 6, 2009), and three web oriented methods introduced in the Web Development course (Software 7, 2009). The web oriented methods are relevant because the project, among others, concerns developing a web application.

Evolutionary Development (Evo) Uses a front-room and back-room concept for delivery of components to the customers. It focuses on continuous measurement of progress and risk.

Extreme Programming (XP) Defines an iterative process driven by user stories. It focuses on being customer-driven, which requires the presence of customer representatives under the entire development process.

Scrum Suitable for small teams of maximum seven people. Scrum defines an iterative process driven by the use of product and sprint backlogs. The amount of ceremony and work products used is not defined and should be adapted to the specific project.

Unified Process (UP) Defines an iterative process with focus on handling high risk items in the early iterations. The method also defines a number of work products which can be selected depending on the project.

WebML Defines an iterative process for web development with focus on data design.

WSDM Defines an iterative process for web development with focus on users and their tasks.

OOHDM Defines an iterative process for web development with focus on application navigation.

The web oriented methods were discarded since they did not apply easily to our non-web-related components which are a considerable part of the project. Extreme programming was discarded based on its key value of having an on-site customer at all times combined with practices such as pair programming which we have bad experiences with.

We chose Scrum because it had the most appealing development life-cycle and because of the idea of daily measuring the progress and the usage of backlogs seemed interesting. However, Scrum does not define work products for development activities such as creating requirements. Also, it does not define the structure of the individual iterations. Thus, we decided to use work products, defined in the other development methods, where applicable. This way, we ended up using a tailored version of Scrum.

2.2 Scrum in Detail

This section is based on material obtained from Larman [2003].

Scrum is an iterative and incremental development method which is commonly applied in agile software development. In fact, Scrum can be regarded as a framework since it does not explicitly dictate various work products which should be applied in the different phases. Instead, Scrum puts emphasis on values and practices, which capture the key adaptive and agile qualities and only encourages devising a minimal number of work products for organising the development.

Given the fact that Scrum does not specify which activities must but conducted but fully places this responsibility on the team, it is flexible and can thus be combined with or complement other methods.

2.2.1 Scrum Life Cycle

The Scrum life cycle is composed of four phases. These are briefly summarised in the following.

Planning The goal of this initial phase is to establish the vision and set the expectations of the project. During this phase, all stakeholders are allowed to contribute with various features, use cases etc. which should be recorded in a work product called the product backlog.

Staging In this phase, more features are identified. Additionally, time estimates are set for each task in the product backlog. Together with the product owner, the release backlog is identified. This backlog constitutes tasks which must be completed at the end of the release phase.

Development The purpose of this phase is to develop the system consisting of the items listed in the release backlog. This is done through a series of 30-calendar day iterations called sprints in Scrum terminology. Note that even though a sprint is set to 30-calendar days, only the weekdays are used for developing. Before each sprint, a meeting is held in which the product and release backlogs are refined by the stakeholders. Additionally, they choose goals for the next iteration which may be based on highest business value and risk. The development team, called the Scrum team, meets with the product owner and discuss which subset of the release backlog must go into the sprint backlog. The number of items in the sprint backlog is restricted by the time they have been estimated to complete, since it must not exceed the time available in the 30-calendar day sprint.

Release This phase aims at deploying the release. Also, if documentation is required by the customer, it is made. It may also be the case that the users of the developed system need to be trained. After this phase, the life cycle of Scrum starts over again, and a new subset of goals are selected from the product backlog and placed in the release backlog.

Besides the backlog work products, Scrum also encourages the use of a burn down chart depicting the progress of the given sprint. Specifically, the graph shows the estimated remaining effort required to complete the sprint as a function of calendar-days. Hence, the Scrum team can get an idea of whether or not the total number of tasks that needs to be completed can be realised in the 30-calendar time frame by looking at the tendency of the graph. The sprint backlog graph is updated each day together with the sprint backlog.

2.2.2 Scrum Key Values

As mentioned, Scrum emphasises on various key values. The values we found the most appealing and most different from the traditional methods are explained below.

Self-directed and self-organising team During an iteration, no one besides the Scrum team itself are allowed to interfere in relation to how the team should achieve the iteration goals, solving internal problems and in planning the order of the activities other than when being explicitly requested for help. One team member is designated the Scrum master whose main purpose is to enforce that practices and key principles of Scrum are followed. She also represents a mediator between management and the Scrum team and should also be capable of solving problems when asked.

No extra tasks during an iteration Once management has agreed on the work to be done for a given iteration, it must not be changed during the iteration. This is in order to keep the Scrum team focused.

Scrum meetings Each day, a meeting where all the team members participate must be held. Scrum encourages these meetings to be stand-up meetings where each team member answers a set of questions in turn. From these questions, the team members learn from each other by sharing gained knowledge.

Working demo after each iteration At the end of a sprint, a working demo is shown to the external stakeholders where feedback is encouraged. The feedback can then be used when planning the next sprint, if changes need to be made.

Client-driven adaptive planning Since each sprint must be planned in cooperation with the product owner, planning becomes client-driven. Furthermore, it is important that planning is adaptive. It should e.g. be possible to change the product backlog during the project period. This is in contrast to predictive planning which emphasises on making a specific development plan structured around producing a pre-determined end result within a specific timeframe.

2.3 Employing Scrum

In this section, we describe how we have employed Scrum in the project. The section describes many concepts and decisions not introduced yet and can be skipped until a later point.

We chose to use user stories as a work product, because we previously had been presented with this combination of Scrum and user stories by a company called Atira. They had used this combination with success in their projects.[Atira, 2009]

A user story contains a name and an estimate of the time it takes to finish the story, and if needed acceptance tests. The name must be a keyword or sentence that is used to refresh the conversation regarding the specific functionality.[Cohn, 2004]

The set of user stories comprised our product backlog. In Appendix A, we have shown a picture of our product backlog and an example of a user story.

Before starting each of the sprints, we decided which user stories to be conducted during it. These were then decomposed into tasks such that the estimated time for a task did not require more than a story point. This decomposition was encouraged by Scrum, in order to more easily track the progress. Afterwards, the tasks were placed in a Scrum work product called Sprint Backlog with three states: Planned, Ongoing, and Done.

In a project involving a customer, the choice of user stories to be implemented in a sprint should be made in cooperation with him, but since we did not have a customer, we chose the stories ourselves. We based the selection of user stories on their prioritisation in the project. The prioritisation could be based on high dependency to other user stories, hence making the user story in question central. The prioritisation could also be based on our assessment of the user story having high value for the system.

As described previously, Scrum encourages the use of time estimating the user stories that need to be completed. When making these, we followed the guidelines given by Cohn [2004], who recommends using story points. We defined a story point as a full day of work. Making the time estimates was conducted as a team activity by using the Poker Planning Game[Atira, 2009]. In the Poker Planning Game, each team member estimates a given task and compares it with the estimates made by the other team members. The members with the highest and lowest time estimates, respectively, then explain their estimates. Based on this explanation, new time estimates are made and compared. This activity is done until the team members agree on a time estimate.

The Poker Planning Game gives insight in different member's interpretation of the tasks, and through the subsequent discussion, the team members should achieve a joint understanding of them.

In the following we describe the activities conducted in the phases of Scrum.

2.3.1 Pre-game

We started the pre-game phase by analysing how automatic clocking by localisation could be achieved. In the analysis, we examined how we could localise the butchers, using different localisation techniques and technologies.

After this, we defined the vision of the project, made an analysis of the problem and application domain[Design, 2000], and from these, we identified three actors in the system: butcher, manager, and Easy Clocking consultant, whom we described in detail through personas. These activities were conducted in order to understand the needs of the customer. Using these work products was beneficial for us because they gave a good overview of the project.

The result of the pre-game was a list of user stories.

2.3.2 First Sprint

For the first sprint, we implemented a login system for the web application, capable of authenticating users and restrict their access by using an authorisation mechanism. Also, we created an initial navigation[Dolog et al., 2009] and data design[Dolog et al., 2009] of the web application. These work products were then extended, when necessary, throughout the subsequent sprints.

Also, we wanted to set up the location sensors with new customised firmware and develop a simple application to detect the presence of butchers within its range. The application was developed as a prototype in order to get acquainted with the C Bluetooth library.

Since this was our first sprint, we did not know whether the time estimates were correct. Thus, we decided to have a shorter sprint duration of one week instead of the two weeks used in the subsequent sprints. This way, it was possible for us to get a clear view of whether we should change the estimates in the rest of the sprints. If Scrum is used in future projects, we can omit this experimental sprint and have a first sprint of full duration.

At the end of the sprint, we had a number of unfinished user stories. This indicated that we had underestimated our tasks, and therefore needed to correct this in subsequent sprints. We observed that our estimates relating to the web application were nearly correct, while the estimates related to the location sensor were not. We theorise that this is a result of having more experience in developing web applications than working with the location sensor hardware. Thus, in the future, we should be more aware of time estimates of user stories we have little experience in.

As an example, we had estimated one and a half days for one person to update the firmware and install support for Bluetooth on the location sensors. This ended up taking three days for three persons. This resulted in a number of user stories not being finished in the sprint. These stories were added to the second sprint. As a result of this, we corrected our estimates, in subsequent sprints, and chose to keep a couple of days free as buffer time for accommodating tasks, such as meetings which were not recorded in the Sprint Backlog.

Screenshots of the web application after the first sprint are shown in Appendix B.

2.3.3 Second Sprint

In the second sprint, we completed the unfinished stories from the first sprint in addition to a set of new user stories. The focus of the sprint was to experiment with ranging, and implement ranging and clocking on our location sensors. In addition, we configured the location sensors to form a mesh network and customised the firmware installation by removing unused services.

In contrast to our first sprint, we successfully finished all selected user stories on time even though we still had trouble with some time estimates being too low or too high. Furthermore, we observed that we should set aside more time for tasks not tracked through Scrum than we thought after the first sprint. Even though we had reserved time for meetings and documentation of the project, we underestimated the time needed in order to maintain the report and improve chapters as the project went forward.

2.3.4 Third Sprint

As with the second sprint, we managed to finish all the selected user stories on time in the third sprint.

The sprint mainly focused on implementing assignment and workstation resources in the web application. This was done as a preparation for the fourth phase, where these were needed in order to handle the conflicts.

We also analysed and implemented the communication between the web application and location sensors. This proved to be less time consuming than originally estimated.

The burn down chart showing the progress of this sprint is shown in Appendix C.

2.3.5 Fourth Sprint

The fourth and final sprint of the project focused on usability, conflict handling, realising the plug-in architecture, testing and deployment. The activity of improving the usability of the system started with analysing various design principles related to usability. After conducting the analysis, we worked on improving the user interface using these. Afterwards, a usability test was conducted to discover flaws in the user interface. The flaws uncovered in the test, were evaluated and the user interface was updated accordingly.

Another aspect of this sprint was the realisation of conflict handling. The process of doing this turned out to be difficult because it involved many design issues which were hard to comprehend before doing the implementation. Thus, the code for the conflict handler was rewritten multiple times and much time was spent on this part during the sprint. Also, since the conflict handler did not use the standard templates Ruby on Rails provides, we made a presentation design[Dolog et al., 2009] of it, which helped us in clarifying what the views should contain.

We also analysed how the plug-in architecture could be realised. It turned out that Ruby on Rails already contained ways of doing this. However, the code already written for the web application needed to be re-structured in order to support the usage of plug-ins.

Finally, we started conducting acceptance tests according to the requirements set for the system. Revealed bugs due to these tests were corrected and hence we could start deploying the system. The deployment comprised getting a stationary PC on which the web application was hosted. The location sensors were configured to post their data to this and we verified that everything worked as expected.

3 Requirements

The following lists the requirements for the project. These are divided in functional and non-functional requirements.

The requirements have different purposes for the project. Some are made to fulfil the learning goals, other to define the scope of the project and finally those that were elicited from the user stories. To indicate which of these purposes had formed the particular requirement, we mark them with (Learning goal), (Scope) or (User story), respectively.

3.1 Functional Requirements

The functional requirements for Easy Clocking are divided in web application and location sensor. These are listed below.

3.1.1 Web Application

- The managers and Easy Clocking consultants must be able to log in to the web application. (User story)
- The managers must be able to view/add/edit/delete butchers in the system. (User story)
- The managers must be able to view/add/edit/delete managers in the system. (User story)
- The Easy Clocking consultants must be able to view/add/edit/delete consultants in the system. (User story)
- It must be possible for the managers to view/add/edit/delete assignments at specific workstations to butchers. (User story)
- Easy Clocking consultants must be able to view/add/edit/delete location sensors including their configurations. (User story)

- The Easy Clocking consultants must be able to view the status of a location sensor. (User story)
- It must be possible for the managers to associate a butcher and a tag. (User story)
- The web application must be able to detect conflicts in clocking information and allow managers to solve these. (User story)
- The system must be able to determine if a butcher has been present at multiple workstations. (User story)
- The manager must be able to print out a list of the assignments that need to be conducted for a given day. (User story)

3.1.2 Location Sensor

- The location sensor must be able to detect butchers located within a 10 meters range with 5 meters accuracy. (Scope)
- The location sensor must be able to detect up to ten butchers working simultaneously at the same workstation. (Scope)
- A butcher being within the detecting range, must be clocked in. (User story)
- The location sensor must calculate the distance to a detected butcher. (Learning goal)
- The location sensor must be capable of determining if an employee is absent for more than a specified time interval. If this time is exceeded, the location sensor must mark the butcher as clocked out. (User story)
- If a butcher exceeds a specified range, he must be clocked out. (User story)
- To address the issue of the location sensors being unable to directly communicate with the web application, these should operate in a mesh network such that communication can be routed through other location sensors to the web application. (User story)
- The location sensor must be configurable from the web application. (User story)
- The location sensors must take into account issues with connectivity to the web application. (User story)

3.2 Non-Functional Requirements

Following are the non-functional requirements for Easy Clocking.

- The system must be easy to work with for the managers. (Learning goal)

- The system must be able to handle a capacity of 300 employees corresponding to a relatively large company. (Scope)
- The system must be able to operate indoors since the workstations are primarily located in buildings. (Scope)
- To have the possibility of tailoring the system to the specific needs of a company, it must be extensible through plug-ins. (Scope)
- There should be minimal interaction between the system and the employees carrying the mobile devices. (Scope)
- The mobile devices should be appropriate in terms of dimensions to be carried by the employees. Furthermore, they should be economical in terms of power consumption. (Scope)

Besides these non-functional requirements, we also identified quality factors for Easy Clocking. These are described in the following.

3.2.1 Quality Factors

Quality factors are used to express high-level quality attributes of software. The quality factors represent the customer's interests in the software and which aspects of the software having quality from the customer's point of view.[van Vliet, 2008]

By defining a prioritised list of quality factors, we were able to guide our architectural and design decisions throughout the project. We prioritised the quality factors in the following way:

Very Important Quality factors having a large influence on our product or is a large focus in the project.

Important Quality factors important for our project with some limitations to the extent they influence the end product.

Less Important Quality factors which we do not focus on, but still take into account to the extent of what is a good practice. These will not be used for argumentation of choices made throughout the report.

Irrelevant Quality factors which are not focused on. These are not seen as factors the resulting product should reflect.

Our list of quality factors is based on van Vliet [2008], and the quality factors scalability, availability and durability. The definitions of the quality factors are listed in Appendix E. Furthermore, we grouped the quality factors into the following groups, as defined by van Vliet [2008]:

Product operation This deals with factors regarding the daily use of the product.

Product revision This regards the evolution of the product

Product transition This regards possible redeployment of the product on a new platform.

Table 3.1 lists our prioritisation of the quality factors.

Quality Factor	Very Important	Important	Less Important	Irrelevant
Product Operations				
Correctness	✓			
Usability	✓			
Reliability		✓		
Efficiency		✓		
Durability		✓		
Integrity			✓	
Availability				✓
Product Revision				
Flexibility	✓			
Maintainability			✓	
Testability			✓	
Scalability				✓
Product Transition				
Reusability		✓		
Portability				✓
Interoperability				✓

Table 3.1: Prioritisation of the quality factors.

The following argues for the prioritisation of the factors shown in the table.

Very Important

Correctness We chose correctness because the final system must satisfy the requirements. This is needed, because these dictate what the system must include to be applicable for a company that wants to apply automatic clocking.

Flexibility The non-functional requirements dictate that it must be possible to customise the system, using plug-ins, in order to be applied in different contexts. Therefore, this is a major concern when developing the system.

Usability This quality factor must be taken into account to make the replacement of the manual clocking system a success, we wanted Easy Clocking to not be more complicated than the originally.

Important

Reliability This quality factor was important in various aspects of the system. First of all, it was important that the system correctly determined which workstation a given employee was at. If this factor was not upheld, it could have consequences for the employees if the system reported them as being at another workstation than in reality.

Efficiency This factor was to some extent important in the system. Specifically, the importance of efficiency was significant in relation to the location sensors because these are based on hardware with limited computational and memory resources. In relation to the web application part of the system, efficiency was a minor concern.

Durability This quality factor was important in relation to the data generated by the location sensors. This data is fundamental for the system to operate in terms of determining where employees are located. If the system was unable to determine the locations because of data loss, it could influence how the employees were paid, for instance. Hence, mechanisms were deployed to reduce the probability of data loss.

Reusability This quality factor was considered important because the system must be deployable in different companies either as part of another system or as an independent application.

Less Important

The quality factors with this prioritisation were not focused on. We did, for instance, not want to focus on testability, nor did we ignore it. This meant that we only wanted to make the system testable to the extent that we could show that the final system fulfilled the requirements.

Irrelevant

We did not put any effort in the factors given the irrelevant priority. Some of them, such as scalability, were ignored due to the scope of the project. Regarding scalability the requirements define that the system must be able to handle location data generated by 300 butchers which we consider manageable without taking scalability into account. In relation to the web application, we did not expect it to be frequently accessed and handle a great number of concurrent users.

Part I

Analysis

4 Localisation Technologies

In order to do automatic clocking, we need to know which workstation the butcher is located at. To accomplish this, a localisation concept is used. First, we describe our choice of localisation concept, location estimation technique and finally localisation technology using a set of criteria.

4.1 Localisation Concept

Before choosing a specific technology to be used for localisation of employees, it is important to know which general localisation concept to use, since it affects the choice of technology. For instance, the chosen concept may require that the location sensors have significant low power consumption. During the analysis of which concept to use, we found two; Real Time Location Systems (RTLS)[Malik, 2009] and Wireless Sensor Networks (WSN)[Romer and Mattern, 2004].

RTLS is used for real time localisation of equipment or people. This is generally achieved by attaching small electronic tags on the equipment to be located. The tags are read by location sensors which are placed at known positions such that they cover the area of interest. When the location sensor registers a tag, this information is passed to a location engine. The location engine then uses the information gathered from one or more location sensors to estimate a location of the tag. Finally, this information is used by application software, e.g. to display the location to a user of the application. The system requires that the location sensors are always connected to the location engine such that they can send the raw localisation data instantly when obtained.[Malik, 2009]

WSN are, as the name implies, a network of wireless sensors. The sensors does not make use of a persistent power source, for instance, because they are randomly placed in impassable terrain. Because of this, the sensors in WSN are resource constrained, and power usage is therefore a big concern in such networks. Examples of where a WSN is used are for environmental monitoring where researchers monitor climatic changes and for monitoring vibrations of buildings. In both examples, the sensors are used in a wide-area and therefore, the sensors are configured to use an ad-hoc approach in order to get reports from the sensors far away to the location engine.[Romer and Mattern, 2004][David Culler, 2004]

In order to illustrate the different advantages of RTLS and WSN, Table 4.1 summarises these with respect to the project.

Advantage	RTLS	WSN
Detect movement at workstation	✓	✓
Indoor usage	✓	
Real-Time Tracking	✓	
Tag identification	✓	
Relatively few nodes	✓	
Robustness		✓
Cooperating nodes		✓

Table 4.1: Comparison of RTLS and WSN.

As shown in the table, RTLS has the most advantages related to this project, compared to WSN. Therefore, we chose to use RTLS as localisation concept. This choice conflicts with the requirement stating that the nodes should be able to be placed in a cooperating ad-hoc network. Since RTLS fulfils the other requirements, we chose to implement the ad-hoc network structure into RTLS in order to accommodate this remaining requirement.

To realise RTLS in the project, the butchers will be equipped with a tag which then can be located by location sensors at each workstation. Doing this would make the project similar to the Active Badge Location System, one of our role models, described in Appendix G.

4.1.1 Location Estimation Techniques

Ranging and positioning are two groups of techniques used to determine the position of tags.

There exist several techniques that can be used for localisation using ranging or positioning. In the following, we have described different techniques which are applicable to our project. This also is part of the fulfilment of our learning goal regarding location techniques.

The techniques described are introduced by Malik [2009], and have been further supported by the material from Golden and Bateman [2007], Ciurana et al. [2007], [Lionel M. Ni and Patil, 2004] and [Feldmann et al., 2003] all of which were read as part of this report.

4.1.2 Ranging Techniques

Ranging is the concept of estimating the distance between a tag and location sensors. Various techniques are used such as signal strength and time based calculations on signal propagation.

Time of Arrival(TOA) By measuring the time it takes for a signal to travel from a location sensor to a tag, it is possible to estimate the distance between the two devices. This can be done since the propagation speed for the wireless communication is known. Hence, by multiplying the measured time with the propagation speed gives the distance. This requires the clock of the location sensor and tag to be accurately synchronised since they must be compared. Note that it is also possible to measure the total time for the signal to propagate to the tag and back again, as shown in Figure 4.1.

Time Difference of Arrival(TDOA) This technique is similar to TOA. However, TDOA uses multiple location sensors to do the measurement. The time when a signal from a tag is received is stored by multiple location sensors and by measuring the difference in the arrival times it is possible to make a more precise estimate of where the tag is placed. This technique requires the clocks of the location sensors to be synchronised.

Received Signal Strength Indicator(RSSI) When a signal leaves its source, it has a specific strength. As the signal propagates, this strength decreases with a known relationship between distance and signal strength. It is therefore possible to estimate the distance based on the RSSI. A problem with this technique is that the signal strength is affected by obstacles, temperature, humidity etc.

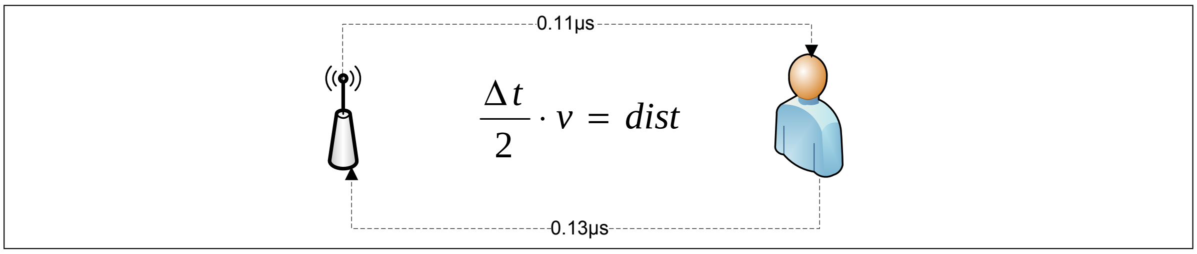
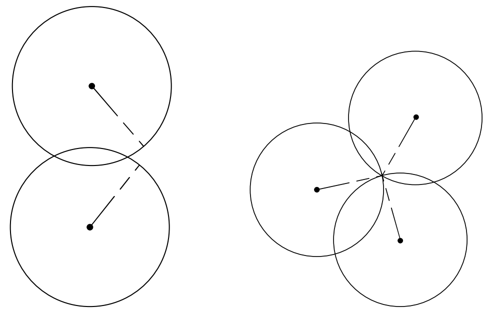


Figure 4.1: Time Of Arrival using the round trip time as time measurement. t is the time, v is the propagation speed and *dist* is the distance between the location sensor and the tag.

4.1.3 Positioning Techniques

Positioning is the concept of determining the position of a tag either absolute or relative to one or more location sensors. Position estimation algorithms, such as triangulation and trilateration, use data from multiple location sensors in conjunction with ranging techniques in order to determine the position.

Trilateration The position of a point can be determined by knowing the distance to three or more known points. By finding the intersection between the diameters of the known points, the searched point can be determined. Figure 4.2(a) shows an example of two known points, which are able to determine two possible points, and Figure 4.2(b) shows three known points determining a single possible point. The distances are calculated using ranging techniques.



(a) Two sensors pinpointing two possible positions.

(b) Three sensors pinpointing a single position.

Figure 4.2: Trilateration technique used for positioning.

Triangulation In triangulation, a position is determined by knowing the angle a given signal travels from three or more points in relation to a common reference line. This technique is shown in Figure 4.3 where a point is determined by drawing three lines from three known points.

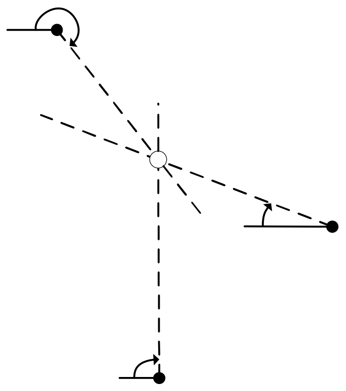


Figure 4.3: Triangulation technique used for positioning.

Nearest Neighbour It is possible to determine the position of a point by detecting nearby known points. Ranging techniques are used in order to determine which points are close.

4.1.4 Choosing Ranging or Positioning

If ranging is used, a sensor can be placed on each workstation and configured to detect and report tags within the range of interest, thus allowing us to detect if a butcher is close to a given workstation. Additionally, the placement of the sensors on the workstations opens up new possibilities. One could imagine that the sensor is able to communicate with the workstation such that the workstation could be turned off if the respective butcher leaves it. For instance, this could be appropriate if a given workstation requires certified butchers because the work at the workstation is dangerous. This approach thus requires a sensor for each workstation.

If positioning is used, the location of butchers can be determined more precisely and the workstation areas can be defined in more detail, in contrast to defining the workstation area to be the perimeter around a sensor. This solution requires the sensors being placed strategically in either each room or building, depending on the chosen technology, such that all areas are covered.

Both ranging and positioning techniques are usable in our scenario. Depending on the density of workstations, they require different amounts of hardware from scenario to scenario. If the workstations are placed in a relatively small area, positioning is capable of covering the room by placing three sensors and using trilateration. However, if the workstations are placed wide apart in a relatively large area, using positioning would require relatively much hardware to cover the area, whereas ranging would only require one location sensor at each workstation. Also, we consider the effort required to implement positioning higher than implementing ranging.

Based on this analysis of ranging and positioning, we have decided to use ranging techniques in our project. The primary argument for this decision is the simplicity of the ranging approach, where positioning requires multiple sensors and relatively complex calculations are needed for localisation determination.

We are aware that ranging potentially requires more equipment in larger deployments than using positioning, but we have weighted simplicity over equipment usage. Using positioning would have been the preferred solution if we had been more interested in tracking the butchers in the sense of determining their precise location but our requirement is only to determine symbolic positions.

In our analysis of ranging techniques, we identified three techniques to be use for localisation. The TOA and TDOA techniques use time measurements and the RSSI technique uses signal strengths in order to determine distance.

We chose not to use the time-based techniques, since we found multiple projects discarding these based on the time precision requirement[Thapa and Case, 2003][Kotanen et al., 2003] while other projects using these methods utilised specialised hardware[Fischer et al., 2004]. The problem with time-based techniques is that the delay between the receiver receives a request and send the response, is non-deterministic and can result in a significant calculation error in the distance if used for a time-based ranging technique.[Golden and Bateman, 2007] Because of this, and since we identified several projects describing their success in using RSSI, we chose to use it.[Papamanthou et al., 2008][Helen et al., 2009].

4.2 Technologies

We have used a set of evaluation criteria for choosing localisation technology. The criteria were based on previously determined requirements and considerations for the butchery.

The criteria, which are described in more detail in Appendix D dictate that the technology must:

- Work in an indoor environment with electrical noise.
- Not require recharging during a work day.
- Give the possibility of localising butchers in a ten meters range.
- Have a localisation accuracy of five meters.
- Not cost more than 1000 DK/Kr per workstation and 200 DK/Kr per tag.
- The technology must be reliable.

There exists a number of different technologies used in RTLS, each having their advantages and disadvantages. Malik [2009] mentions Wi-Fi, Bluetooth, RFID, Ultra Wide Band (UWB), ZigBee, Infrared and GPS as possible technologies for RTLS.

Early in the analysis, we excluded UWB, ZigBee, Infrared, and GPS. UWB and ZigBee both lacked a substantial user base and available consumer products. In the case of UWB, current governmental regulations prevent the technology from being used at distances larger than a few meters. This is in order to prevent interference with other radio communication products, thus not satisfying our requirements regarding detection distance.[Commission, 2007] GPS was discarded since it can not be used indoors, again not satisfying one of our requirements.[Malik, 2009] Infrared was discarded on its requirement of line-of-sight between tags and sensor. This can be a problem because the butchery contains lots of industrial equipment preventing that line-of-sight can be upheld.

We will in the following describe the remaining candidates: Wi-Fi, Bluetooth, and RFID. Later, in Section 4.3, we describe the choice of which technology to use.

4.2.1 Wi-Fi

Wi-Fi, or 802.11 wireless LANs, can be used to determine ranges using ranging techniques. This technology is interesting, in the sense that many companies and institutions already have a Wi-Fi infrastructure in place and Wi-Fi enabled hand-held devices, such as smartphones, are widespread.

If Wi-Fi was used, a device should be carried by the butchers in order to determine their location in relation to workstations. These would then be able to detect Wi-Fi access points using beacons. The access points would also be capable of detecting the devices based on probe requests transmitted from the devices.[Malik, 2009] Furthermore, Wi-Fi devices can be distinguished by their hardware address, making it possible to link this with a butcher.

The following describes advantages and disadvantages of Wi-Fi in respect to our evaluation criteria:

Interference Wi-Fi is capable of propagating signals in both in- and outdoor environments. The signals are affected by environmental changes such as humidity, temperature, re-arrangement of furniture and people moving around in the environment.[Malik, 2009] This adds considerable challenges when working with signal strength based solutions.

Interference from other devices depends on the chosen Wi-Fi standard. 802.11 b/g clients both operate in the 2.4 GHz spectrum, shared by technologies such as Bluetooth, while 802.11 a/n clients work in the 5 GHz spectrum which are affected by much less interference from other devices.[Xirrus, 2007]

Power Consumption Sharkey [2009] describes the power usage of different Google Android based mobile phones, and their power consumption under different modes. He states that an idling system draws 5 mA and a system using Wi-Fi (watching youtube) draws 340 mA. According to the criteria, we need a battery lifetime of eight hours corresponding to a work day. A HTC Magic has a battery capacity of 1350 mAh meaning that if it is under load drawing 340 mA, the battery will last for $1350mAh/340mA \approx 4h$. From this result, we can conclude that the system must be idling for over half of the time in order to meet the criteria of eight hours battery lifetime. We do not expect this would influence localisation because data would not need to be exchanged constantly but instead only periodically. Also, taking into account this calculation is based on watching YouTube, we expect to have lower power consumption when using Wi-Fi for localisation.

Distance The indoor connection range depends heavily on the interior of the building and antenna construction[Stein, 2000]. Other projects have shown connection ranges in the order of 30 meters[James F. Kurose, 2007, c. 6] to be possible, hence, this satisfies our requirement.

Accuracy The accuracy of ranging depends on the method and algorithms employed. As an example it has been shown that an accuracy of less than 1 meter is possible for ranges up to 30 meters using the TOA technique.[Ciurana et al., 2007]

Price Wireless access points in essence are small computers with Wi-Fi capabilities, allowing us to select consumer hardware for the workstations. These access points have costs in the vicinity of 700 DK/Kr. Most smart phones are Wi-Fi enabled and can therefore be used as tags. However, these are considered too expensive for our needs, but one could imagine that special purpose build hardware, only providing the needs of being tracked, could reduce the costs considerably.

Reliability The reliability of the Wi-Fi network depends on the load on the network and possible interference resulting in lost connections. For localisation we do not consider the network load as a problem since only little information must be transfered.

4.2.2 Bluetooth

This technology is a short-range wireless standard for electronic devices using radio frequencies. It is used in a variety of electrical appliances such as mobile phones. Bluetooth is designed to operate at low power in a Personal Area Network, making it popular for communication for remote devices. The newest version of Bluetooth as of September 2009, is called version 3.0 and offers a transfer speed of 24 Mbps whereas its predecessors, version 1.2 and 2.0 offers 1 Mbps and 3 Mbps, respectively. Since we have bounded this project to not include transmitting data to the butchers other than the data needed for localisation and identification, Bluetooth 1.2 or 2.0 are considered sufficient.[BluetoothSIG, 2009]

Devices with Bluetooth enabled are discoverable by other devices with Bluetooth. This make them appropriate for tags. Furthermore, Bluetooth devices can be distinguished by their hardware address, making it possible to link a hardware address with a butcher.

Using the ranging techniques with Bluetooth, can be realised similarly as Wi-Fi.[Malik, 2009]

In respect to the technology criteria, we have concluded following:

Interference Bluetooth uses Frequency Hopping Spread Spectrum, meaning that the devices have 79 different frequencies, within the assigned range, which can be chosen randomly. This reduces the interference between multiple devices. Notice that Bluetooth and Wi-Fi will interfere since they use the same spectrum.[Malik, 2009]

Power Consumption By experiments with a mobile phone with enabled Bluetooth, we have verified that recharging during a work day is not necessary. Also, as stated, Bluetooth is designed for low power consumption.[BluetoothSIG, 2009]

Distance Bluetooth devices are categorised in one of three power classes providing ranges from 1 to 100 meters. A class 2 Bluetooth device has a range of 10 meter which is required for this project.[BluetoothSIG, 2009]

Accuracy [Malik, 2009, c. 11] describes the typical accuracy for Bluetooth to 2 meter. This accuracy is a guideline and depends on the used localisation techniques.[Malik, 2009]

Price For the workstations, we have found hardware that costs 700 DK/Kr. We were not able to find a Bluetooth device which purpose was to be used as a tag. However, any device which has Bluetooth can be used and the cheapest we found costs 200 DK/Kr. If the butchers already have been given a work phone which has Bluetooth, this can be used as tag.[mitEDB]

Reliability BluetoothSIG [2009] describes Bluetooth as a robust technology, meaning that it can be considered reliable.

4.2.3 RFID

RFID, or Radio-Frequency Identification, is a wireless technology that can be used for identifying and tracking tags placed on objects. The tags consist of a small chip attached to a radio antenna. The chip stores information such as a unique ID which can be read from a device called a reader. The reader has antennas which emit radio waves and receive signals back from the tags.

RFID operates in various frequency ranges and are classified as low-frequency (LF), high-frequency (HF) and ultra high frequency (UHF). The read distance of the tag is dependent on which frequency range they use. Furthermore, tags can be passive, semi-passive and active which also affect the read distance. These are described below:

Passive tags These tags do not use an internal power source to transmit signals to the reader. Instead, they communicate with the reader by a method called back scattering where the signal from the reader is reflected by modulating the signal to transmit data. However, the passive tags only back scatter about 10-15 percent of the received signal, thereby significantly reducing the distance to which the tag can be read to around a few inches or feet.

Semi-passive tags These works similarly as the passive tags by using the back scattering method. However, they contain a small battery which allows the circuit of the tag to be constantly powered. Hence, there is no need for using the received signal from the reader to power the tag which increases the reading distance compared to that of passive tags.

Active tags These contain an internal power source, but in contrast to semi-passive tags, they broadcast their own signal, often in predefined intervals. Active tags have practical ranges of tens of meters. One should be aware that semi-passive and active tags requires maintenance because of their battery usage.[Malik, 2009]

If RTLS is to be realised with RFID for location determination, active tags would be the appropriate variant to use mainly because of the relatively high reading distance. In our scenario, RFID readers would be placed on each workstation and every butcher should carry their personal active RFID tag. The active RFID readers take measurement according to one of the previously described ranging methods and forward this data to the location engine which computes the distance from the reader to the tag.

In respect to the technology criteria we have concluded following:

Interference For the various frequencies LF, HF, and UHF there are different concerns one should be aware of. LF has a severely low read rate in noisy environments, such as being situated close to power transformers. UHF has a reduced performance in environments containing liquids and metals. HF works well in environments with electrical noise similar to the noise mentioned for LF.[Malik, 2009]

Power Consumption The power consumption depends on the hardware, but we have found an active tag capable of running for months. Passive tags do not contain batteries meaning that recharging is unnecessary.[OpenPCD-Shop]

Distance The reading distance depends on which type of RFID tags is used. For the hardware we have found, the indoor reading distance for the active tags is 25 meters and 10 to 50 centimetres for a passive tag.[OpenPCD-Shop][RFID-specialisten]

Accuracy [Malik, 2009, c. 11] describes the typical accuracy for RFID using passive tags to one meter and using active tags one to three meters. This accuracy is a guideline and depends on the localisation techniques used.

Price For each workstation, the hardware costs 1200 DK/Kr and 190 DK/Kr for an active tag.[OpenPCD-Shop][mitEDB]

Reliability All RFID tags can be read despite extreme environmental factors, such as snow, fog, ice, paint, and other visually and environmentally challenging conditions. Furthermore, Hiltunen [2007] describes that reading the tags can be considered reliable.

4.3 Selection of Technology

To ease the task of choosing the appropriate technology, we chose to compare Wi-Fi, Bluetooth and RFID in Table 4.2 with respect to the criteria. For each criteria, we gave a score of 0-3, where 0 was used when the criteria was not satisfied and 3 when the criteria was more than satisfied.

Criteria	Wi-Fi	Bluetooth	RFID
Interference	2	3	2
Power Consumption	1	2	3
Distance	2	3	2
Accuracy	3	3	3
Price	1*	3	2
Reliability	2	3	3
Total	11	17	15

*The score assumes that we cannot obtain a special purpose tag.

Table 4.2: Overview of the different analysed technologies and how they compare based on our evaluation criteria.

On the basis of our analysis, we concluded that all of the three technologies were suitable. Wi-Fi was the lowest scoring technology, outnumbered by Bluetooth on several criteria.

RFID was a good candidate as our choice of technology. Unfortunately, both the price of available equipment is high, and the equipment within our price-range lacked documentation and was on an experimental stage to the extent that we were not entirely convinced that the hardware would be functional.

Then, the choice had to be made between Wi-Fi and Bluetooth. Generally, Bluetooth seemed as a better solution for this project based on the scores. Therefore, we chose using this technology for localisation.

5 Extensible Architecture using Plug-ins

In this chapter, we will document how plug-ins can be used to create more extensible software, and how a plug-in architecture was incorporated in the web application.

As stated in our non-functional requirements in Section 3.2, the web application should be flexible by extending its capabilities with plug-ins. The motivation was to allow different companies to extend Easy Clocking in order to support their business logic and work flows, e.g. by adding a salary payment plug-in.

5.1 Plug-in Methods

First, it is important to note that there exists a number of methods which can be used to improve the extensibility of software without a plug-in architecture. Examples of alternatives include packaging parts of the software in reusable components of code and using Aspect Oriented Programming. In this paradigm, the primary goal is to avoid scattered code such as a logging functionality being used in multiple unrelated methods. This is achieved by isolating the logging functionality in its own module and additionally specify where the module should be used in the code. In a similar way, one could imagine, that the plug-in code could be placed in its own module and specify where the code should be executed with the use of the aspect oriented construct called advices.[Filman et al., 2004]

We chose to focus on plug-in architectures since we have never previously worked with these. In general, plug-ins are used to extend or change behaviour in software by using extension points. An extension point defines a part of the underlying software where new functionality can be added by the plug-ins[Birsan, 2005]. An extension point can be implemented using e.g. callback functions or signals. Stylos [2009] gives a number of examples of applications using plug-ins to extend their functionality, such as the mp3-player Winamp allowing skins to change its appearance and the web-browser Firefox supporting additional video formats.

Birsan [2005] divides plug-in architectures into two groups: traditional and pure plug-in architectures. Traditional plug-in architectures consist of host applications which are extended by plug-ins. This is known from applications such as the Firefox and Winamp examples given above. Pure plug-in architectures consist of a small plug-in engine dividing all functionality into plug-ins which are combined by the plug-in engine in order to assemble the application. In the pure plug-in architecture, the plug-ins are capable of extending other plug-ins or providing functionality for other plug-ins to build upon, while the traditional architecture only allows isolated plug-ins.

By using a pure plug-in architecture, it is possible to create new applications or reconfigure applications by changing the configuration of chosen plug-ins. This approach is suitable when the application is to be changed drastically[Birsan, 2005]. However, this is at the cost of a more complex architecture, since a large set of dependencies between the plug-ins may potentially be created with the pure plug-in architecture. This is in contrast to an application built upon the traditional plug-in architecture with few but well defined isolated plug-ins.

Using the pure plug-in architecture makes it possible to add new plug-ins or replace existing plug-ins in order to add new functionality or alter existing functionality. If a traditional architecture was used, it would be necessary to know where possible companies would like to extend our application and foresee the functionality they would like to add. As an example, some possible uses of Easy Clocking are calculation of employee salary and measurement and monitoring of employee work habits. In some systems, Easy Clocking would become a small component of a larger system, made possible by structuring the Easy Clocking system as a plug-in and designing it such that it can be combined with other plug-ins.

On the basis of the above, the pure plug-in architecture would be most appropriate for Easy Clocking, since we do not know in which context the application is used and in which company.

6 Usability

In this chapter, we introduce the Gestalt theory, design principles, conceptual models, and interaction styles used in the field of usability and user experience. These were used in the project when implementing and evaluating our user interface for Easy Clocking. We also reflect upon which elements from the conceptual models and interaction styles we should use in our project, based on the actors described in Section 1.4.

Preece et al. [2002] defines usability and user experience as how well the user can use the system, and how aesthetically and rewarding the experience is, respectively. However, we have not emphasised on user experience because in our opinion this is not important for Easy Clocking.

6.1 Gestalt Theory

The Gestalt effect, which is described in Gestalt theory, is the capability of the human senses to perceive shapes and forms from a collection of simple objects. For instance, if a collection of objects not forming a coherent structure are arranged circularly, a human perceives the structure as a coherent circle. Generally, the phrase "The whole is greater than the parts" describes what Gestalt theory is about.[Kalbach, 2007]

Harnessing these human properties can be relevant when designing the layout of Web pages because the Gestalt laws, some of which are described briefly below, can be used to enhance the overview of information. Hence, the utilisation of the Gestalt laws can improve the usability of the web application.

Below a brief description is given for the Gestalt laws which may be relevant for Web page layout.[Kalbach, 2007][Stafford and Webb, 2004]

Proximity By clustering objects into groups such that the distances among themselves are the same, regions are created which suggest that objects in each region are related.

Closure If elements are organised approximately in a specific shape, the shape will be completed perceptually even though, in reality, the elements are not completely making up the shape.

Continuity If page elements are arranged in a continuous whole such as a line, it may appear that the line extends even though it stops.

Similarity If shapes contain similar properties, the viewer will associate these. For instance, if a subset of elements has the same colour, this suggests that they are associated.

Figure 6.1 depicts examples of each of the Gestalt laws. Combining the Gestalt laws, can further improve the usability.

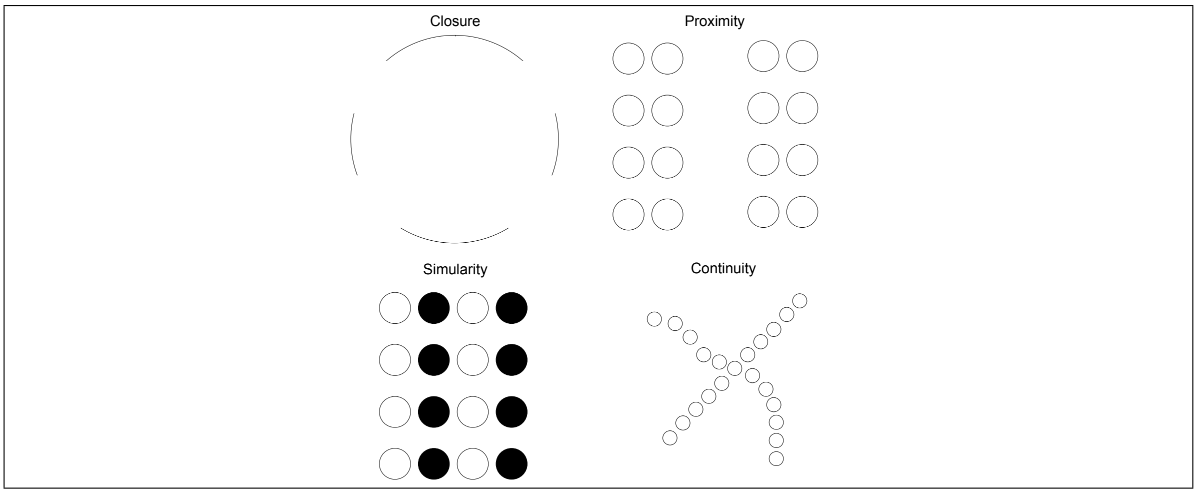


Figure 6.1: Examples of the Gestalt laws; closure, proximity, similarity and continuity.

6.2 Design Principles

When considering enhancing the usability of a user interface, there are some design principles one should take into account. These do not describe how the various elements on the user interface should look like but instead provide some overall considerations that are important regarding usability.

The design principles introduced in this section primarily concern how users see and interact with the user interface. Preece et al. [2002] lists the following design principles; Visibility, Feedback, Constraints, Mapping, Consistency and Affordance.

Visibility Visibility is the design principle concerning making clear what elements the user can interact with. One could imagine an image on a web page which contains areas that are clickable to produce actions. If these areas are not clearly visible, the user may have difficulties in interacting with the functionality on the web page. In addition to not being easily locatable, it may not be clear what action the clickable area provides.

Feedback Feedback concerns responding, in a reasonable amount of time, with information about what action has been performed including what has been accomplished. In some sense, Feedback also relates to Visibility. E.g feedback is not produced in the example with clicking on a particular area of an image, it is not clear to the user whether he has performed an action or not and, hence, located the area or not. Instead, feedback in the form of e.g. marking the particular area after being clicked and in addition producing a sound will act as confirmation that the area has successfully been clicked.

Constraints The Constraints design principle concerns restricting ways of interaction that are not related to the user's current activity. For instance, if user information is updated through the use of forms, it should not be possible to do the actual update before typing in new values in the fields.

Achieving the Constraints design principle can be done in various ways. A common approach, which can be used in the example introduced previously, is to disable the button responsible for performing the update functionality. A disabled button can appear greyed out and no action is provided when clicked.

Mapping The Mapping design principle is concerned with how controls and their effects are related. It is important that this seems intuitive to the user, because wrongly mapped controls and actions can have consequences. As an extreme example, a label with the text "Save" should not perform a delete action. Another aspect of the Mapping design principle refers to how controls are put relative to each other. If a page provides two links placed next to each other for going to the previous and next page, respectively, it would seem counter-intuitive if the link for going to the next page was placed to the left of the link for going to the previous page.

Consistency Consistency refers to designing the user interface in such a way that similar operations achieve similar results. By doing so, the interface will be easier to learn and use because a single operation performs the same result on various types of objects. For instance, one could imagine a specific icon that is consistently present at all listings of information. By clicking the icon, the entire list of information is removed no matter which listing the icon is clicked next to. The consistency design principle somehow also relates to convention. As an example, a button that is disabled, is conventionally greyed out.

Affordance Affordance is the design principle of visually indicating how a particular object is used. A physical button in the real world, affords the user to push it. However, on screen based interfaces, this kind of affordance is not obtainable because the interaction forms with all elements are more or less constrained to the actions that can be performed with the keyboard and the mouse. Therefore, on screen-based interfaces, the term *perceived affordance* is used, so a button presented on a web page can afford being pushed if the button is graphically presented with various elements to make it look like a real button.

6.3 Conceptual Model

The conceptual model concerns how a system is to be presented and how users are to interact with the system in order to support their tasks. First, it is necessary to determine the user's tasks and the steps needed in order to accomplish these. The following description of conceptual models is based on [Preece et al., 2002, c. 2] who states that the conceptual model can be modelled using two different views, namely, activities or objects.

6.3.1 Activities

Activities concern how the users are to interact with the system in order to accomplish their tasks. These can be used individually or combined in order to support an interaction. Common activities are:

Instructing The user instructs the system to perform actions. Examples of this are systems such as terminals in Unix where the user writes commands to perform an action or in a text editor which can be instructed to count the number of words in a document by pressing the item performing the word count action in the menu.

This activity supports quick and efficient interaction with a system and is suitable when these are conducted in a repetitive manner.

Conversing The user and system goes through a conversation, asking each other questions and giving each other answers. Help desks often have a form used to ask the system questions regarding problems or information, resulting in a number of answers or even new questions.

This activity is suitable when users are trying to uncover specific information through questions. One advantage of this activity is that it allows the user to formulate a question in a way that is natural to the user.

Manipulating and Navigating The user manipulates and navigates in an environment which functions as the real world. This is used in web browsers when displaying the submit button in forms. The button is rendered as if it was a real button sticking out of the page, which can be pressed in order to render a result.

This activity lets users quickly learn how to interact with the system and users using the system infrequently are more capable of remembering how to use the system. In general, this activity is more intuitive and lets users feel like they are in control.

Exploring and Browsing The user is encouraged to explore and browse information if it is laid out such that the user easily can navigate to the correct information. This is one of the corner stones of the World Wide Web, which allows users to navigate between pages in order to find the desired information.

Using this activity, users often scan through the information to find what they are searching for. However, it requires that some effort is put into determining the structure of the information.

6.3.2 Objects

When basing the conceptual model on objects, one tries to model the system around a number of real world objects which are known by the user. This allows the user to quickly recognise how to interact with the objects and understand what they represent.

An example of such a conceptual model is the one used in most spreadsheet applications today. A spreadsheet builds upon the ledger sheet which allows users working with these types of sheets to easily understand the concepts of spreadsheet programs and to migrate their work processes from the physical world to the digital. The benefits of modelling the system against real world objects are lost in case the user has no experience using these.[Preece et al., 2002]

6.3.3 Choice of Conceptual Model

In our scenario, there are two actors interacting directly with the system: managers and Easy Clocking consultants. Both of the actors interact with the system on a daily basis.

From the above, a user interface supporting *instruction* and *manipulating and navigating* would be suitable. Instruction would allow the users to quickly accomplish their tasks. *Manipulating and navigating* allows for a quicker learning curve of the system. This may seem unnecessary since the actors works with the system on a daily basis, but according to the personas, managers are not as proficient in computers and hence we anticipate that a more user friendly environment would be appreciated by them.

6.4 Interaction Styles

Interaction styles extend the activities from the conceptual model, and define how the activity is to be accomplished. This is done by defining how the interface is to be implemented, and the elements it consists of. Skov [2009] defines the following interaction styles;

Command Direct input of commands into a terminal supports the instruction interaction. This type of interaction style is targeted at expert users, supporting quick usage at the cost of a steep learning curve.

Menus Menus, such as pull-down menus in GUI applications or navigation menus on web pages, support the *instruction* and *exploring and browsing* activities. This style lets users quickly learn how to use the system, adding structure to the decision processes and reduce typing at the cost of screen space and the danger of overloading the user with menus.

Dialogues Dialogues navigate the user through a process, asking the user questions and making decisions based on the answers, such as the process a user goes through when withdrawing money from an ATM. This supports the conversation activity and allows many different users of different technical levels to use the system.

Forms Forms allow users to type in information into a system in a structured fashion, thus supporting the instruction and conversation activities. This style is best suited for frequent users since the individual fields in the form requires knowledge to understand. The style can be slow to use, but allows users with limited training to insert large amounts of information.

Windows, Icons, Menus and Pointers (WIMP) This interaction style is used as desktops in most modern operating systems such as Windows and Linux. Elements can be dragged and manipulated in order to accomplish the user's goals, supporting the manipulation and navigation activity. This style is easy to learn by new users and to remember by infrequent users.

6.4.1 Choice of Interaction Styles

The selected interaction styles depend on factors such as the types of users which should be supported, the amount of desired training, amount of needed information and most importantly the task which needs to be solved. A system can consist of multiple interaction styles to address different users of the system.

The managers need a more user friendly environment to work with. Even though they work with the system on a daily basis, the command interaction style would be difficult for them to learn. A combination of menus and forms would be more suitable, using menus to navigate, issue instructions and create some form of structure to the work flow. The form interaction style supports them in inserting information into the system and requires only a small amount of training.

For our scenario, the Easy Clocking consultants are not considered expert users according to the personas. Therefore, we do not assume that the command interaction style would be appropriate for them. Hence, wherever possible, the interaction styles used for the managers should be applied for the consultants as well.

Part II

Design

7 Technical Platform

This chapter identifies the technical aspects of the Easy Clocking system and prepares us for which languages and technologies will be used in the implementation. The chapter takes starting point in describing the technical aspects of the location sensors which comprise decribing the equipment at our disposal, the firmware, and the programming language. In this relation, we also describe the network topology used to connect the location sensors with the web application. Afterwards, we describe the technical aspects of the web application which mainly include arguing for the framework and programming language used.

Section 7.3 gives an overview of the choices and observations made in this chapter.

7.1 Location Sensor

This section identifies the technical requirements of the location sensor.

7.1.1 Hardware Specification

As described in Chapter 4, we chose to use Bluetooth as localisation technology. During the analysis of this technology, we identified hardware that could fulfil our needs. The hardware we acquired consisted of an Asus WL-500gP V2 wireless router and a Kensington K33902 Bluetooth dongle. The wireless router was chosen because of the possibility of deploying a custom firmware to extend its functionalities and because it had USB ports meaning that it was possible to connect a Bluetooth dongle to it. Since the location sensors are based on consumer-grade routers, this makes the Easy Clocking system a relatively inexpensive solution for companies to deploy.

Because we needed to develop software for the router, it was important to know the hardware specifications of the device since an embedded device like the router may pose constraints on its capabilities because of relatively few resources. The part of the hardware specification[DD-WRT, 2009] that concerned our development is shown in Table 7.1.

Note that the router has 32 MB of system memory and 8 MB flash memory available. The flash memory is used to store the firmware, applications and persistent data since it

Platform	Broadcom 5354 Chipset
CPU	MIPS32 CPU running at 240 MHz
Flash memory	8 MB NAND
System memory	32 MB 16-bit DDR SDRAM
USB ports	2 x USB 2.0
Wireless radio	Broadcom 802.11b/g
Network switch	4 $\times$ 10/100 Mbit LAN and 1 $\times$ 10/100 Mbit WAN

Table 7.1: Hardware specifications for the Asus WL-500gP V2 wireless router.

is maintained between reboots of the device, and the system memory is used for running applications.

Since flash memory is limited in the number of possible writes, excessive writes to it should be reduced. Depending on the used hardware, between 5.000 and 100.000 writes are possible on the same block of memory[Thatcher et al., 2009]. The type of hardware in our router is unknown since it has not been specified in its specification. Thus, the flash memory should only be used for storing applications and data that are rarely changed.

7.1.2 System Firmware

Since the Asus WL-500gP V2 router did not support Bluetooth devices and the possibility of installing additional programs by default, a new firmware was needed. There are several wireless router firmwares, giving us these features. Hence, in order to choose which firmware was most suitable, we defined certain criteria that the firmware must fulfil. These are described in the following.

Hardware Support We wanted the device to be officially supported by the firmware in order to reduce the amount of time needed to install the new firmware and get it into an operational state.

Documentation Documentation must be available on how to install the firmware with USB and Bluetooth support.

Ad-Hoc According to our requirements, it must be possible to connect the location sensors in an ad-hoc network. The firmware must therefore support this.

Execute Applications We want to make our own software to run on the location sensor for locating Bluetooth devices. Therefore, it must be possible to execute our own applications on the device.

In the analysis, we considered using Free-WRT, OpenWRT, Sveasoft, Tomato, DD-WRT and Oleg. Through a brief analysis, we excluded Free-WRT since it did not support the Asus router.[FreeWRT, 2009] Also, we excluded Tomato and Sveasoft because of no USB support and missing documentation, respectively.[polarcloud, 2009]

The remaining firmwares were DD-WRT, OpenWRT and Oleg which all are Linux based. The Oleg firmware is a modification of the factory firmware from Asus targeting Asus devices, whereas OpenWRT is a more generalised firmware supporting multiple wireless routers.[OpenWRT, 2009b] Finally, DD-WRT is a firmware based on OpenWRT[DD-WRT, 2009].

Through our analysis, we discovered that all firmwares fulfilled the criteria, and that there were no specific advantages in using the one over the others. Therefore, we decided to make the choice of firmware based on knowledge gained through experiments. We had two devices and could therefore try firmwares in parallel. We started with DD-WRT and Oleg. With DD-WRT, we had troubles of getting the device to register USB devices, and since the firmware is based on OpenWRT, the documentation linked to solutions for the OpenWRT firmware. Therefore, we decided to stop experimenting with DD-WRT and do experiments with OpenWRT instead.

When experimenting with OpenWRT, we discovered several tutorials describing how Bluetooth could be supported by the Asus router. This documentation seemed promising because several sources described that they had an Asus WL-500gP V2 supporting Bluetooth. Because of this, and since the experiments with Oleg showed that it would require relatively high effort to get Bluetooth working, we decided only to focus on OpenWRT.[Oleg, 2009] A side effect of excluding Oleg is that the location sensor's hardware is not constrained to Asus products.

Through several experiments, we were able to detect other Bluetooth devices using OpenWRT Kamikaze 8.09.1 with a Linux kernel 2.4.35[chalermlab, 2009].

Since the location sensors have limited resources, we chose to remove some of the utilities that were not needed from the firmware. We removed utilities such as, gzip, chgrp, diff and httpd. Even though, removing the utilities did not decrease the size of the firmware significantly, it is, in our opinion, still important to address the relatively limited resources of embedded devices.

The location sensors are installed and configured by the Easy Clocking consultants by following the procedure described in Appendix H.

7.1.3 Programming Language

In the beginning of this project, we had no experience in programming embedded devices. We therefore examined which programming language to use and found that Barr and Massa [2006] recommend C. They also mention C++, Forth and Ada as languages used by embedded programmers, but they all suffer from certain disadvantages. Forth is described as extremely low-level, C++ programs are not as efficient as C programs and Ada has not gained much foothold.[Barr and Massa, 2006, c. 1]

We decided to use C, which main advantage, in respect to embedded programming, is that it is a high-level language with the possibility of having a high degree of direct hardware control.[Barr and Massa, 2006, c. 1]

7.1.4 Build Environment

As described previously, the router we are using is based on the MIPS architecture. This means that when we had to develop the location software on our computers, which are based on the x86 architecture, we needed an environment for generating MIPS code. What was needed was a cross-compilation toolchain, consisting of software such as a compiler, a linker and an assembler, running on an x86 based host system and generating code for a MIPS target system. OpenWrt makes available this cross-compilation toolchain.[Open-Wrt, 2006]

7.1.5 Ad-Hoc Wireless Mesh Routing

According to our requirements, we needed to implement a mesh network to allow location sensors to communicate with the location engine even though they have no direct reachability to it. OpenWRT makes available OLSRD[1] as an ad-hoc mesh network routing daemon, whose purpose is to help routing between nodes.

OLSRD implements the Optimised Link State Routing Protocol (OLSR) protocol, which is a proactive routing protocol, meaning that it regularly exchanges topology information between the nodes in the network. When a node is added to the network, it selects neighbours as Multipoint Relays (MPR) such that these can announce to the their MPRs that they can reach the new node. When finding the path between a node and another destination in the network, the information from the MPRs is used.[NetworkWorkingGroup, 2009]

We chose to use OLSRD since OpenWRT supports it and has documentation describing the installation and configuration of it.[OpenWRT, 2009a].

7.2 Web Application

In this section, we introduce the hardware requirements, operating system and implementation language used for the web application part of the Easy Clocking system.

7.2.1 Equipment and Operating System

The equipment needed for hosting the web application was limited. The only requirement is access to the Intranet of the company and the capability of running a web server for hosting the web application. The web application will only be accessed relatively few times during a work day by the managers and only once in a while by the consultants, so the system will not

[1] `olsr.org`

have to deal with many concurrent requests and frequent access. Therefore, we assessed that a small computer with limited processing capacity would be sufficient.

We have chosen that the machine responsible for hosting the web application is based on Linux. Specifically, we have chosen the Ubuntu 9.04 distribution[2]. The choice of using Linux instead of some other operating system is based on the group having experience with administration, maintenance and web development on this platform. The choice of Ubuntu is affected by the group members' experience with this particular distribution.

7.2.2 Framework

According to our learning goals, we want to learn how to use a framework for developing the web application. Before analysing which frameworks would be appropriate, we give a brief description of the opportunities a web application framework provides.

A framework can consist of a set of programs, libraries and concepts which purpose is to aid the developer in constructing web applications on top of the framework. For instance, web applications often make use of a database but instead of leaving the responsibility to the developer of writing a translator for mapping methods in the programming language to interactions with the database, the framework often provides these. Furthermore, a web application framework often provides a unified API for database access. Therefore, in case the database backend is substituted, the developer does not need to make changes in the code. If not using this functionality of a framework, this interaction between the application and the database could among others be constructed by stored procedures. However, the implementation of stored procedures varies from one database system to another and hence they do not provide a unified API for accessing different database systems.[Black, 2006]

Many frameworks also contain a bridge between the tuples of the relational database backend and the classes, instances and method-calls of the programming language. This kind of mapping is called Object/Relational Mapping (ORM) and lets the developers focus on a higher level of abstraction because interaction with the underlying database is done implicitly by manipulating the objects in the programming language.[Black, 2006]

Some frameworks also support various architectural patterns such as MVC, which can be utilised for e.g. enhancing the separation of concerns which provides a good overview of the application.[Fowler, 2002]

Besides the previously described features, frameworks often provide a variety of functionalities such as automatically generated tests. However, the general idea of using a framework is that many commonly needed features, such as database communication, are provided by default and only a minimal effort is required by the developer to make use of them. Additionally, frameworks for web development often automatically generate the appropriate HTML form elements given some functionality described in the particular programming language.[Black, 2006]

[2]`ubuntu.com`

Selection Ruby on Rails

As will be described in Chapter 8 concerning the architectural design of the system, we chose to use the Model View Controller (MVC) architectural pattern so the chosen framework needed to support this.

There exist many frameworks supporting the MVC pattern. Hence, in order to reduce the number of candidates, we have focused on frameworks available for some of the most popular programming languages as of September 2009[TIOBE-Software, 2009]. Some of these are: Java, PHP, Python, Ruby, and C#. Since we chose to use Linux as platform, we excluded C#, and thereby the ASP.NET MVC framework, as a candidate. For each of the remaining, an example of a framework supporting MVC is given below.

- Apache Struts Framework[3] for Java.
- CakePHP[4]for PHP.
- Django[5] for Python.
- Ruby on Rails[6] for Ruby.

When choosing the framework, we had some requirements. As stated in the learning goals, we wanted to gain knowledge of using a framework and also a new programming language. Therefore, we wanted a framework which uses a programming language that we are not acquainted with. This decision narrowed down the potential candidates to Ruby with the Ruby on Rails framework since we had experience with the remaining languages. Furthermore, we found the philosophies and principles emphasised in Ruby on Rails interesting. In the following, we briefly describe these[Thomas et al., 2006].

DRY: Don't Repeat Yourself This principle means that code should never be duplicated in a Rails application. This means that when making modifications to the code, the programmer does not have to make changes numerous places.

CoC: Convention over Configuration CoC is the design philosophy of decreasing the number of decisions that need to be made by the developer. The general achievement behind this philosophy is simplicity in the form of removing configuration which may need to be done manually by instead following a predefined set of conventions. Simplicity should not be at the cost of reduced flexibility which is accommodated by introducing configuration if one needs to deviate from the conventions. As an example, by convention, Ruby on Rails assumes that for a *customer* model there exists a corresponding *customers* table in the database.

[3] `struts.apache.org`
[4] `cakephp.org`
[5] `djangoproject.com`
[6] `rubyonrails.org`

Agile Ruby on Rails is said to be agile since it meets the agile manifesto. It is e.g. possible to have working code early in the process, making it possible for a customer to give feedback. Through this feedback, it is possible to accommodate the customer's needs, and the changes can be made quickly, because of the DRY principle, among others.

Fast cycle The modification cycle in a compiled language is, modify-compile-run and this cycle typically takes places hundreds of times a day. Therefore Ruby, which is an interpreted language, has an advantage in the development phase, since its modification cycle is modify-run. This makes the development of an application faster than using a compiled language.[Burd, 2007]

POLS: Principle of Least Surprise The creator of Ruby wanted to have fun when programming in Ruby. In Ruby's community this has been called POLS since the programmers believe it is fun to program when not surprised by new curiosities of the language.[Venners, 2003]

In addition, Ruby on Rails allows for quick construction of web services such as RESTful interfaces and supports rapid prototyping through the concept of scaffolding which can auto generate parts of the application such as controllers and views used for editing the models.

Ruby on Rails is, as the name implies, build on the Ruby programming language. Ruby is an interpreted scripting language supporting multiple programming paradigms and emphasises on being portable, which means the same Ruby program will run on a wide distribution of platforms including Windows and Unix. Additionally, Ruby is an object oriented language but it also supports functional, imperative and reflective programming styles and combines a syntax inspired by Perl with Smalltalk-like features.[David Flanagan, 2008]

For developing the web application, we have used Ruby version 1.8.7 with the Ruby on Rails framework version 2.3.4.

7.2.3 Web Server

The choice of web server for hosting the Ruby on Rails web application was not based on specific requirements. The web server did not need to contain scalability features and load balancing because, as stated, we did not expect the web application to be used frequently and handle many concurrent users.

There are numerous web servers that offer the possibility of hosting Ruby on Rails web applications. According to a survey conducted by Netcraft[Netcraft, 2009] showing the web server usage shares as of September 2009, some of the most popular web servers supporting the Ruby on Rails framework are Apache and Lighttpd. However, these require a more complex configuration in order to support Ruby on Rails.

A default installation of Ruby on Rails comes bundled with an HTTP server library called WEBrick. WEBrick offers the possibility of hosting a Ruby on Rails application with a minimum of configuration. Because hosting the Ruby on Rails application is our only requirement of a web server, we chose WEBrick.

7.3 Overview

An overview of the choices and observations made in this chapter is shown in Table 7.2 and is provided because many decisions have been made throughout this chapter.

Location sensor hardware	Asus WL-500gP V2 and Kensington K33902
Location sensor firmware	OpenWRT Kamikaze 8.09.1
Mesh Routing	OLSRD v. 0.5.6
Location sensor programming language	C
Location sensor architecture	MIPS
Web application programming language	Ruby v. 2.3.4
Web application framework	Ruby on Rails v. 1.8.7

Table 7.2: Overview of the decisions made in this chapter.

8 System Architecture

The purpose of this chapter is to describe the system architecture of Easy Clocking. A system architecture helps ease the understanding the structure of software.[Design, 2000, c. 1] During the development process, we iteratively updated the architecture, which helped us to keep an updated view of the structure of our components and more easily make large structural changes to the design.

In the following, we introduce the architecture consisting of the two primary components: the web application and the location sensors. The communication between these will be described in the following section and afterwards we describe their responsibilities and parts. These two components, and how they communicate, were identified on the basis of our analysis of the problem and application domain, as described in Section 1.3.

The web application is based on the architectural design pattern MVC, a section is dedicated for describing this and how it influences the architecture.

Finally, because the architecture must reflect the flexibility quality criteria, a section is dedicated for describing how this is realised with the use of Ruby on Rails.

The architecture of the Easy Clocking system is depicted in Figure 8.1.

8.1 Communication Protocols

From Figure 8.1, we identified the need for communication between the web application and the location sensors. To realise this, we analysed which data should be communicated. In Figure 1.2, Section 1.3, we divided Easy Clocking into three parts: the web application, conflict handler, and the location sensors. From this, we observed that information exchanged between the location sensor and the conflict handler was location sensor configurations and clocking information. The configurations are only exchanged between the web application and location sensor, whereas the clocking information from the location sensors are transmitted to the conflict handler. Since the conflict handler is implemented in the web application they will not be described separately.

Exchanging data between the web application and the location sensors required network communication, since they are physically separate. Before choosing the communication

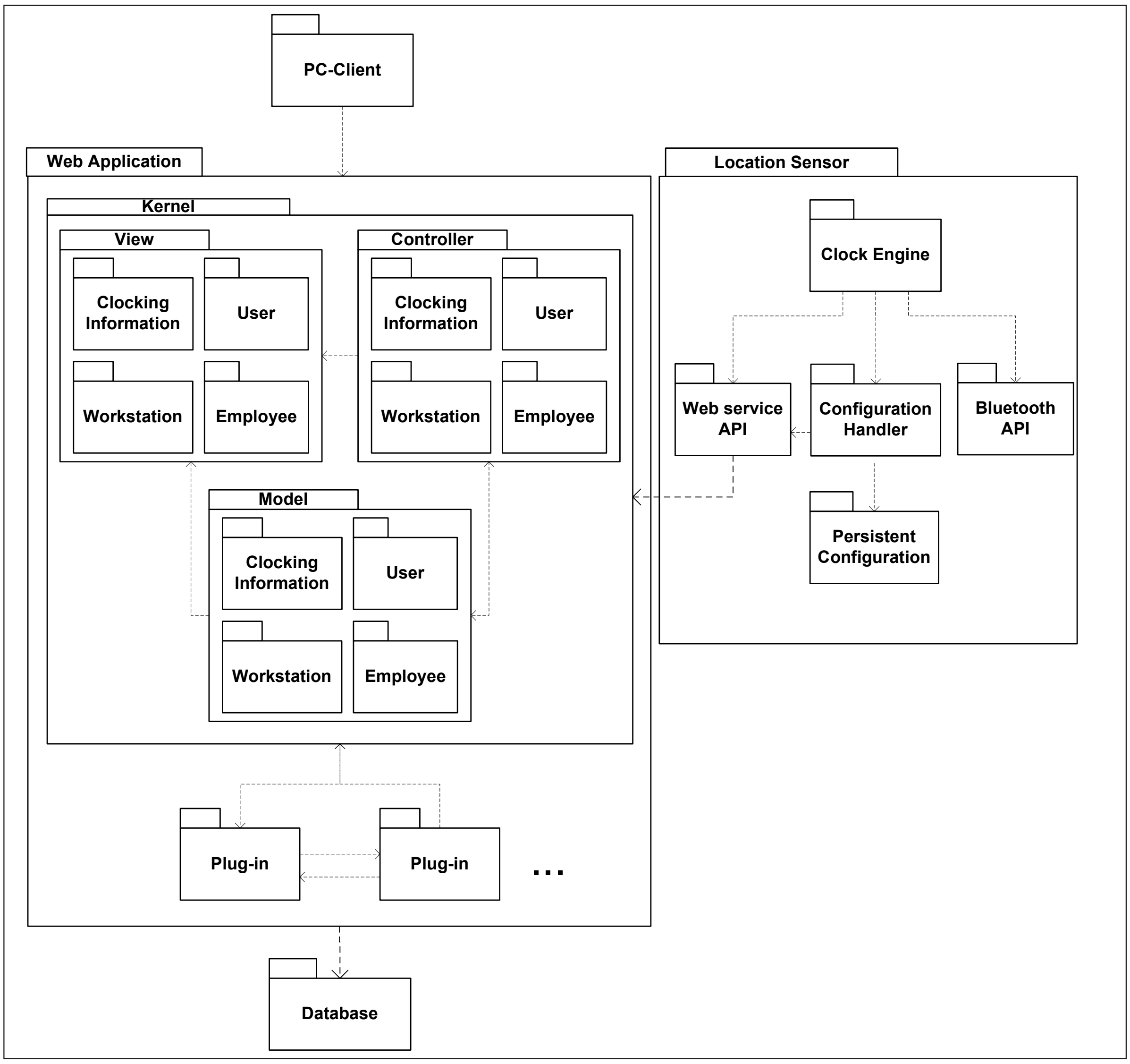


Figure 8.1: Package diagram showing the architecture of Easy Clocking, involving the location sensor and web application components.

method, we chose to make two Business Information Flow Models[Dolog et al., 2009, c. 4] in order to give a more detailed description of the information flow. This helped us in understanding what information needed to be communicated and how the information flowed through Easy Clocking.

8.1.1 Information Flow in Easy Clocking

Figure 8.2 depicts the flow of clocking information between the location sensor and the web application. When a butcher is close to a location sensor, it detects the MAC address of his tag and measures the distance to the butcher using RSSI values. The MAC address, timestamps specifying the start and end clock, and the measured distance must be combined, and passed to the web application which in turn stores the information. The managers are then able to query the information using the web application.

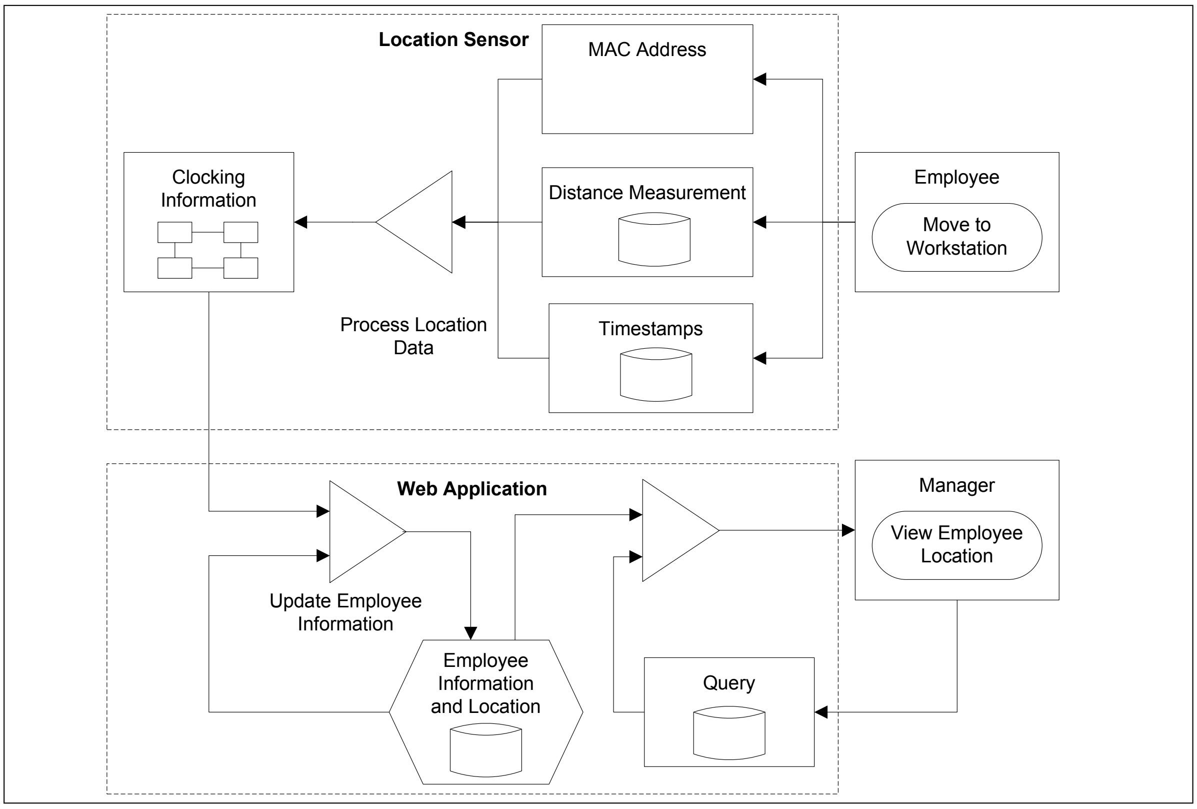


Figure 8.2: Flow of information in Easy Clocking, initiated by a butcher close to a sensor.

Figure 8.3 depicts the flow of information involved in changing the configuration for a location sensor. When an Easy Clocking consultant uses the web application to change the configuration of the sensors, the new configuration is stored in the web application. We chose to store this copy to ensure that even if a sensor fails, its configuration can later be restored. Afterwards, it is passed to the location sensor, which overwrites its current configuration with the new one. The configuration of location sensors is not considered to be a frequent operation since it should only be needed when the environment in which it is deployed is changed.

8.1.2 Communication Methods

We had a number of options when choosing a communication method. First, we needed to choose the component to be responsible for initiating the connection and how this connection should be realised.

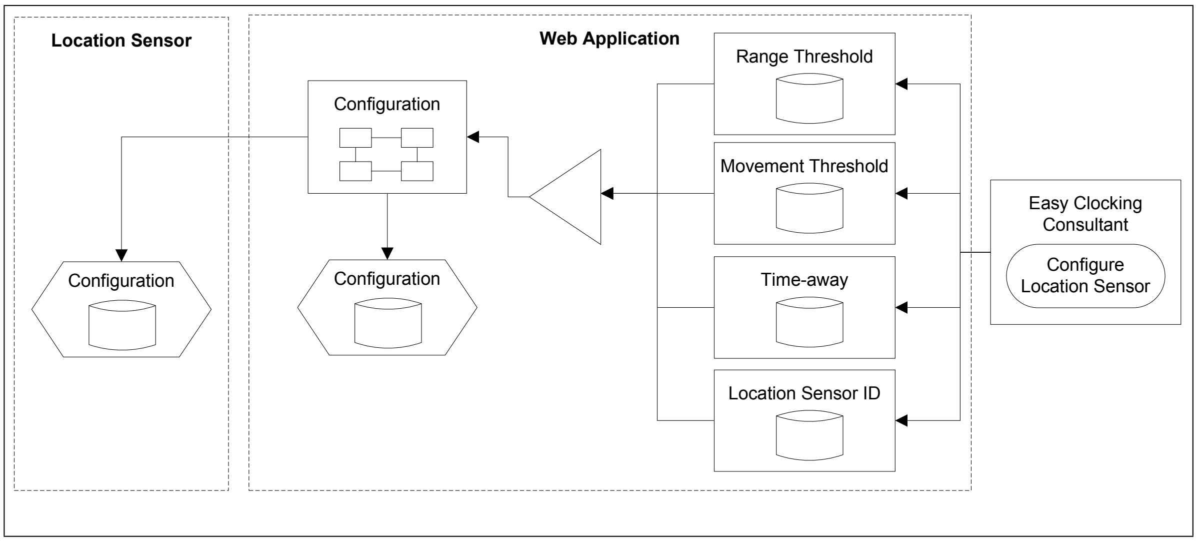


Figure 8.3: Flow of information in Easy Clocking, initiated by an Easy Clocking consultant changing a sensor configuration.

We chose to let the location sensors initiate the connection because we wanted to only transmit information when available. If the connection was initiated by the web application, this would result in probing for information even if there was not any, hence this would waste unnecessary resources.

From a previous course, we were introduced to a number of communication methods which could be used. These were SOAP, RESTful, and Internet Sockets. On previous projects, we have used Internet Sockets. Because of this, and since we through a brief analysis showed that SOAP and RESTful web services would be more appropriate choices because of their library methods and built-in support in Ruby on Rails.[Thomas et al., 2006, c. 25] Therefore, we excluded Internet Sockets, respectively.

A web service is called RESTful, if it fulfils the architectural principles of REpresentational State Transfer (REST). This means that the web service must fulfil the following:[Dolog et al., 2009, c. 2]

- Operations must use the HTTP methods; GET, POST, PUT or DELETE to interact with resources. GET should be used to retrieve a resource, POST to create a resource, PUT to update a resource and DELETE to delete a resource.
- The web service must not store session data, meaning that the web service is stateless.
- URIs hold the state information. E.g. the client can use this to continue interacting with the web service.
- The data is transferred using HTTP. Data can be represented in a markup language such as XML. The payload must be written in the body of the HTTP method.

RESTful web services are by default supported by Ruby on Rails, meaning that models can be rendered as XML documents without requiring great effort. This means that if only resources, which are already models in MVC, need to be accessed by web services, this is relatively easy to implement.

Applying SOAP in Easy Clocking would, in our opinion, require unnecessary implementation, because remote procedure calls should be implemented for each functionality, to achieve the same already existing in RESTful. Through remote procedure calls, we could emulate the resource-oriented nature of RESTful. For instance, in order to update the name of a butcher in the model, this is done by specifying the URI of that butcher and use the PUT HTTP method in RESTful and specify the name attribute to be updated. In SOAP, this functionality would have been implemented through a remote procedure call such as `update_butcher_name(butcher_id, name)`.

We chose to use RESTful web services instead of SOAP since Ruby on Rails, supports the concepts of REST by default. The fact that the location sensor configuration and clockings are models in Rails, hence resources, we did not need to communicate information that was not already represented a model in Easy Clocking. This is in contrast to SOAP where the web services are more focused at exposing operations than resources through remote procedure calls.

When using scaffolding in Ruby on Rails, the GET, POST, PUT and DELETE HTTP methods on resources automatically comply with REST, and the client can choose to receive the data rendered in XML. Even though we only wanted to use the web service for realising the two previously described communication paths, it would not require much effort to make other resources available by means of web services.

8.2 Location Sensor

The location sensors detect employees and register their arrival with a clock in timestamp, and their departure with a clock out timestamp. This information is later sent to the web application.

We considered the possibility of letting a device carried by the employee handle the clock in and out logic. However, in our opinion this was an inappropriate solution since it would be possible for an employee to manipulate the clocking information to his own benefit if appropriate security mechanisms were not implemented.

As shown in Figure 8.1, the location sensor is based on a layered architecture, that is, functionality of one layer builds upon the functionality provided by the layer below. This helps increasing the abstraction of lower layers in upper layers, which also improves the cohesion of the components of the location sensor. E.g. all functionality related to using Bluetooth devices is encapsulated in a single component, which publishes a simple API, free of implementation details, to the above layers. This makes the code in the above layers more understandable.[Design, 2000]

The different components of the location sensor are described below.

`Clock Engine` This component uses the `Bluetooth API` to detect butchers in a given range from the sensor. To do this, it uses the `Configuration Handler` and the `Bluetooth API` to access the configuration and to communicate with the Bluetooth device, respectively. Finally, when clocking information is ready for being passed on to the web application, the `Web service API` is used.

`Configuration Handler` The purpose of this component is to provide an abstraction of the `Persistent Configuration` component with functions such as reading and writing to the configuration. The abstraction is used by the `Clock Engine` because the configurations define the behaviour of the clock engine.

`Persistent Configuration` This component stores the configuration of the particular location sensor. The configuration comprises variables defining the behaviour of the location sensor and variables such as the URL to the web application. The configuration of the location sensor is stored persistently and hence provides a means for the consultant to configure the location sensor when it is deployed such as setting the URL to the web application.

`Web service API` In Section 8.1, we described the information flow in Easy Clocking. From this we identified that the location sensors needed to communicate with the web application through web services.

The `Web service API` gives an abstraction of the communication with the web application. The component is therefore used by the `Clock Engine` and the `Configuration Handler`. The `Clock Engine` component uses the component for posting the retrieved clocking information to the web application. The `Configuration Handler` uses it for interacting with the functionality for checking the consistency of the configuration file kept on the web application and on the location sensor. If the configuration file is inconsistent, the `Configuration Handler` uses it for retrieving the configuration kept on the web application. Web services defined in `Clocking Information` and `Workstation` of the kernel are used for the described purposes.

`Bluetooth API` This component creates an abstraction of the communicating the Bluetooth hardware.

8.3 Web Application

The web application component constitutes the part of the Easy Clocking system which will be responsible for interacting with the users of the system, that is, the managers and the Easy Clocking consultants. Moreover, it is also the part of the system which will be responsible for communicating with the location sensors through the RESTful web services.

The web application is comprised of two different components, namely the kernel and plug-ins interacting with the kernel. These two are introduced in the following.

8.3.1 Kernel

The kernel component contains the logic needed for realising the fundamental functionality of the system. Specifically, its area of responsibility is to publish web services to the location sensors and to provide basic functionality not influenced by the specific business logic of the various companies using the system.

As shown in Figure 8.1, the sub-components of the kernel conform to the architectural pattern of MVC, which will be described in the following section. Each of the components constituting the MVC pattern contains four components which altogether realise the kernel. These are introduced in the following.

`Clocking Information` This component is responsible for handling the clocking information of the system. First of all, by adhering to the principles of REST, it makes it possible for location sensors to post their clocking information. The `Clocking Information` component is after retrieval of this information responsible for storing it in the model, and, hence, the database. Finally, the view part displays the clocking information.

`User` The `User` component contains the logic needed for a user system consisting of two different roles; manager and consultant. In addition, it also contains a pseudo-user for the location sensors, needed for restricting access to the web services and to the functionality for posting clocking information. The `User` component also contains an authorisation system which restricts access to the web application if the user has not logged in.

`Workstation` This component represents the workstations of the system and, hence, there is a workstation entity stored in the model for each location sensor. The `Workstation` component also contains the configurations for the individual location sensors. Through the view of this component, consultants can alter the configuration which will be propagated to the corresponding location sensor. The propagation process is realised by publishing web services for the location sensors.

`Employee` This component represents the employees of the system. Each employee has a MAC-address associated which is used to connect retrieved clocking information with a specific employee.

Model View Controller

As shown in Figure 8.1, we chose to use the MVC architectural design pattern for the web application. This pattern separates the presentation layer from the model and separates the controller from the view.[Fowler, 2002, c. 14] Thus, the fundamental idea is to separate concerns of the development. This way, it is possible to reuse code on several pages instead of otherwise having duplicated code. Also, a significant advantage of this separation is that it provides a natural way of developing a web application.[Black, 2006]

Note, that there exist a number of alternative design patterns, such as the Front Controller and Page Controller, which can be used to structure web applications. Front controllers uses a single controller, parsing the request and determining which concrete action to perform. Using the page controller, a separate file is created for each logical page, relying on the web server to determine which page controller to invoke.[Fowler, 2002]

The different components of MVC are described in the following and is illustrated in Figure 8.4.[Dolog et al., 2009, s. 7.1.3]

Model The model contains the business logic. Generally the model defines business actions which can be reused. Furthermore, it should be mentioned that the model ignores how the data are represented for the users.

View The view contains the user interface logic. A model may have multiple views which can be used for different purposes.

Controller The controller manages requests made to the application by intercepting the request and then decide which business action to be used. This business action then changes the state of the application, activating the view which then presents the processed data for the user.

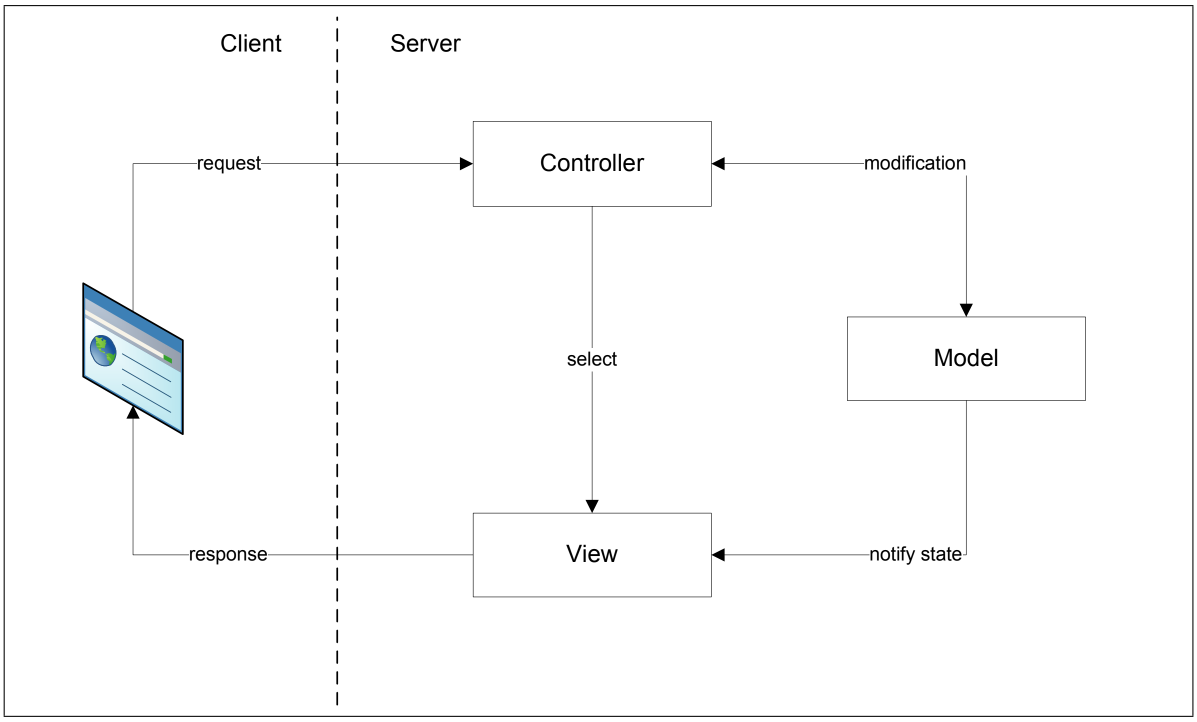


Figure 8.4: The Model View Controller architectural pattern used for a web application.[Fowler, 2002, c. 14]

The following example shows how MVC can be used to structure the development process.

Example 1 (Example of how MVC can be used.)
Initially, the development of the web application can start with identifying all the entities that constitutes the domain of the web application. In this phase, the developer does not need to consider which actions should be provided or how data is presented. Instead, the model of the web application is developed. Afterwards, decisions about which actions should be provided to interact with the model can be considered which constitute the development of the controller. Finally, the developer can make decisions about how the actions should be made available or represented in the view.[Black, 2006, c. 2]

8.3.2 Plug-in

As described in Chapter 5, we chose to use a pure plug-in architecture for Easy Clocking. This is depicted in Figure 8.1 where plug-ins can be connected to the kernel. Having plug-ins connected to the kernel might seem as conforming to the traditional plug-in architecture. Actually the kernel is intended to be implemented as a plug-in itself, which other plug-ins depend on, giving the opportunity of using Easy Clocking as a plug-in in other systems. This helps achieving the flexibility quality factor, since Easy Clocking either can be used as the main application using external plug-ins or be a plug-in in a larger application.

Ruby on Rails has a built-in plug-in architecture, with the same characteristics as the pure plug-in architecture. The architecture allows packaging models and controllers into a plug-in which can be combined with additional plug-ins into a single application. We extended Ruby on Rails with the Desert[1] Ruby gem, which enhanced the plug-in architecture by adding support for migrations and views in the plug-ins.

The pure plug-in architecture has some pitfalls as described by Birsan [2005]. We have added notes regarding these pitfalls and how they apply to the plug-in framework for Ruby on Rails in Appendix J.

Using Ruby on Rails' built-in plug-in architecture extended by Desert fulfilled the requirement of being a pure plug-in architecture and was supported by Ruby on Rails simplifying future implementation.

[1]github.com/pivotal/desert

9 Location Sensor Design

In designing the location sensor software, we chose to use flow diagrams to represent the logic of it. This makes it possible to give a structured design of a procedural language such as C before writing the code. Blauch and Johnson [2001] describe several advantages of having a structured design, such as early detection of design flaws. This is a great advantage because it may require a lot of effort to fix a design error found at the end of the implementation. Using the structured design approach also helps splitting the application into modules which then can be individually developed by multiple teams.[Blauch and Johnson, 2001]

Flow diagrams are shown for the clocking process and the configuration consistency process, which we found non-trivial and most interesting. We chose that the clocking and the configuration consistency process run concurrently. Separating these in their own thread of execution is convenient because it provides an easy way of controlling when for instance the configuration consistency check is conducted.

9.1 Clocking Process

In the design of the clocking process, we used the knowledge gained by examining the application and problem domain, described in Section 1.3. From these, we were able to formulate the necessary logic used when clocking in and out, the configuration parameters, and conflicts. In the following, we briefly describe the parameters and argue for their necessity.

Distance Threshold To limit the area in which the location sensors registers butchers, a predefined distance threshold is used. E.g. a workstation in a cold store may require a relatively large detecting range in contrast to a workstation at an assembly line.

Time-away Threshold In some circumstances, it may also be necessary for the butcher to leave the workstation temporarily. A time-away threshold denoting for how long a butcher is allowed to be absent from his workstation is used. This is in order to avoid clocking out the butcher immediately.

Movement Threshold If the butcher moves beyond this threshold from his average observed distance, then a new clocking information is created for that new average. Thus, if he walks from a machine to an adjacent machine, clocking information will be created for the time intervals he was close and far away from the location sensor, respectively.

In respect to the conflict detection in overlapping areas, we chose not to let the location sensors be responsible for resolving this issue. Not doing so would require them to exchange information about registered butchers. Instead, we chose to let the location sensors pass all their location registration data to the web application, which would then be responsible for resolving multiple occurrences of a butcher from different location sensors.

On the assumption that the manager has specified which tasks a particular butcher has to perform during his work day, the web application would be able to resolve multiple occurrences of the same butcher by comparing the location sensors corresponding to his tasks with the location sensors that passed information to the web application. The location data sent by the sensors, not related to his tasks, could be marked as discarded by the web application.

Finally, to address the conflict of the scenario where a butcher is detected by two workstations in the same time interval, which he is assigned to work at during the work day, we decided to let the location sensor store ranges from the butcher to the location sensors in time intervals. This way, it would be possible to compare ranges in time-intervals hence determine how the butcher had been working. Therefore, we chose to design the application such that if a given movement threshold between the butchers previous range and his current range is exceeded, the butcher must be clocked out with his previous range and clocked in with the new range. If the movement threshold was not exceeded, the previous range would have to be updated taking into account previous ranges and the current range. An example of the result of using the movement threshold is shown below:

Example 2 (A butcher detected at multiple workstations.)
Michael has been detected in the following time intervals with the corresponding range to the bone cutter.

13:34 - 13:56, 1 meter
13:56 - 14:56, 8 meter

In the same time intervals he is detected at the mincer.

13:34 - 13:56, 9 meter
13:56 - 14:56, 2 meter

This example illustrates how the range for time intervals can be used to infer that Michael has been working at the bone cutter from 13:34 to 13:56 and at the mincer from 13:56 to 14:56.

Figure 9.1 depicts the flow diagram of the clocking process. As shown, a distance threshold is checked when unprocessed localisation data is received. If this holds, the butcher is either clocked in, if it is the first time he is detected, merged if he has not moved beyond the movement threshold since last detection or, if he has moved, clocked out with his previous range and clocked in with a new range. Finally, when the localisation data has been processed, the butchers are clocked out if they have not been detected within the time-away threshold.

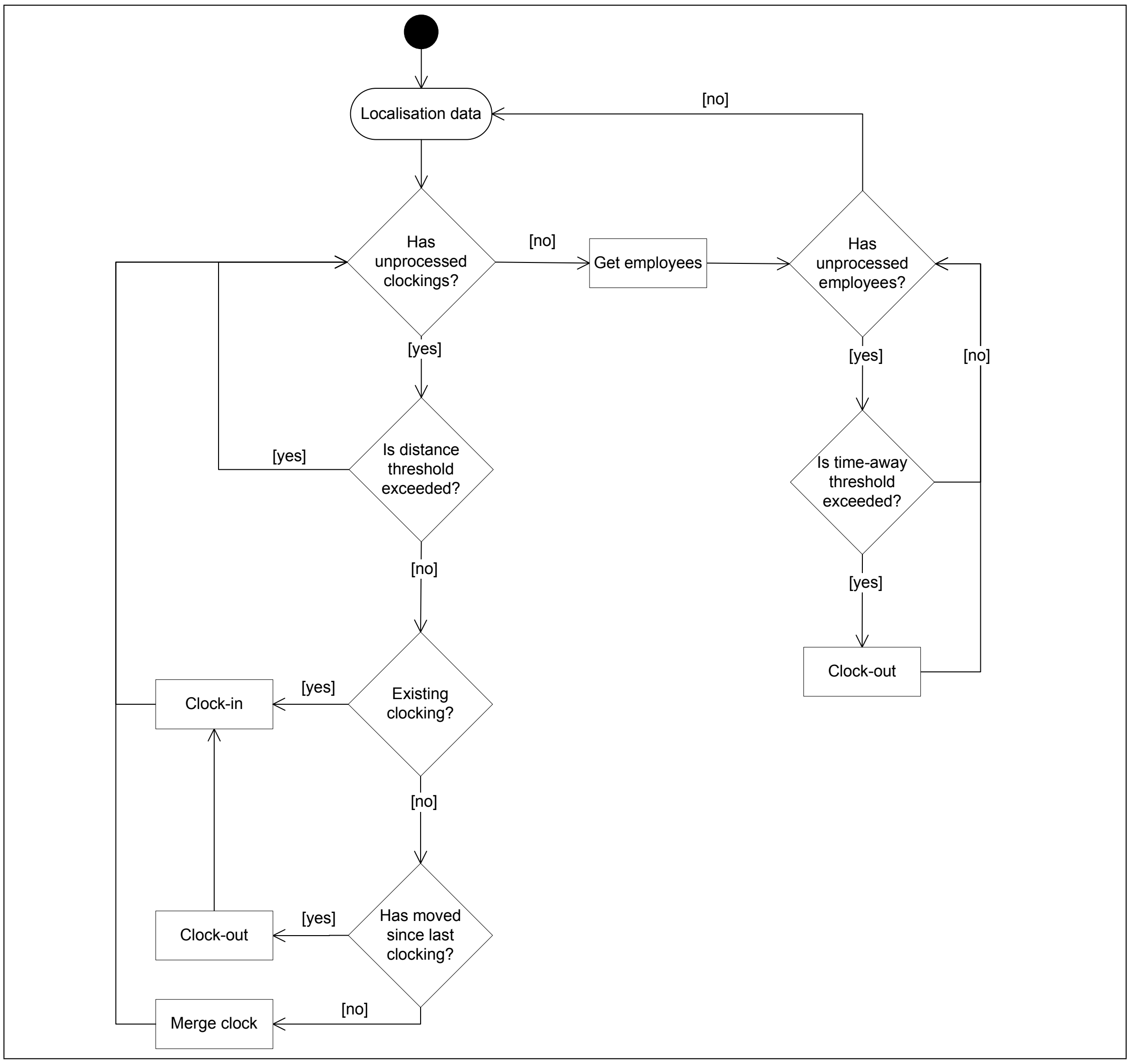

Localisation data
[no]
Has unprocessed clockings?
[no]
Get employees
Has unprocessed employees?
[yes]
[yes]
[no]
[yes]
Is distance threshold exceeded?
Is time-away threshold exceeded?
[no]
[yes]
Clock-out
Clock-in
[yes]
Existing clocking?
[no]
Clock-out
[yes]
Has moved since last clocking?
Merge clock
[no]

Figure 9.1: Flow diagram showing the processing procedure for clocking information and subsequent check of employees that must be clocked out.

9.2 Configuration Consistency Process

The configuration of the location sensors is made by an Easy Clocking consultant through the web application, as described by the consultant's responsibilities in Section 1.3.

In Section 8.1, we described it was the responsibility of the location sensors to initiate a connection to the web application in order to get the configuration. To keep the configurations on the web application and on the location sensors consistent, it was therefore required by the location sensors to regularly request the web application for updates of their configurations.

This would require the configuration to be passed over the network each time the location sensor requests for potential changes in the configuration. Since we did not expect the configuration of the location sensors to be changed often after the initial installation of the location sensor, we wanted to only transmit the configurations when changes had been made to them. Thus we divided the update process into two calls to the web service of the web application. One call checks if an updated configuration is available and one fetches the configuration if this is the case. This reduced the amount of overall network traffic between the location sensors.

Note that after the initial installation of the location sensors where they are configured according to the given environment, the configuration can only be expected to be changed in case the environment in which the workstation is situated, is changed. We assessed that checking for configuration updates once a day was sufficient. However, this results in periods where the configuration could be inconsistent.

Figure 9.2 depicts the flow diagram of the process described above.

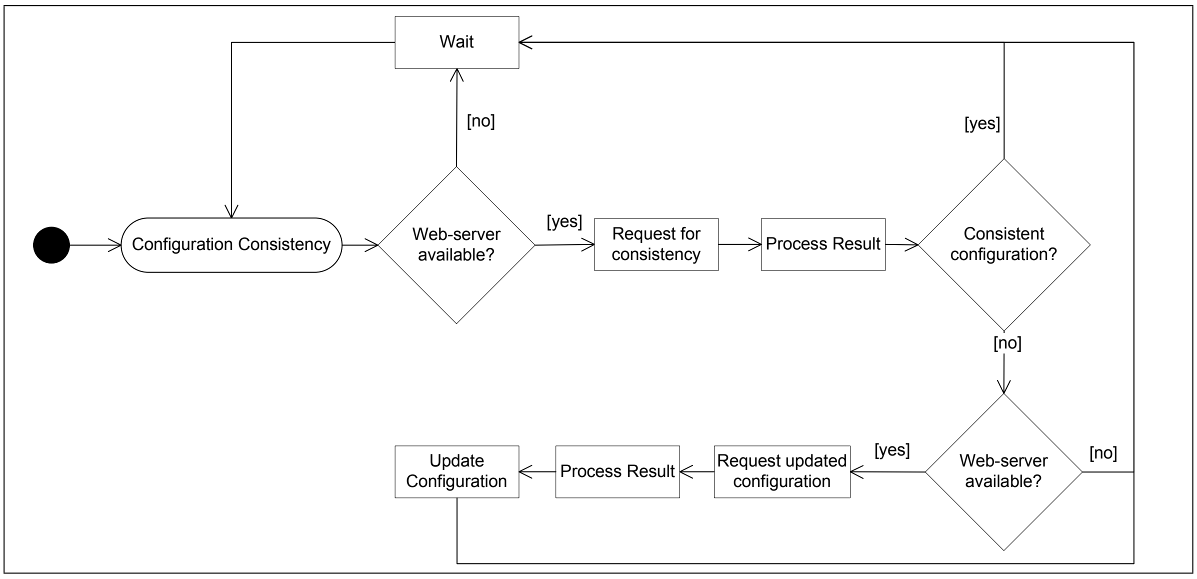


Figure 9.2: Flow diagram showing the process of keeping the configuration consistent with the configuration stored by the web application.

10 Web Application Design

In this chapter, we document the design of our web application using the work products: navigation design, data design, workflow design and presentation design and architecture design. These are described by [Dolog et al., 2009, c. 5] as models for documenting web applications. Notice that the architectural design was made in Chapter 8

The different work products helped us in both understanding the structure of the web application and in identifying the functionality we needed to implement.

10.1 Navigation Design

A navigation flow diagram consists of views and relations between these depicting navigation paths for the user.

The navigation flow diagram for our web application is shown in Figure 10.1. It was based on an analysis of the requirements for the web application as described in Chapter 3 and further refined throughout the development process.

We identified a number of resources, each containing a generic set of views such as a view for adding new resources or editing a resource. The generic views are shown in the navigation flow diagram in the upper right corner. Each resource, such as the Assignments resource, encapsulates a number of views related to it. This is used later in the implementation when creating controllers, where there will be a controller for each resource.

If the user's credentials are invalid, he will be redirected to the Login page. When his credentials are valid, he will be allowed to proceed to other parts of the application depending on his user role, that is, manager or consultant. The consultants are allowed to access the consultants and workstations resources, and managers the remaining resources.

10.1.1 Web Services

In order to support the location sensors, it was necessary to provide two web services, namely, one to handle consistency of configurations and one to receive new clocking information as previously described. In Section 8.1 we argued that RESTful web services must be used. The

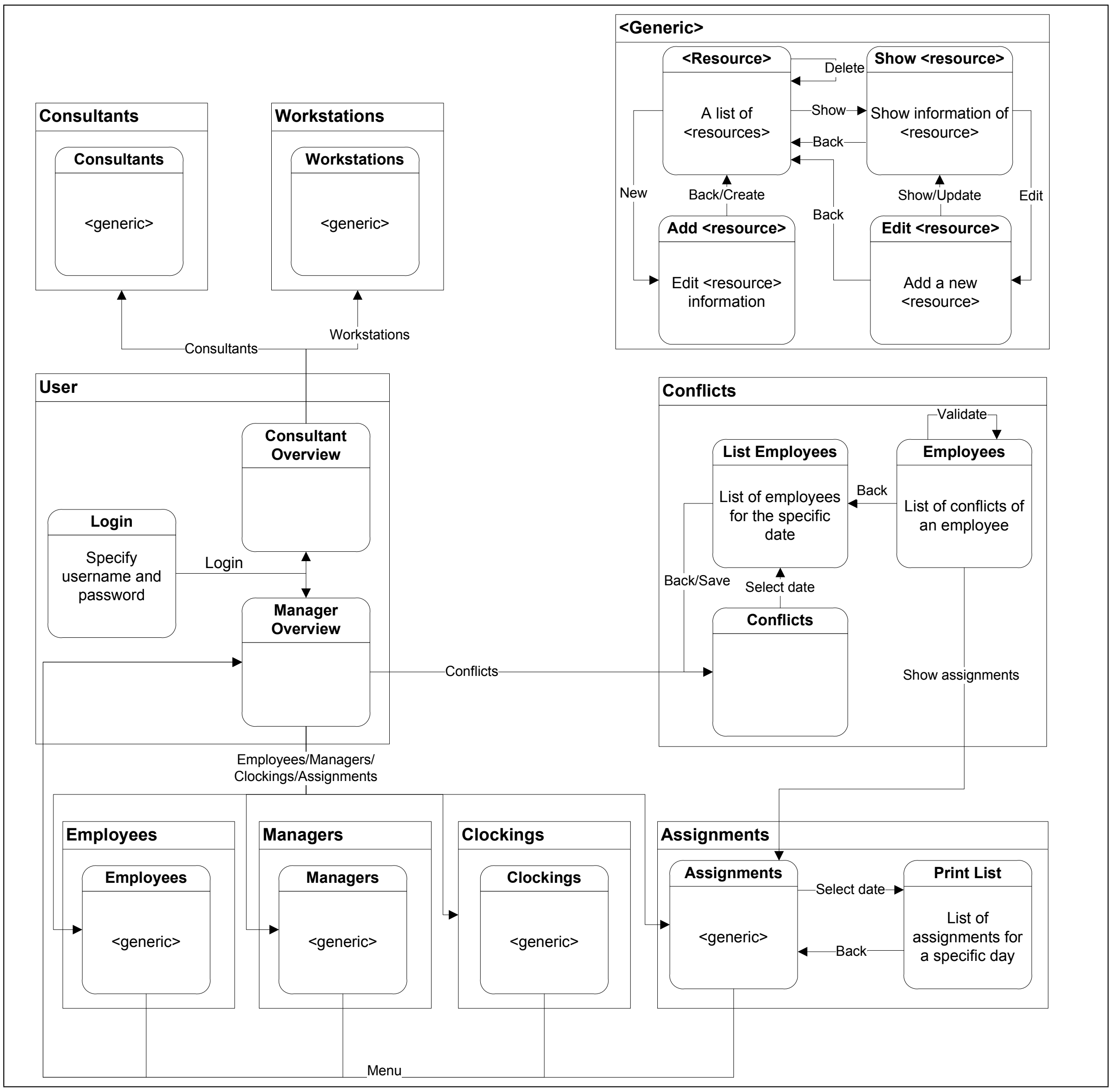


Figure 10.1: Navigation flow diagram of the web application including boxes indicating the different controllers.[Design, 2000]

following two views are made available to the location sensors, and are not included in the navigation diagram since the location sensor does not navigate through the views:

Clocking According to the navigation flow diagram, it is possible for a manager to add new clocking information manually. Since the location sensors also require this functionality, we decided to reuse it by letting the location sensors post information to the same view as the managers after successfully authenticating themselves.

Configuration Consistency The location sensors needed to access their configuration as a step in their configuration consistency process. Therefore, we allowed access for the location sensors to the workstation configuration view. This view returns the configuration rendered in HTML for the consultants and XML for the location sensors.

To reduce the amount of transmitted traffic over the network, an additional view was added which allows the location sensor to check if a new configuration is available.

10.2 Data Design

In the data design, we analysed which data must be stored by the system and how the data were interrelated. This resulted in the ER-diagram shown in Figure 10.2.

First of all, Easy Clocking had to contain users who could use the web application, namely the managers and the consultants. Since the same type of information needed to be stored for each type of user, we decided to have one table for users and distinguishing their role by an attribute. In addition, employees, assignments, workstations, clockings and the configuration of the location sensors were also stored in the database.

As shown, there is a many-to-many relationship between assignments and employees. The reason for this is that butchers can be assigned multiple tasks and multiple assignments can have multiple butchers. Also, there is a many-to-one relationship between task and workstation since assignments can only be related to one workstation, but a workstation can be related to several assignments.

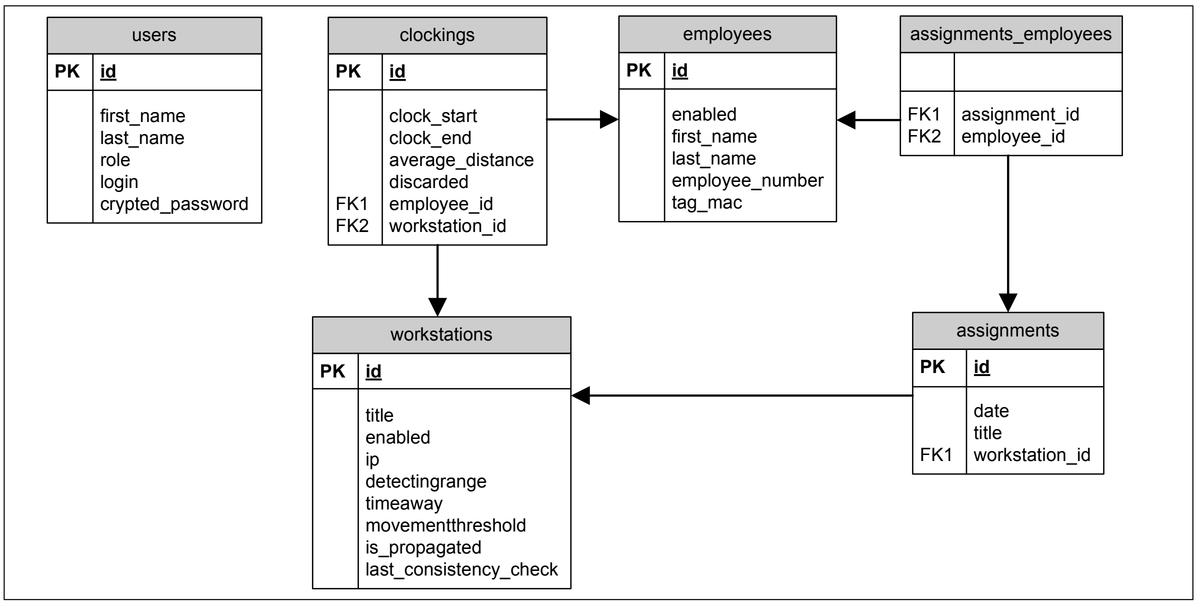


Figure 10.2: ER-diagram showing the data in the web application.

Using the ER-diagram, it was possible to identify the models in MVC. As an example of how this was done, one can consider the employee table. First, this is an entity in the system, which indicates that it is a resource hence a model. Furthermore, the relationships between the models can be inferred from the diagram.

10.3 Workflow Design

The workflow design model, as described by [Dolog et al., 2009, C. 5], is used to model the workflow of a web application and how a user interacts with it throughout the process.

We chose to create a workflow model for managers solving conflicts in the location sensor data. This diagram was made since conflict handling is non-trivial and is central for the usage of Easy Clocking. Using this, we improved our understanding of the process and identified functionality which needed to be implemented.

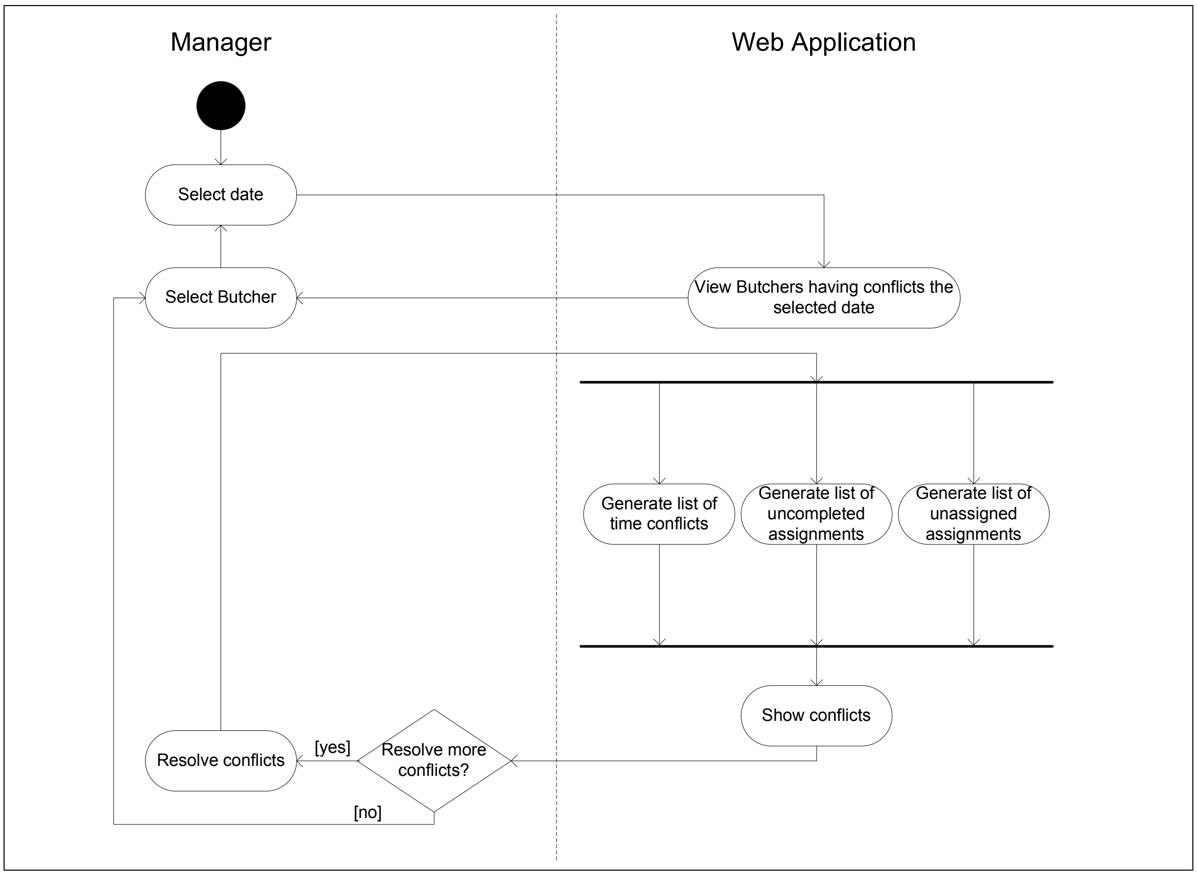

Manager
Web Application
Select date
Select Butcher
View Butchers having conflicts the selected date
Generate list of time conflicts
Generate list of uncompleted assignments
Generate list of unassigned assignments
Show conflicts
Resolve more conflicts?
[yes]
[no]
Resolve conflicts

Figure 10.3: Workflow model showing the process involved when the manager resolves conflicts in the system.

Our workflow is shown in Figure 10.3 and is based on a UML activity diagram, as suggested by [Dolog et al., 2009, C. 5]. The figure shows that the manager initially selects a date where he wants to resolve conflicts. The web application responds by showing the butchers having conflicts at the specified date. The manager now has the opportunity of resolving the conflicts for one of the shown butchers. If he selects a butcher from the list, the web application will perform some algorithms on the clocking information of the selected butcher which will output the conflicts. This is represented in the diagram by the fork to three actions which are joined afterwards to display the total conflicts.

After the conflicts have been generated, the manager now has the opportunity of resolving them. Because there may be dependencies among the various conflicts, the manager can validate his changes whenever he likes. The manager can at any point choose to go back to the the list of butchers having remaining conflicts.

10.4 Presentation Design

The goal of the Presentation Design is to structure the information on individual pages in order to bridge the gap between the web application design and the visual representation of it.[Dolog et al., 2009, C.5]

We will use the abstract widgets design, as described by [Dolog et al., 2009, C.5], to describe the user interface at a high level of abstraction. The abstract widgets design consists of compositions of widgets, each assigned a role which indicates its responsibility such as displaying text or receiving user input. This helped in determining the different elements a view consists of without taking into account layout and graphics.

We only used this work product in order to design the conflict page, since this was one of the non-trivial pages in the web application.

Figure 10.4 shows a simplified abstract widgets design for the conflict page, where parts of the contents have been simplified in order to reduce its size.

The Easy Clocking element, which is the main application itself, is composed of a Main Menu, a Main Body and a Site Banner widget. The Main Body contains a Page Title, a Status Indicator for Conflicts, and an element for each type of conflict. The List of Clockings widget, which is used to display time conflicts, further contains Clock Collections which again contains individual Clockings. The necessity of this widget is described later. The Clock Collection widget continues down to individual pieces of information shown in the view, such as the name of a workstation for a clocking.

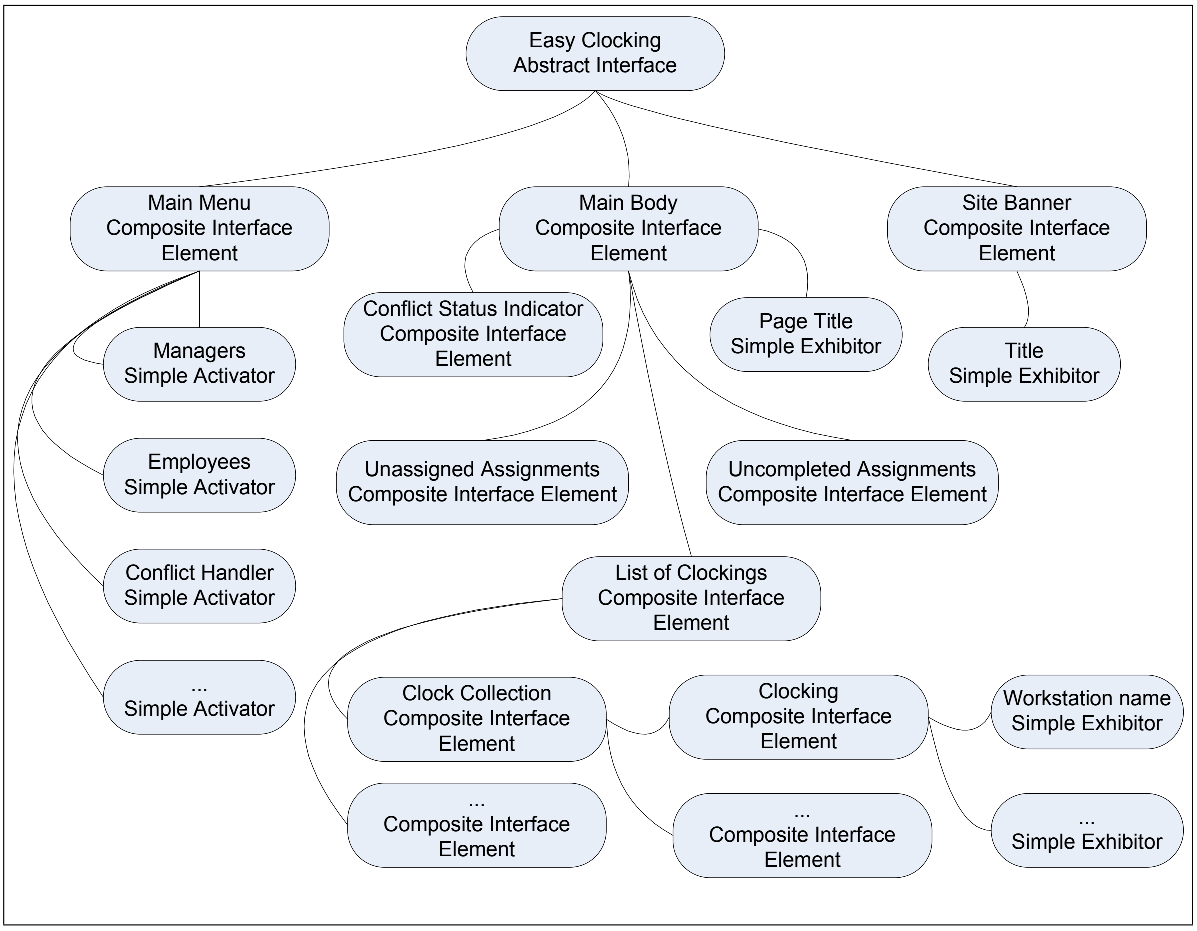


Figure 10.4: Abstract widgets design for the web page displaying a butcher's conflicts for a given day.

Part III

Implementation

11 Location Sensor Implementation

The location sensor part of the system is divided into two separate main loops: the employee monitor loop and the configuration consistency loop. The design of these was given in Chapter 9. The employee monitor loop processes employee location information and notifies the web application when an employee clocks in or out. The configuration consistency loop updates the location sensor configuration if it has changed. In our implementation, the two loops run concurrently which is realised by using an implementation of POSIX Threads.

In the following, we provide a brief introduction to the start-up phase of the location sensor application in order to give the reader an idea of how the application on the location sensor works. Generally, the start-up phase can be divided into two steps, namely: initialise configuration and initialise threads.

Initialise configuration When the location sensor application is executed, it starts by requesting the configuration of itself from the web application. The web application will respond with a set of configuration variables. The location sensor will afterwards write these to its own configuration file and, in addition, set the internal configuration of the application accordingly.

Initialise threads After the configuration has been initialised, two threads are created for the previously described loops and the location sensor application is transitioned to the running phase.

In the following, we will document the implementation of the two loops.

11.1 Employee Monitor Loop

In the design of the location sensor described in Section 9.1, we identified the activities involved in the employee monitor loop. In this, employee location information is processed and employees clocked out. Afterwards, the clocking information of clocked out employees are sent to the web application. Figure 11.1 shows a sequence diagram of the process described earlier.

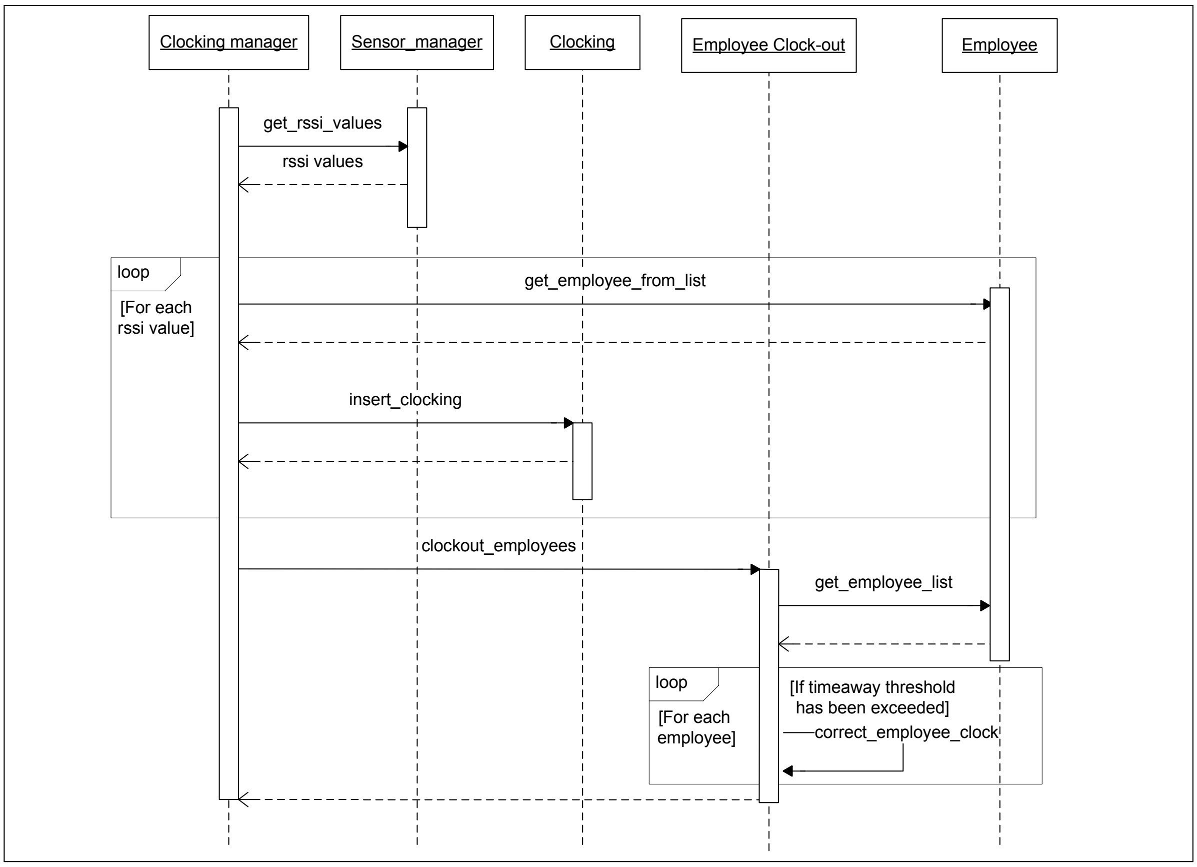


Figure 11.1: Sequence diagram showing the employee monitor loop.

Each entity in the figure, such as the sensor manager, corresponds to a file containing a set of C functions and data structures needed for realising the functionality of the particular entity.

In the following, we give a brief description of each of the entities and functions involved in the sequence diagram.

Clocking Manager The responsibility of the clocking manager is to process the raw data retrieved from the location sensors, calculate the distance to the butcher and store the data for later usage. The calls shown in Figure 11.1 originates from the function `clocking_start()`.

- `clocking_start()` - The purpose of this function is to start the clocking process. When the incoming RSSI values are found, the clocking manager is used to insert them into the correct RSSI buffer. An RSSI buffer is associated with each employee instance and is used as a part of the process of merging clocking information and making new clockings.

Sensor Manager The sensor manager abstracts the communication between the Bluetooth device and the rest of the application. Communicating with Bluetooth devices is realised by using the BlueZ library. BlueZ is the official Linux Bluetooth protocol stack[1].

- `get_rssi_values()` - The purpose of this function is to get RSSI values from all the detected tags, and return these with the corresponding MAC addresses.

Employee Defines an employee data structure and functions used to manipulate this. The employee data structure stores the employees and pointers to the relevant clocking data structures.

- `get_employee_from_list()` - When the function is invoked with the MAC address of the detected butcher, either a butcher structure already present in the list is returned or a new structure is added to the list and returned.
- `get_employee_list()` - The purpose of this function is to return the data structure of the employees.

Clocking Defines the clocking data structure used to represent the time intervals employees have been present at different ranges from the location sensor.

- `insert_clocking()` - The purpose of this function is to insert new clockings into a clocking list.

Employee Clock Out Defines the logic needed for clocking out employees.

- `correct_employee_clock()` - The purpose of this function is to correct the clock of the employee if he has been absent for longer than the time-away threshold. In this case, the clock out timestamp of the last inserted clocking element of his clocking list must be corrected with the timestamp for which he was last seen, that is, the last time he was within the detecting range of the location sensor.

11.1.1 Data Structures

The purpose of this section is to give an overview of the date structures in the clocking component.

The data structures, including interrelationship and details about their contents, are depicted in Figure 11.2.

The clocking and employee data structures are interrelated sequentially in singly- and doubly-linked list, respectively. First of all, a doubly-linked list data structure was chosen for representing the employee elements. We also considered using various tree data structures or

[1] `bluez.org`

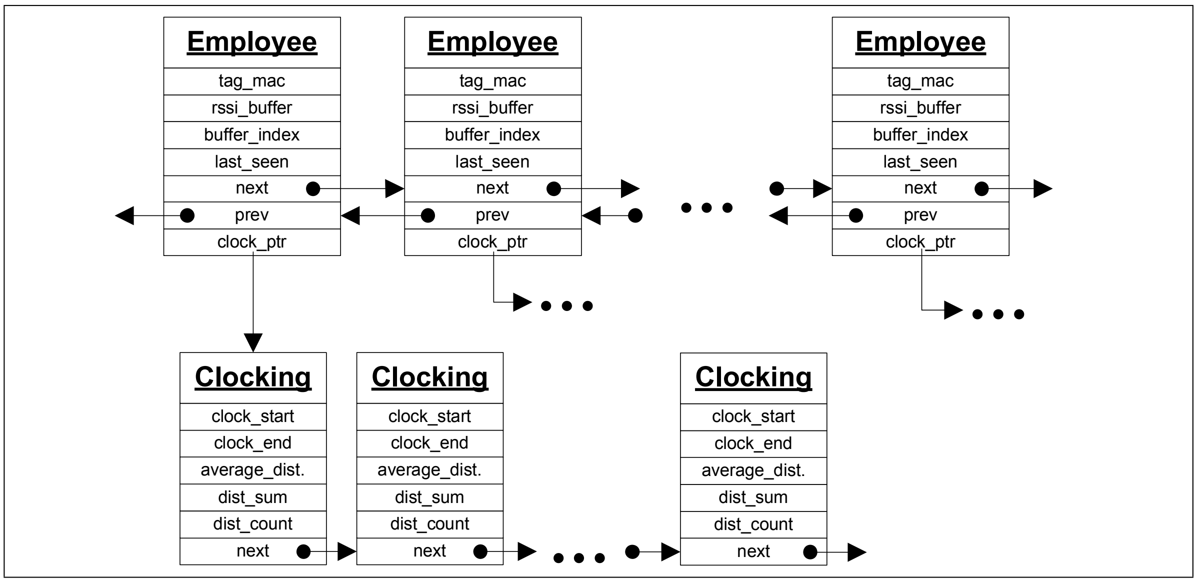


Figure 11.2: Illustration of the data structures of the location sensor.

hash-table structures. However, even though such data structures perform better in lookups, we chose to use linked lists. This decision was made because the implementation was simpler and we only anticipated a small amount of employees in the list. We assumed that only ten employees would be present at the location sensor at one time, as defined in our requirements. If this assumption would prove to be wrong, then changing the data structure would be trivial, as long as the API is preserved to access and use the data structure.

Choosing a doubly-linked list instead of a singly-linked list was based on the need of sometimes removing employee elements present in the middle of the employee list. This is easier with a doubly-linked list.

Each employee element contains a pointer to a list of clocking elements containing information about clock in and clock out times etc. The clocking elements are ordered into a singly-linked list. This is sufficient because we primarily insert new elements to the head of the list and search sequentially through it.

Arranging the clocking elements in a data structure was done as part of implementing a caching mechanism. According to our non-functional requirements, an important consideration was to ensure that the clocking information remain durable. It must be ensured that whenever clocking information are captured and stored temporarily in the location sensor, loss of this data must not happen as a consequence of a connection failure to the web application. Data is only removed from the internal memory of the location sensor when the clocking information has successfully been transmitted to the web application. The linked list of clocking information is used to store clockings that have failed to be transmitted. When a connection to the web application is available again, the linked list of clockings is emptied after successfully sending the clockings.

Memory Consumption

In Chapter 7, we stated that we had relatively limited memory resources available, namely, 32 MB system memory shared between all processes. To ensure that the above mentioned data structures could fit into this memory in case of a network failure such that clocking information is accumulated, we conducted the following calculations. A single entry in the employee list and clocking list takes up 48 bytes and 32 bytes of memory, respectively. These numbers have been obtained by using the `sizeof` operator from C.

The following example illustrates the number of possible clockings using 1 MB of the total memory.

Example 3
If we assume the location sensor has registered 1000 employees, the following number of clockings can be stored in 1 MB of memory.

$$\frac{1MB - 1000 \times 48byte}{32byte} = 29.750$$

This results in space for 29.750 clockings, or approximately 30 clockings for each employee.

Using the above example, we assessed that the available memory is enough and that it would be possible to accumulate clockings for at least a couple of days if the communication path between the sensor and web application fails.

11.1.2 Code examples

To document the implementation of the employee monitor loop, code examples are shown in the following.

Listing 11.1 shows the implementation of how clock in is made.

```
static void filter_dist_and_insert_to_clocking_list(employee_str *employee, float
    new_dist) {
   if(is_list_empty(employee->clock_str) == IS_EMPTY || fabs(employee->clock_str->
       dist_average - new_dist) >= get_movement_thres())
      employee->clock_str = insert_clocking(employee->clock_str, new_dist);
   else
      merge_dist_value_and_clock_str(employee->clock_str, new_dist);

   update_employee_timestamp(employee);
}
```

Listing 11.1: The function responsible for making new clockings and merging existing clockings.

As shown, the condition checks if the given butcher already contains a clocking struct. If it does, a new clocking struct is inserted to the front of the list. Besides this check, the current range between the employee and location sensor and the average distance of his most recent clocking is compared against the predefined movement threshold, namely the movement threshold. If none of the above mentioned conditions are fulfilled, the new range is merged with the current clocking struct by updating the average range of the time interval. Finally, the timestamp indicating when the butcher was last detected, is updated.

To clock out, we have implemented the code shown in Listing 11.2

```
void clockout_employees() {
    employee_str *employee_list = get_employee_list();

    while(employee_list) {
        if(difftime(time(NULL), employee_list->last_updated) > get_time_away_thres()) {
            if(is_list_empty(employee_list->clock_str) == IS_NOT_EMPTY)
                correct_employee_clock(employee_list);
        }
        employee_list = employee_list->next;
    }
}
```

Listing 11.2: The function responsible for clocking out employees.

The loop of the function traverses the list of butchers and checks if their last seen time exceeds the predefined time-away threshold for the particular workstation. If this is exceeded, the butcher is considered away from the workstation and must be clocked out. This is done by inserting the last seen timestamp in the newest clocking struct.

When RSSI values have been captured, corresponding distances computed and clockings assigned to the correct employees, the list of employee is traversed sequentially. For each employee, their list of clockings is processed. All clocking structures containing a timestamp specifying a clock out time, are send to the web application. The process of sending clocking information is the last step conducted in the employee monitor loop.

The implementation of the caching mechanism is part of the process of sending the clocking information. The caching mechanism is realised simply by checking whether or not the data could be sent to the web application. If it could not, the information is kept in the location sensor's internal memory in the linked list data structure as previously described, and no information is modified. The data is attempted sent when the employee monitor loop processes the list of clocking structures again. When the data has been sent, it is removed from the linked list and the employee monitor loop starts over again.

11.2 Configuration Consistency Loop

The configuration consistency loop is responsible for keeping the configurations of the location sensor and the web application consistent. In Section 9.2, we presented the flow diagram of this loop. The implementation realising the decisions and processing states of this diagram is shown in the sequence diagram depicted in Figure 11.3.

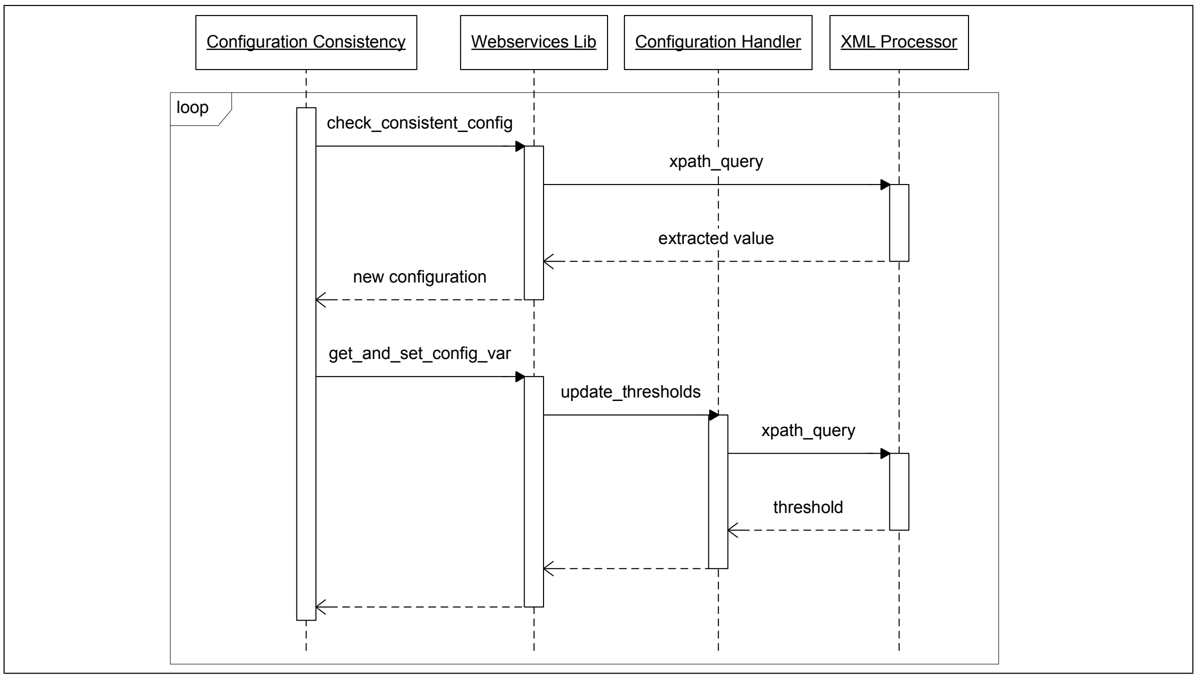


Figure 11.3: Sequence diagram showing the process involved in updating the local configuration because a new configuration has been made in the web application.

The sequence diagram shows the interrelationship between the various entities involved in the scenario where the configuration consistency loop determines that a new configuration has been made in the web application.

The general process of performing configuration consistency on the location sensor starts with making an HTTP GET request to the web application. Based on this request, the web application looks-up if any new configuration is available that needs to be propagated to the sensor. In the scenario where a new configuration is available, the HTTP GET request returns an XML document indicating this. A new HTTP GET request is made for retrieving an XML document containing a representation of the configuration of the given workstation. The values of the configuration are extracted using XPath expressions. The local configuration, which includes the configuration file and internal configuration variables, are updated according to the extracted values.

When we implemented the configuration consistency part, we had some considerations with respect to how the retrieved XML document should be processed. Specifically, we had to choose which technology to use for extracting the values of the XML documents. We considered using XPath or XQuery for this purpose. XPath is a domain specific language which models the XML document in a tree structure with each node corresponding to an element in the XML document. For selecting specific nodes in the tree structure, XPath makes use of path expressions which hierarchically specify the nodes to be selected of the tree.[Møller and Schwartzbach, 2006, C. 3]

XQuery provides a means of both extracting and manipulating XML documents. Furthermore, it can be used for XML data transformations and provides many additional features such as the opportunity of using FLWOR expressions that can be used for iteration and binding of variables for temporary results. XQuery uses an XPath expression syntax for specifying nodes.[Møller and Schwartzbach, 2006, C. 6]

We needed a technology with the capability of extracting single values from an XML document. Both XPath and XQuery fulfilled this. However, XQuery provides many additional features which were unnecessary for our purpose. E.g. if transformations of retrieved XML documents were necessary, XQuery would have been appropriate. Based on this, we decided to make use of XPath for extracting the values of the retrieved XML document.[Cohen, 2009]

In the following, we summarise the purposes of the entities presented in the sequence diagram of Figure 11.3.

Configuration Consistency The configuration consistency entity represents the overall abstraction of all the details involved in the process of determining whether a new configuration has been made available and in this case update the local configuration accordingly.

- `start_conf_consistency()` - The purpose of this function is to start checking the consistency of the configuration stored in the web application and the configuration stored locally on the sensor. The function will run in an infinite loop and perform the configuration consistency check periodically according to a specified interval.

Web services Lib The web service lib entity contains all functionality which relate to communication using RESTful web services. It abstracts over setting up all the HTTP requests with the appropriate HTTP method. This is done using libcurl[2] which is a URL transfer library. The web service lib has been made thread safe using mutex on critical regions because the employee monitor loop and the configuration consistency loop use this library concurrently.

- `check_consistent_conf()` - This function performs an HTTP GET request to the web application. Due to the decision of using RESTful web services, the URL receiving this request is constructed such that it identifies the specific workstation making the request. The `check_consistent_conf()` function returns a value specifying whether or not a new configuration has been made available.

[2]`curl.haxx.se`

- `get_and_set_config_var()` - This function will be called in case there is a new configuration available on the web application. The function performs an HTTP GET request with a URL containing an ID of the workstation requesting the new configuration. The function processes a buffer containing the HTTP GET response which is an XML document and set the configuration parameters accordingly.

XML Processor The XML processor entity is a library containing all functions needed for processing the XML documents used in the application. In the location sensor, the library provides functions for evaluating XPath expressions on XML documents stored in string buffers. The functions made available by this library are based on libxml2[3].

- `xpath_query()` - This function evaluates a specified XPath expression on an XML document. The result is stored in a string for further processing. In making configuration consistency, the `xpath_query()` function will initially be used to determine whether there is a new configuration or not. If there is, additional XPath expressions are performed on the received XML document containing the configuration variables in order to extract the specific values of the new configuration. These are used later in the configuration consistency process.

Configuration Handler The configuration handler entity is responsible for all functionality concerning maintaining the configuration locally on the location sensor. This responsibility involves keeping the configuration file consistent with the internal configuration variables used in the application. It provides functions for reading, writing and updating every configuration variable. The functionality needed in relation to configuration consistency is only to update all configuration variables according to the configuration that has been received. As with the web services lib, the configuration handler has also been made thread safe using the same mechanisms for achieving this. This was necessary because the configuration handler is used by both loops.

- `update_thresholds()` - This function is used to update the local configuration in the sensor. This process involves updating the configuration file which specifies the settings the given sensor is using. Besides the configuration file, internal configuration variables are also updated to reflect the new values.

11.2.1 Code Example

In this section, we provide a code example of performing configuration consistency between the web application and the location sensor. Besides the main loop which has been described previously, an interesting part of the implementation is how the XPath expressions are evaluated on an XML document. Specifically, this operation is conducted by the `xpath_query()` function which was briefly introduced in the previous section. The code for this function is shown in Listing 11.3.

[3] `xmlsoft.org`

```
   int xpath_query(const char *xpath_expr, char **result) {
       xmlDocument xml_document;
       xmlXPathObjectPtr xpath_obj;
       xmlNodePtr node;
5      char *tmp_result;

       xml_document = *opened_xml_doc;
       xpath_obj = xmlXPathEvalExpression((xmlChar *)xpath_expr, xml_document.xpath_ctx);

10     if (!xpath_obj) {
           report_error("Error: unable to evaluate xpath expression", xpath_expr);
           xmlXPathFreeContext(xml_document.xpath_ctx);
           return 1;
       }
15     node = (xmlNodePtr) xpath_obj->nodesetval->nodeTab[0];
       tmp_result = (char *) node->children->content;

       *result = (char *)safe_malloc(sizeof(char) * strlen(tmp_result));
       strcpy(*result, tmp_result);
20
       xmlXPathFreeObject(xpath_obj);

       return 0;
   }
```

Listing 11.3: The function responsible for evaluating an XPath expression on an XML document stored in a buffer.

On line 8, *`xpath_obj`* is initialised with the result of evaluating the XPath expression provided as argument on the XML document. The evaluation is performed by the `xmlXPathEvalExpression()` function which is defined in libxml2. The *`xpath_obj`* contains information about the evaluation of the XPath expression. In order to extract the specific node which was matched with the XPath expression, line 15 extracts this and the information is pointed to by *`node`* which is defined as an *`xmlNodePtr`* type. Line 16 finally extracts the content of the node and this information is afterwards copied to *`*result`* whose address is given as argument to the function.

Prior to calling this function, a requirement is that the XML document is read into memory, and prepared to be queried. This is realised with a call to `open_xml_document()`. This function sets *`opened_xml_doc`* in line 7 to the correct XML buffer read into memory. Similarly, when the XPath expression has been evaluated on the document, one must explicitly close the document with a call to `close_xml_document()` which also de-allocates the memory used for the memory representation of the XML document.

The reason for not giving `xpath_query()` the responsibility of opening and closing the document before and after evaluating the XPath expression, respectively, is that we need to evaluate multiple XPath expressions on the same document in order to extract the configuration variables.

Listing 11.4 shows an example of a location sensor configuration rendered in XML.

```
<workstation>
   <created-at type="datetime">2009-11-10T07:53:11Z</created-at>
   <id type="integer">3</id>
   <ip>192.168.1.1</ip>
   <is-propagated type="boolean">true</is-propagated>
   <last-consistency-check type="datetime">2009-11-12T14:52:45Z</last-consistency
      -check>
   <movementthreshold type="integer">4</movementthreshold>
   <timeawaythreshold type="integer">5</timeawaythreshold>
   <detectingthreshold type="integer">10</detectingthreshold>
   <updated-at type="datetime">2009-11-12T14:52:45Z</updated-at>
</workstation>
```

Listing 11.4: XML document of a location sensor configuration.

To select the threshold nodes, the path expressions shown in Table 11.1 are used.

Path expression	Result
/workstation/movementthreshold	4
/workstation/timeawaythreshold	5
/workstation/detectingthreshold	10

Table 11.1: Using path expression on the location sensor configuration shown in Listing 11.4.

When publishing web services which communicate with applications by exchanging XML documents, it is generally a good idea to perform a validation check on the received document. This is primarily done because a naive approach of simply processing the document can result in anomalous behaviour if the document does not conform to what is expected. The approach used for validating XML documents is to define a schema describing the requirements of the XML documents. Additionally, defining such a schema, can be used as a means of documenting the API of the web service.[Ray, 2003]

The schema can be defined in various schema languages such as XML Schema and Document Type Definition(DTD) schema. DTD is a rather simple language with limited expressive power such as the inability of constraining the data type of the values in the elements and no support for namespaces. XML Schema on the other hand is more expressive than DTD, expresses the schema in XML and is self-describing.[Møller and Schwartzbach, 2006]

In Appendix I, the XML Schema and DTD defining the requirements of an XML document representing a workstation is given.

Ray [2003] states that one should consider omitting XML validation if one is reasonably certain that the XML documents conform to the rules anyway because a schema is more or less a quality-control tool. In Easy Clocking, we only make available the web services for the location sensors which we have developed ourselves. Because of this fact, validation is unnecessary since we are sure that the XML documents received on the location sensor are correct. In our case, validation would only add overhead when exchanging the documents. The validation process would be more appropriate in situations where the web services are more publicly available such as on the Internet for other users. Validation would also be important if XML documents are manually constructed by humans because this open for mistakes.

12 Web Application Implementation

In this chapter, we describe the implementation of the web application by describing our usage of scaffolding, how we have implemented authentication and authorisation ending with an in-depth description of the interface for conflict handling. Code examples for the conflict handling and screenshots of the web application are provided.

12.1 Scaffolding

Ruby on Rails offers a concept called scaffolding when developing a web application. Scaffolding is *"..a quick way to generate some of the major pieces of an application."*[Gunderloy, 2009]. It is capable of generating the necessary models, views and controllers for listing, viewing, editing, adding and deleting a resource using a command line tool.[Gunderloy, 2009]

Using this concept has certain advantages. It conforms to agile development because it is possible to have a presentable application early in the process. Furthermore, it results in fast development if the generated code does not require too many modifications in order to fulfil the specific needs.[Thomas et al., 2006]

Even though the generated code may require many modifications, Thomas et al. [2006] mentions that scaffolding still can be used. Using it in these cases make it possible for the developer to rely on the functionality of the generated code when systematically modifying or replacing it. Again, the advantage is to have an early working application with all general functionalities. This way, both developer and customer have an overview of the entire application which iteratively is developed to the final application.[Thomas et al., 2006]

Since we had no experience in using Rails, we saw it as an advantage that Rails was able to generate code which we could modify later. This way, we had the possibility of getting to know Ruby on Rails by examining the generated code. Because of these advantages, we decided to make use of scaffolding which for instance was used in developing the functionalities of adding, removing, editing and listing the butchers. A method generating the views for butchers, is given in Listing 12.1.

```
> ./script/generate scaffold Employee name:string first_name:string last_name:string
    employee_number:integer tag_mac:string
```

Listing 12.1: Command to create add, remove, edit and list views, a model and a controller for butchers through scaffolding.

12.2 Authentication and Authorisation

Ruby on Rails does not provide an authentication or authorisation framework, but RubyOnRails [2009] lists some reusable implementations.

Initially, we discussed the advantages and disadvantages of writing our own authentication and authorisation framework versus using an existing solution. Writing our own solution would result in a system better matching our requirements and problem domain, while using an existing solution would save us time and reduce the risk of faults resulting in security problems since the code had been used by a larger user base.

We chose to use one of the existing solution mentioned by RubyOnRails [2009] which were: RESTful-authentication[1], Authlogic[2] and Clearance[3]. The concepts behind the implementations were similar, providing the central authentication and authorisation logic while defining mechanisms used to integrate with our own *user* models and controllers.

Our requirement for the solution was authentication through a username and password with authorisation through the use of roles. We discarded Clearance since it lacked documentation, so we could not verify its compliance with our requirements. Of the two remaining options, we chose to use Authlogic, since its documentation better described the integration procedure.

The implementation consisted of installing the Authlogic plug-in, add the authentication business logic to the user model and add authorisation methods in the controllers. We created an `ApplicationController`, which all other controllers inherit from. In this we added a method for each user role, such as `require_consultant`, which is used to restrict access by only allowing consultants for certain views. Listing 12.2 shows the implementation of this method.

```
def require_consultant
   require_role([0], "You must be a consultant to access this page")
end

def require_role(role_ids, error_msg)
  require_user
   if current_user && !role_ids.include?(current_user.role) then
     store_location
     flash[:notice] = error_msg
     redirect_to new_user_session_url
     return false
   end
end
```

Listing 12.2: Authorisation method, only giving access to consultants.

[1]github.com/technoweenie/restful-authentication
[2]github.com/binarylogic/authlogic/tree/master
[3]github.com/thoughtbot/clearance

In line 2, `require_role` is called with a possible error message and the role to require. Line 7 checks if the user has the correct role, and if not, in line 9, shows a message to the user, and line 10 redirects to the login page. If the method does not return, the view is shown as normal.

12.3 Conflict Handling

This section documents how the conflict handling was implemented in the web application. The purpose of the conflict handler is to support the work flow described in Section 10.3 regarding managers resolving conflicts in location data.

We chose to document the conflict handler in detail since it was one of the interesting parts of the web application.

12.3.1 Models

The conflict handling component contains a single model with business logic in order to detect conflicts between clockings. The following four methods are contained in the model.

`findcollections()` This method takes a list of all the clockings for a given day and returns a list of clock collections. These are described later in the view part. The collections consist of merged clockings based on their clock start, clock end and workstations. The method is capable of filtering discarded clockings if needed, which is used when the clockings are used to detect conflicts.

`findtimeconflicts()` By providing a sorted array of a butcher's clock collections this method returns an array containing which collections, conflicted with each other.

`findmissingassignmentconflictsi()` Given a list of assignments and clockings, it returns a list of assignments that have not been completed by the butcher.

`findextraplaceconflicts()` Given a list of assignments and clockings, it returns a list of workstations that the butcher has been working at when not supposed to.

Code Example

Since the conflict handler does not contain any models which map to the database, we have chosen to show another model from the kernel of Easy Clocking. Listing 12.3 shows the `clocking` model. The model specifies which other models it relates to but does not define the contents of the model. This is deduced by Ruby on Rails based on a number of migration files which define changes to the database.

```ruby
class Clocking < ActiveRecord::Base
  belongs_to :employee
  belongs_to :workstation
end
```

Listing 12.3: The class file of the `clocking` model specifying the relationships between other models.

Listing 12.4 shows how we implemented conflict detection of clockings with overlapping time intervals.

```ruby
def Conflicthandler.findtimeconflicts(clockcollections)
    allconflicts = Array.new
    i=0
    while(i<clockcollections.length)
      if(clockcollections[i].first==nil)
        i += 1
        next
      end
      localconflicts = Array.new
      j = i+1

      while(j<clockcollections.length)
        if(clockcollections[j].first==nil)
           j += 1
           next
        end
        if clockcollections[i].last.clock_end > clockcollections[j].first.clock_start and
            clockcollections[i].first.workstation_id != clockcollections[j].first.
            workstation_id
           localconflicts.push(i) if localconflicts.empty?
           localconflicts.push(j)
        end
        j += 1
      end
      allconflicts.push(localconflicts) if !localconflicts.empty?
      i += 1
    end
    return allconflicts
  end
```

Listing 12.4: The method responsible for locating time conflicts.

As shown, the method takes one argument, which is an array of clocking information for one butcher gathered into collections. A collection is a list of time intervals, sorted on their start time, which abstracts over one or more clockings. Multiple clockings can be combined into a collection if their end time equals the start time of another clocking measured on the same workstation, which is the case if the workstation clocks an butcher out and in again in order to register a new average detection distance.

Each collection is compared to the other collections to find potential conflicts. This is done using the two loops in line 4 and 12. Time conflicts are detected in the condition shown in line 17. This condition compares the two current collections to check if the first collection's end timestamp is greater than the start timestamp of the second collection. Furthermore, a check is made to ensure that the collections are belonging to two different workstations.

If a conflict is detected, the index of the two conflicting collections are pushed to an array called *`localconflicts`* in line 18 and 19. Note that the compared collection is only added once to the list, avoiding duplicates in the list. Finally in line 25, the *`localconflicts`* is pushed to *`allconflicts`* and the loop continues with a new collection.

12.3.2 Views

From the navigation design constructed in Section 10.1, the conflict handling part of the web application is comprised of three different views. These three views are briefly described in the following.

`Date Selection` This is the initial view of the conflict handling part of the web application. For selecting a specific date for resolving conflicts, a calendar is provided. After choosing the date for resolving conflicts, the manager is redirected to the page showing the butchers related to that date.

`Conflicting Butcher Selection` This is the page to which the manager is redirected after specifying the date from the view described previously. This view shows a list of all the butchers related to the selected date. Corresponding to each butcher is a link providing the manager the opportunity of resolving the potential conflicts at another page. When the conflicts of a butcher have been resolved, this will be indicated to the manager such that he knows which butchers have remaining conflicts.

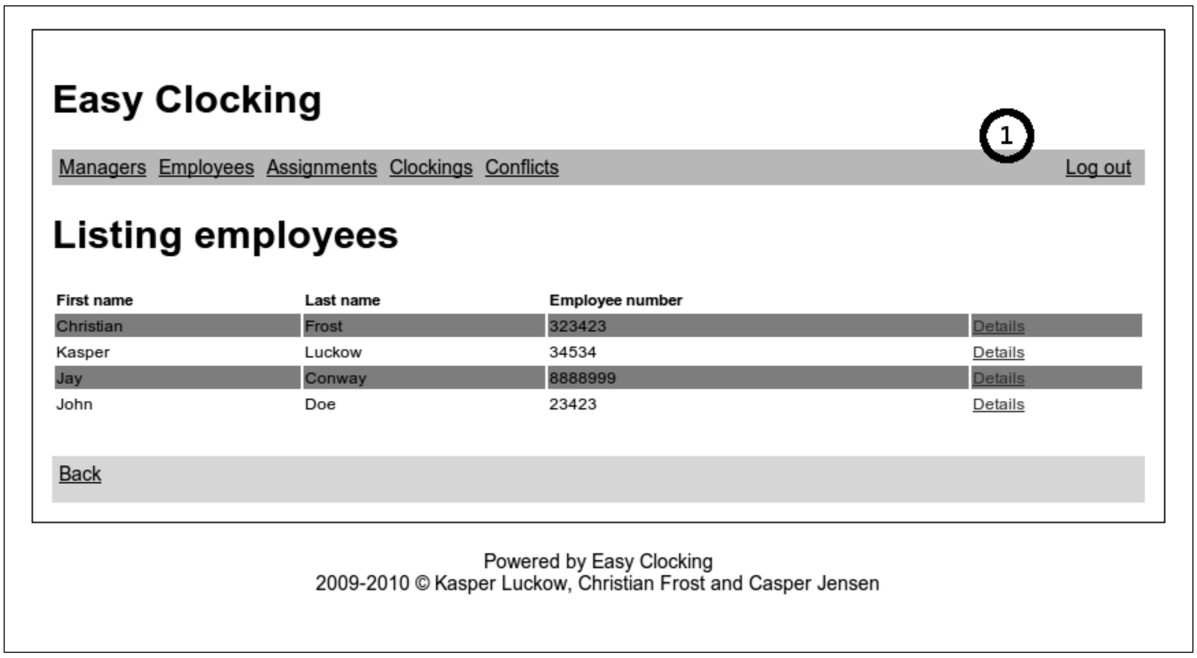
`Resolving Conflicts` This page is redirected to when the manager chooses to resolve conflicts for a particular butcher. The page shows all the recorded clockings associated with the butcher. Consecutive clockings, that is, clockings that can be chained are ordered into collections which are indicated on the page to give a better overview. Conflicts related to clockings overlapping in their time intervals, have a link next to each other which indicate the clockings of the list they are conflicting with. The manager can select clockings that should be committed to the system, provided that they do not conflict. Furthermore, the view also displays conflicts regarding a butcher being at too few and too many workstations according to his assignments. In this relation, functionality is provided to resolve these.

Code Example

In the design of our web application, we have used the design principles and Gestalt effects as described in Chapter 6. We have focused on applying these to the pages related to conflict handling, especially `Resolving Conflicts`. In the following, we will describe the design of `Conflicting Butcher Selection` and `Resolving Conflicts` in relation to the functionality and usability of the pages.

Furthermore, we used the abstract widgets design, described in Section 10.4, to know which widgets were needed on `Resolving Conflicts`.

Conflicting Butcher Selection When a manager selects a day to solve conflicts for, a list of employees registered to work that day is presented. A screenshot of this page is shown in Figure 12.1. Each employee is listed, and if an employee has a conflict, the corresponding row will be marked by a red colour. At the far right, a link allows the manager to navigate to `Resolving Conflicts`.



Listing employees

First name	Last name	Employee number	
Christian	Frost	323423	Details
Kasper	Luckow	34534	Details
Jay	Conway	8888999	Details
John	Doe	23423	Details

Back

Powered by Easy Clocking
2009-2010 © Kasper Luckow, Christian Frost and Casper Jensen

Figure 12.1: Screenshot of a list of employees registered at the location sensors for a specific date. Employees highlighted with red have conflicts.

We have applied some of the Gestalt effects on our menu (1), such as the proximity effect by grouping menu items related to each other and the closure effect by encasing the menu in a coloured box. This makes it easier to identify which items are part of the menu. We have divided the normal navigation items to the left and *log out* to the right of the menu, since *log out* is an action that is not related to navigation.

Design principles such as visibility have been used when deciding how links are displayed, which all are shown with an underline and either black colour for the menu or blue for the content. The black colour is used for the menu since blue clashes with the orange colour of the menu. Only links have underlines and no other elements use this visual indicator.

Also, the consistency design principle has been used in the usage of colours, placement of elements and usage of visual clues. E.g. by always using the same variant of red to mark conflicting items or by placing the menu and headings at the top of the page.

Listing 12.5 contains the view used to render `Conflicting Butcher Selection`. A table is rendered by looping over an array of employees, printing their name, employee number and constructing a link to `Resolving Conflicts`.

```
<h1>Listing employees</h1>

<table>
  <tr>
    <th>First name</th>
    <th>Last name</th>
    <th>Employee number</th>
  </tr>

<% @employees.each do |employee| %>
  <tr <%= conflict_class(employee) %>>
    <td><%=h employee.first_name %></td>
    <td><%=h employee.last_name %></td>
    <td><%=h employee.employee_number %></td>
    <td><%= link_to 'Details', :controller=>'conflicthandler', :action => 'employee', :
        id=>employee.id %></td>
  </tr>
<% end %>
</table>

<div id="controlbar">
  <%= link_to 'Back', :controller=>'conflicthandler', :action => 'index' %>
</div>
```

Listing 12.5: View for displaying a list of butchers and showing if they have a conflict for the given day.

Resolving Conflicts Figure 12.2 shows the page used to resolve conflicts. On this page, we display the three types of conflicts are displayed.

When a conflict is present, a warning will be shown in a red box (1). Otherwise, if no errors are present, the box is green. In it, the types of present conflicts are listed, and a link is provided which scrolls the page down to that group of conflicts and highlights them by fading them in and out of the page a couple of seconds. This allows the manager to quickly locate the particular conflict and navigate to it. Highlighting the elements supports the feedback design principle, by showing something to the user when clicking the link to the conflict. JavaScript is used to create the visual effects, in addition to various other visual effects and client-side logic.

In the same manner, a link is shown at each time conflict (2) which, if pressed, highlights the collection and all other collections of clockings it is conflicting with. All collections part of a conflict, are coloured red.

All clockings part of a collection are hidden, and can be shown by pressing the folder icon located to the left of each collection (3). This is to reduce the amount of information shown on the page such that the user quickly can find relevant information. When clicking the icon, the clockings fades in, and out when clicked again. Listing 12.6 shows the highlight function which takes an array of elements which needs to be highlighted. This function is called by an event handler on the *show conflict* links.

```
var highlighted_lines = new Array();
function highlight_conflicts(highlight) {
  start_color = '#FF5A5A';
  end_color = '#ffff99';

  // disable highlights
  i = highlighted_lines.length;
  while (i > 0) {
    i--;

    if (highlight.indexOf(highlighted_lines[i]) != -1) {
      // line already highlighted
    } else {
      // fade out
      new Effect.Highlight($('collection_' + highlighted_lines[i]),
                          { startcolor: end_color,
                            endcolor: start_color,
                            restorecolor: start_color});

      highlighted_lines.splice(highlighted_lines.indexOf(highlighted_lines[i]));
    }
  }

  // enable highlights
  i = highlight.length;
  while (i > 0) {
    i--;
    new Effect.Highlight($('collection_' + highlight[i]),
                        { startcolor: start_color,
                          endcolor: end_color,
                          restorecolor: end_color});

    if (highlighted_lines.indexOf(highlight[i]) == -1) {
      highlighted_lines.push(highlight[i]);
    }
  }
};
```

Listing 12.6: The function `highlight_conflicts` takes an array of collection ids and highlights their rows.

The function `highlight_conflict` can be split into two main parts. The first part, starting at line 6, iterates over a list of already highlighted lines, checking in line 11 if they should be highlighted again, and if not, line 15, removes the highlight effect.

The second step, starting at line 24, highlights all lines in the highlight list. This creates the effect of the row changing colour. If the line is already highlighted, the colour will flicker shortly. The highlight effect is applied in line 28.

It is possible to remove collections of clockings or individual clockings from the set of used clockings by toggling the checkbox to the right at each row (4). Furthermore, it is possible to discard all clockings or individual clockings by either toggling the checkbox besides the collection or the clocking itself. JavaScript is used to improve feedback when toggling the checkboxes by changing the state of other checkboxes if needed. E.g. if a checkbox next to a collection is unchecked, then all clockings in that collection are unchecked.

12.3.3 Controller

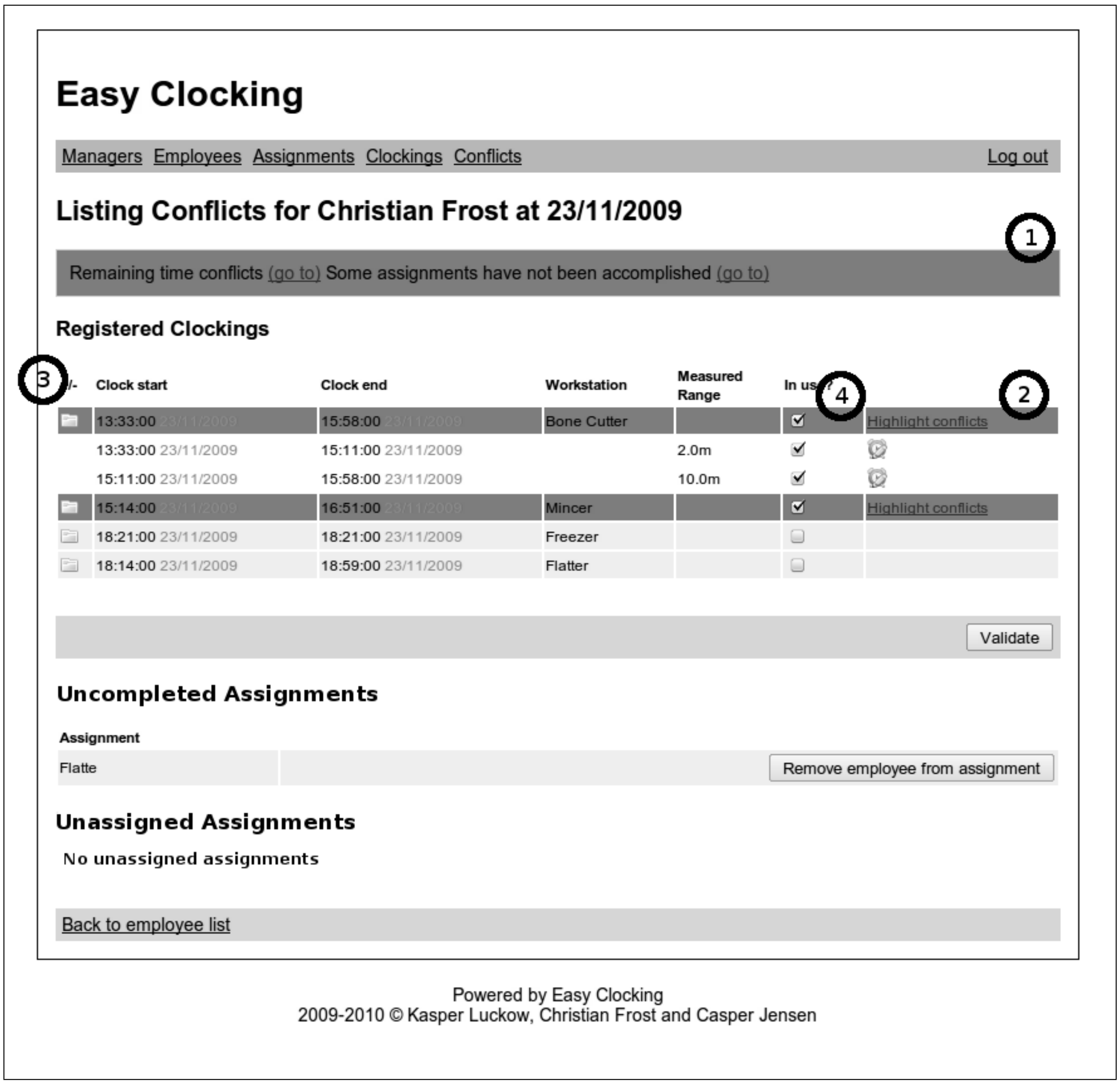
Before implementing the controller part of conflict handling, we identified which methods it should include. First of all, a method for each view is required in order to handle requests to the view. Furthermore, we used the work flow of conflict resolving, described in Section 10.3, to identify what functionality the controller should provide. As shown in Figure 10.3, the controller must contain logic to detect the presence of three types of conflicts.

In the following we have listed the methods identified for the controller.

`index` This method is empty, since all that was needed, was the possibility of selecting a date to be used for the employee index view and thus this only needs to specify which view to render.

`employeeindex` Finds all clocking and assignment information for the given date. This information is then passed to the different conflict methods, to detect the presence of potential conflicts. It instantiates an array of butchers who have conflicts in their information. This array is then used by the view to present the manager with a list of butchers for whom he must resolve conflicts.

`employee` This method is almost similar to `employeeindex`, but it instantiates multiple arrays that are used by the view to present the conflict information. E.g. it instantiates an array holding the uncompleted assignments.


Managers Employees Assignments Clockings Conflicts
Log out
1
2
3
4
Validate
Remove employee from assignment
Back to employee list

Easy Clocking

Listing Conflicts for Christian Frost at 23/11/2009

Remaining time conflicts (go to) Some assignments have not been accomplished (go to)

Registered Clockings

/-	Clock start	Clock end	Workstation	Measured Range	In us	
📁	13:33:00 23/11/2009	15:58:00 23/11/2009	Bone Cutter		✓	Highlight conflicts
	13:33:00 23/11/2009	15:11:00 23/11/2009		2.0m	✓	⏰
	15:11:00 23/11/2009	15:58:00 23/11/2009		10.0m	✓	⏰
📁	15:14:00 23/11/2009	16:51:00 23/11/2009	Mincer		✓	Highlight conflicts
📁	18:21:00 23/11/2009	18:21:00 23/11/2009	Freezer		☐	
📁	18:14:00 23/11/2009	18:59:00 23/11/2009	Flatter		☐	

Uncompleted Assignments

Assignment	
Flatte	Remove employee from assignment

Unassigned Assignments

No unassigned assignments

Powered by Easy Clocking
2009-2010 © Kasper Luckow, Christian Frost and Casper Jensen

Figure 12.2: Screenshot of a conflict page for an employee for a given work day. The page allows a manager to resolve possible conflicts.

Code Example

Listing 12.7 shows the controller used to render the view shown in Listing 12.5. The method `employeeindex` takes a date as input through a post request and shows a list of employees.

```
   def employeeindex
     session[:date] = params[:date] if params[:date]

     if session.has_key?(:date)
5      @date = Date.parse(session[:date])
       datetime = DateTime.parse(@date.to_s)
       assignments = Assignment.find_all_by_date(@date)
       clockings = Clocking.find_all_by_clock_start(datetime..(datetime+1))

10     @employees = Conflicthandler.getemployees(clockings,assignments)

       if request.post?
         Conflicthandler.discardnonassignedclockings(@employees,clockings,assignments)
       end
15
       @conflicting_employees = Conflicthandler.conflicting_employees(@employees,
           clockings, assignments)
     end

     if session.has_key?(:date)
20     respond_to do |format|
         format.html
       end
     else
       flash[:notice] = ’Please select a date’
25     redirect_to :controller=>"conflicthandler",:action => "index"
     end
   end
```

Listing 12.7: Method responsible for rendering the `Conflicting Butcher Selection` view.

In line 4 a check is made to ensure a date has been selected. If not, the body of the method is skipped and in line 25 the user is redirected to a page where he can select a date.

In line 12, a check is made if the HTTP request was a post. If so, the method tries to resolve some of the conflicts automatically by discarding clockings from assignments for which the butcher has not been assigned.

If a date is selected, a list of assignments is fetched for the date in line 7 and a list of clocking collections is fetched in line 8. In line 10 and 16, a list of employees and conflicting employees is generated on the basis of the list of assignments and clocking collections. These are saved in an instance variable, making them available to the view when rendered.

13 Testing

In this chapter, we describe how Easy Clocking has been tested. Initially, we argue for our choice of testing methods and, afterwards, we describe how the tests were conducted.

13.1 Test Methods

As described in our non-functional requirements in Chapter 3, we did not want to emphasise on testing when creating the product. Because of this, we decided mainly to focus on automated acceptance tests to ensure that the functional requirements had been implemented correctly. Additionally, some of the acceptance tests conducted reflect integration testing. Integration tests are mainly performed when testing the communication path between the location sensor and the web application. Furthermore, the tests helped in verifying that the reliability quality factor was achieved.

Using these automated acceptance test made it possible for us to, relatively fast, test if changes in the code had affected the functionality. Therefore, these tests were executed regularly during the project period.

If we had chosen to put more effort into testing, we would have made unit tests for all the methods which especially would have been an advantage if we were regularly re-factoring the code. However, since Easy Clocking was not the case since it was not meant to be deployed at this stage, unit tests were only applied to central parts of the code.

With respect to usability tests, we analysed different test methods. With a customer from a butchery, our plan was to do an Instant Data Analysis[Kjeldskov et al.] test in the usability lab with four to five managers using Easy Clocking. Nielsen [2000] describes that usability tests should not be made with more than five users since you learn less and less as you add more users to the test. Also, his research indicates, that using five users for tests, discovers approximately 84% of the problems. However, since we had no customer, we decided to not put much effort into usability testing. Instead, we chose to conduct a less time consuming test as described by [Krug, 2005, c. 9] who states a more informal approach. The main difference is that the method does not require much planning, other than loosely deciding what should be tested, and the pre-processing of the test data is described to be done during a lunch talk.

Also, [Krug, 2005, c. 9] mentions that one should remember that doing a low-cost usability test, is better than not doing anything.

When developing a web application, an important part is to verify that valid HTML is generated. Therefore, we also wanted to test this.

In order to ensure that the location sensor software can successfully operate in the available memory, we tested both the memory footprint and if memory leaks were present.

To examine that the RSSI ranging technique was usable for Easy Clocking, we conducted experiments to show whether RSSI could be used for determining distance in practice or not. If such a relation existed it would give us the possibility to measure the RSSI and then calculate the distance between the tag and the location sensor.

13.2 Automated Acceptance Tests

Ruby on Rails provides a test framework, which we have used to create automated acceptance tests. The tests primarily focus on using HTTP methods to simulate how the web application is used. An example of such a test is shown in Listing 13.1.

```
test "should create clocking" do
  login_as :web1
  assert_difference('Clocking.count') do
    post :create, :clocking => {:clock_start=>"2009-11-10 10:06:14", :clock_end
        =>"2009-11-10 11:06:14", :average_distance=>"15", :tag_mac=>"AA:FF:AA:FF:AA:FF",
        :workstation_id=>workstations(:work1) }
  end

  assert_response :success
end
```

Listing 13.1: Acceptance test simulating a location sensor posting clocking information.

As shown, the first thing conducted in the test case is to login as a web service. Afterwards, clocking information is posted and if the total amount of clockings in the database increases by one, the test succeeds.

Similarly, we did automated acceptance tests for the location sensors. For this, we used a test framework called CUnit[1] which allowed us to create automated acceptance tests. An example of such a test case is shown in Listing 13.2.

```
void test_caching(void) {
    employee_str *employee;
    clock_str *clock_iter;
    const char *original_base_url;
    int clock_num;

```

[1]cunit.sourceforge.net

```
    original_base_url = get_base_url();

    set_base_url("http://wrongbaseurl");

    employee = insert_employee_in_list("AA:BB:CC:DD:EE:FF");
    for(clock_num = 3; clock_num > 0; --clock_num) {
        employee->clock_str = insert_clocking_in_list(employee->clock_str, 22);
        employee->clock_str->clock_start = time(NULL);
        employee->clock_str->clock_end = (time_t)((int)time(NULL) + 100);
    }

    send_employee_clockings();

    clock_iter = employee->clock_str;
    for(clock_num = 3; clock_num > 0; --clock_num, clock_iter = clock_iter->next)
        CU_ASSERT_PTR_NOT_NULL(clock_iter);

    set_base_url(original_base_url);

    send_employee_clockings();

    employee = is_employee_in_list("AA:BB:CC:DD:EE:FF");
    CU_ASSERT_PTR_NULL(employee);
}
```

Listing 13.2: Acceptance test testing the caching functionality of the location sensor.

First, the URL to the server is set to a non-existent server, such that the location sensor can not connect to it. This is done to simulate the situation where a connection between the web application and the location sensor can not be established. Next, a set of clocking information is added to an employee and afterwards attempted passed to the web application. Since the connection is not established, the clocking information will be left for later delivery. Hence, we assert that the amount of clockings are still present. Finally, the URL is set right, and the clocking information is passed to the web application.

13.3 Usability Testing

In the usability test, we asked two students from 7^{th} semester in Software Engineering who had no previous knowledge in Easy Clocking, to use it.

Since usability primarily was improved with respect to conflict handling, we set up test data to simulate different conflict scenarios which we wanted our test users to solve.

Before each test, we described to the test users what the purpose of Easy Clocking was. Afterwards, we asked them to do certain tasks, one by one. The introduction and the tasks are described in Appendix L.

13.3.1 Results

From the tests, we identified a number of usability problems. Some were related to the test users lacking insight into the work flow and terminology we assume the managers to know, while others could be mapped to the Gestalt laws and design principles described in Chapter 6. Because of time constraints, we selected only to correct those problems that would increase the usability while limiting the implementation time.

A list containing some of the errors and the changes we made is given below.

- The test users were in doubt what the purpose was of the *View conflicts* link, which was placed to the right of a conflict. Pressing the link highlighted other clocking collections which it conflicted with.

  We renamed the link to *Highlight conflicts* and changed the highlight logic to also highlight the clock collection itself. This way, it is more obvious that the link highlights some items, and now all clock collections involved in the conflict are highlighted and not only some of them.

- The test users were in doubt what the field *Measured distance*, for each clocking, was a measure for. One thought it was the distance the employee had moved while he was detected. They were also a bit confused since the number did not contain a unit.

  We added a unit to the number and renamed the field to *Measured range*, which we expect to help in the understanding.

- The test users were able to solve the time conflicts at the top of the page. However, if the corrected time conflicts introduced conflicts related to missing assignments shown lower on the page, they had difficulty in spotting these. The message at the top of the page was still indicating a conflict, which the test users were unable to identify.

  We changed the conflict message to include text describing which type of conflicts were present, and links moving the viewpoint of the browser to the conflict while fading the area containing the conflict in and out of the page. This should make it more obvious where the conflict is located.

We observed that many of the problems we encountered could be solved both by changing the user interface and by instructing the user in how the system works. The problems were not related to the inability of solving the problem, but how it was intended to be done. We changed many of the user interface elements in order to improve usability of Easy Clocking. However, we expect, based on Kjeldskov et al. [2005] who describes that some problems disappear over time because users find their ways through the system, that a manager would have learned his way around the system over time.

One large problem remaining, which we did not implement, was the complexity in establishing the necessary overview in order to fix the conflicts quickly. A possible solution would be to create a more visual representation of the conflicts, by showing the time intervals and overlapping time intervals by drawing lines on a timeline. A possibility would even be to allow the manager to use this diagram to solve the problem by manipulating the lines. A mock-up of this diagram is shown in Figure 13.1.

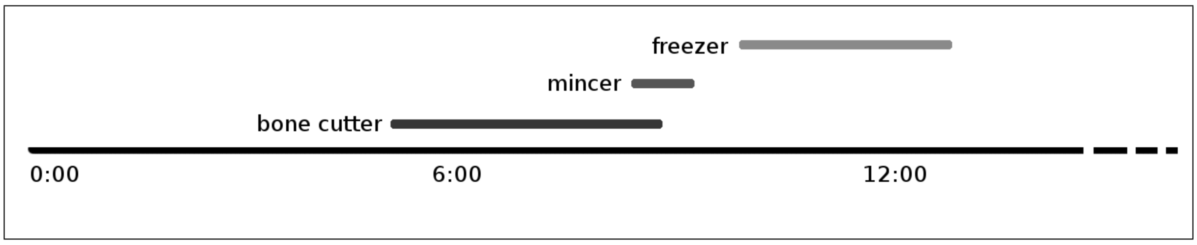


Figure 13.1: Mockup showing how time conflicts could be represented.

In the diagram, clocking information from the mincer and the bone cutter conflict.

13.4 HTML Validation

There are various reasons for verifying the validity of HTML. According to W3C [2009], the problem with invalid HTML is that different web browsers may parse the invalid HTML differently and hence style or layout of the parsed HTML can be presented differently. Conforming to the HTML standard, avoids these problems.

Our web application uses the HTML standard XHTML 1.0 Transitional. The Transitional flavour of XHTML provides the opportunity of making small adjustments to the mark-up which benefits older browsers that do not understand style sheets. There is also the strict flavour of XHTML which separates elements such as font, colour and layout effects to a CSS thus giving a mark-up which is free of any layout details.[W3C, 2009]

To validate that our generated HTML conforms to the Transitional standard, we made use of a plug-in for the Mozilla Firefox web browser called HTML Validator[2] which applies offline algorithms on the HTML of the visited page. The algorithms report errors if the pages contain HTML that is not conforming to the standard specified in the DOCTYPE element of the HTML document.

Verifying the validity of the generated HTML has been conducted by visiting all web pages comprised by Easy Clocking with the Firefox plug-in enabled. After this test, we conclude that all of our pages conformed to HTML Transitional.

[2] `addons.mozilla.org/en-US/firefox/addon/249`

13.5 Memory Consumption

In Chapter 11, we assessed that the 32 MB of available memory are enough for our application, but it is important to ensure that the memory footprint does not grow over time because of e.g. memory leaks.

In order to test for memory leaks in the application, we used the tool Valgrind[3] which can run an application and count the number of allocations and de-allocations of memory. The goal is to ensure that all allocations in the lifespan of an application are de-allocated again.

Running Valgrind on our location sensor software test-suite, we were able to detect some memory leaks which were removed. The end result of the memory-leak test was as follows:

Allocated	De-allocated
3.500	3,438

The end system has some allocations which have not been de-allocated. Analysing these with Valgrind, we concluded that these are allocated at application initialisation and used throughout its lifetime, such as configuration values being read into memory. We can use this information to conclude that, to our knowledge, the location sensor software is free of memory leaks.

13.6 Ranging Technique Test

The purpose of this Section is evaluate the accuracy of distance measurements calculated on the basis of RSSI values, in order to conclude if they are sufficient for Easy Clocking.

We start by describing a radio propagation model, used to calculate distance based on RSSI values. We then conduct an experiment to calculate the path loss exponent, a constant used by the radio propagation model. Using this, we conclude on the accuracy of the measured distances.

13.6.1 Radio Propagation Model

There exist different types of radio propagation models proposing a theoretical relationship between RSSI values and distance. A commonly accepted model is the log-distance path loss model, which is given by;[Papamanthou et al., 2008]

[3] `valgrind.org/`

$$P(d) = P(d_0) - 10nlog\left(\frac{d}{d_0}\right), d > d_0 \tag{13.1}$$

$$\Updownarrow$$

$$\frac{d}{d_0} = 10^{\frac{P(d)-P(d_0)}{-10n}} \tag{13.2}$$

$$\Updownarrow$$

$$n = \frac{P(d) - P(d_0)}{-10log\left(\frac{d}{d_0}\right)} \tag{13.3}$$

In the log-distance path loss model, $P(d)$ denotes the measured RSSI value at a distance d and $P(d_0)$ denotes a measured RSSI value at some reference distance d_0. The RSSI value at the reference point can be determined through experiments. Typically this RSSI value is measured at one meter.[Rappaport, 2002, c. 4] Finally, n represents the path loss exponent, which is an expression of the rate of which the received RSSI decreases in terms of distance.[Mao et al., 2007][Rappaport, 2002, c. 4] Its value can be determined experimentally.

As shown, the model proposes that the relationship between RSSI and distance is logarithmic. Logarithmic growth is very slow meaning that at some distance, a possible increase of the distance will not influence the RSSI significantly. Furthermore, the RSSI values obtained by our equipment are expressed as integers, and therefore, we anticipated problems because after a certain distance, the distance would need to be increased much in order to change the RSSI value.

13.6.2 Received Signal Strength Indication Distribution

As described, we wanted to determine the path loss exponent n experimentally. To do this, we first wanted to ensure that the theoretical radio propagation model conformed to practice. This was done by first examining how obtained RSSI values are distributed when measured from predefined distances. From this, it can be concluded whether or not a single RSSI value is sufficient to calculate a distance to a butcher. If e.g. all the measured RSSI values at a distance are equal, only one measurement would be necessary to calculate the theoretical distance. Next, we examined if the relationship between RSSI values and distances in our experiment followed the propagation model.

The experiment was conducted by placing the location sensor and the reference Bluetooth device about half a meter off the ground. Afterwards, the initial measurement was conducted with one meter between the Bluetooth device and the location sensor. Here, 300 RSSI measurements were captured by the location sensor and stored for later processing. This process was repeated for intervals of one meter up to a total of 20 meters. Conducting 300 measurements at each interval was considered sufficient based on the approach used by other similar projects. Specifically, the project by Bose and Foh [2007] and the project by Malekpour et al. [2008] conducted 80 and 1000 measurements at each interval, respectively. At the end, we had obtained a total of 6000 measurements which would be the base for determining the path loss exponent.

In order to show how the RSSI measurements of the experiment were distributed, we made histograms for each of the intervals. These show the frequency of the various RSSI values at the given distance. The histograms for all the intervals are shown in Appendix K.

Tadakamadla [2006] and Bose and Foh [2007] have done similar research and conclude that RSSI values are normally distributed. We want to verify that this is also the case for our experiment. If we can verify this, then it can be used as an indicator that our experiment is valid and that we can use the measurements for making conclusions about the value of the path loss exponent.

The normalised histogram depicting each of the occuring RSSI values as a percentage of 300 measurements, measured at a distance of 4 meters, is shown in Figure 13.2.

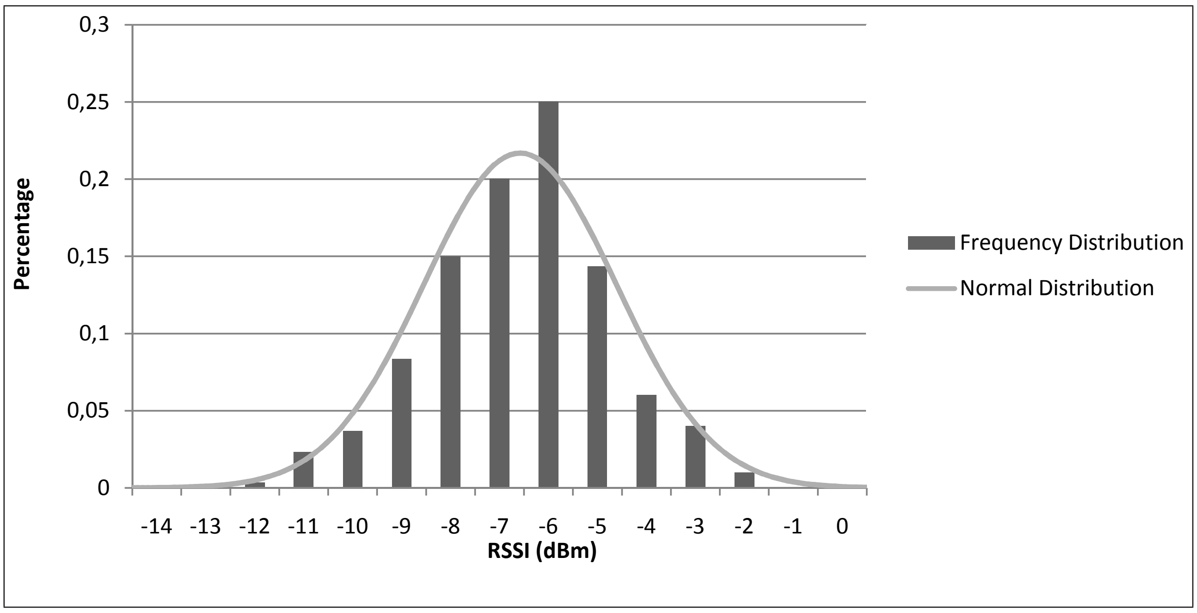


Figure 13.2: Frequency distribution and normal distribution of 300 RSSI values measured at a 4 meter distance.

As shown, we have, besides the histogram, depicted a curve approximately resembling the frequencies of each RSSI value. If this curve is a bell-shaped curve[Rumsey, 2003] the data is said to be normally distributed. Rumsey [2003] describes the properties of a bell-shaped curve to be the following:

- The curve must be symmetric.
- It must have a top in the middle, with tails to the left and right.
- The mean is in the middle of the shape.
- 95% of the data is contained within two standard deviations of the mean.

It can be difficult to determine if the properties hold, since it depends on graphically interpretation of the curve. Another way of determining if the RSSI values are normally distributed is to do a normal probability plot.[Thomas A. Ryan and Joiner, 1976] Hesse [1998] has given an example of the procedure of doing this which we followed to create the graph shown in Figure 13.3.

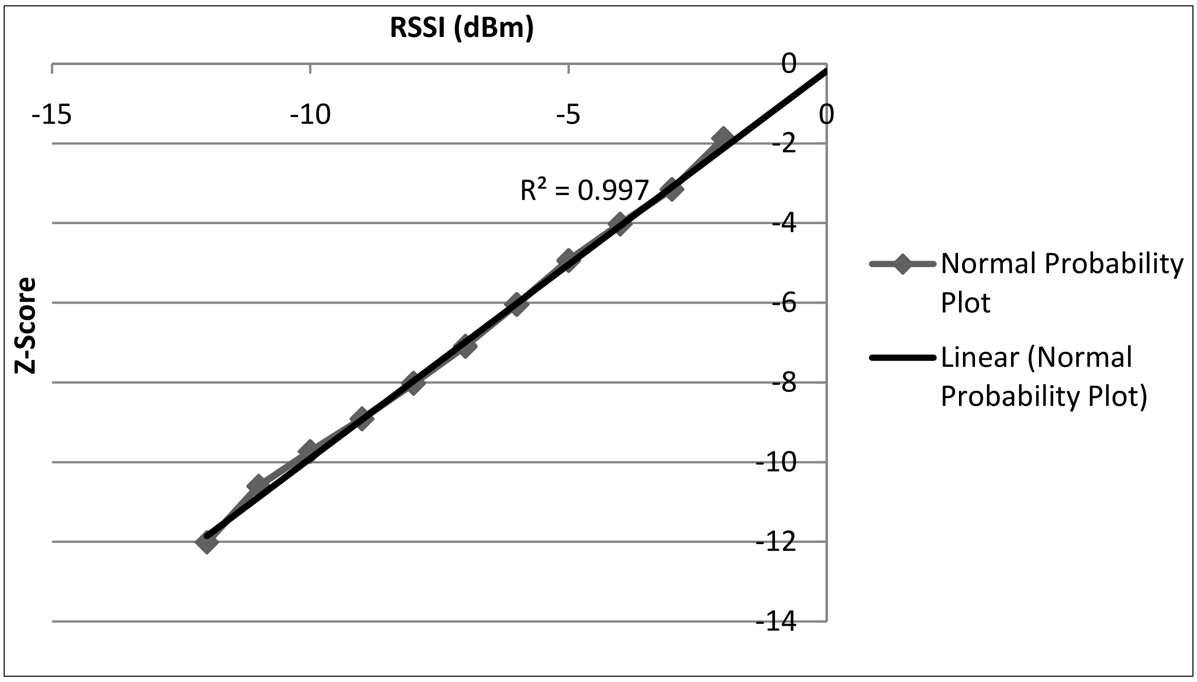


Figure 13.3: Normal probability plot using RSSI measurements at 4 meters distance.

As shown the graph is approximately a constant line which is confirmed by the correlation coefficient of 0.997[Weisstein]. As defined by Thomas A. Ryan and Joiner [1976] this means that the RSSI values are normally distributed.

Now it is possible to depict the standard deviation, which is an expression of how much the values varies from the mean, for each distance to show if it fluctuates at certain distances. If it does, this would result in having many unreliable RSSI measurements at certain distances thus making it difficult to calculate a distance using these. The standard derivation is an expression of how much the values varies from the mean and is computed using the following formula:

$$\sigma = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \mu)^2} \tag{13.4}$$

Here, it is assumed that a random variable X has mean μ and takes random values from a finites data set $x_1, x_2, \cdots, x_n$ with each value having the same probability of occurring. μ is computed as:

$$\mu = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i \tag{13.5}$$

As shown the mean is defined as what would normally be referred to as the arithmetic average.

In Figure 13.4, we have depicted the standard deviation computed for each interval. As shown, connecting the standard deviations, approximately resembles a constant line. This indicates that the experiment has not been affected by environmental changes when the distance between location sensor and Bluetooth tag was increased. The span of standard deviation is from 1.35 to 2.17 which conforms to the project by Bose and Foh [2007] who obtained similar results.

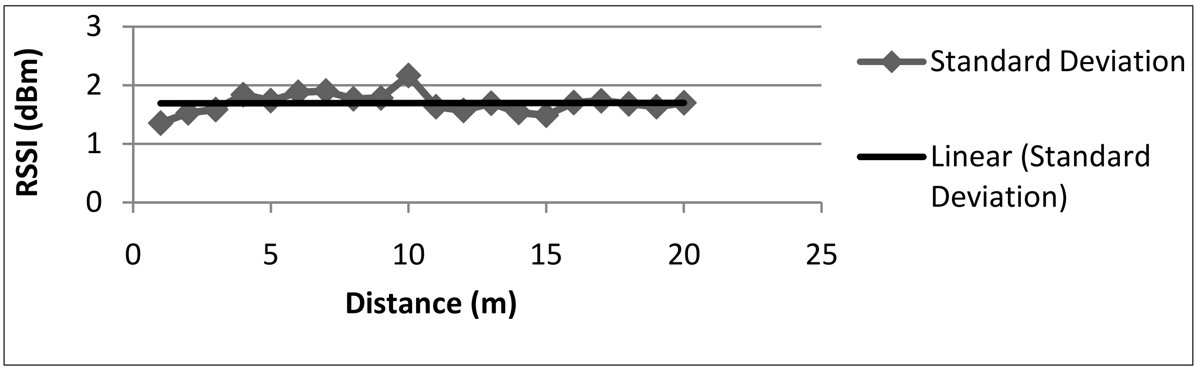


Figure 13.4: Standard deviation, with corresponding linear trend line for distances from 1 to 20 meters.

To verify that the measured RSSI values generally followed a logarithmic model, we depicted the measurements with a corresponding logarithmic trend line. Notice that each point is derived from taking the average of the 300 RSSI values at the particular distance. As shown in Figure 13.5, the measurements tend to follow this logarithmic growth, meaning that the measurements and the radio propagation model has the same tendencies.

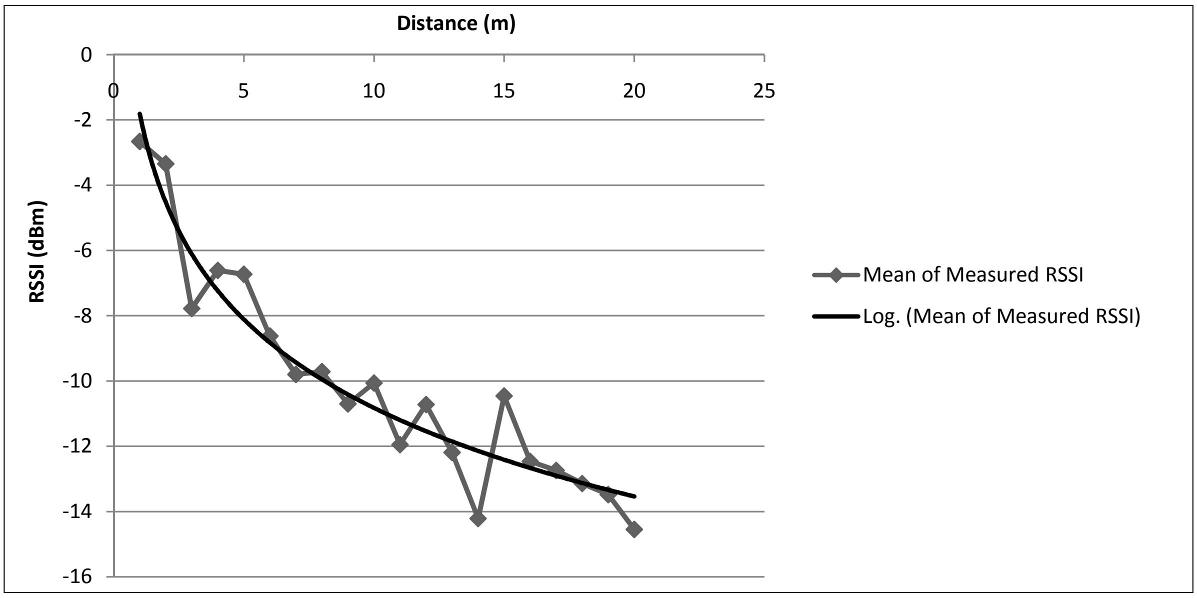


Figure 13.5: Mean of measured RSSI values.

13.6.3 Determining the Path Loss Exponent

To determine the path loss exponent, we initially calculated the path loss exponent for each of the RSSI values using Formula 13.3. Afterwards, we calculated the average of these which turned out to be 0.71. However, when later using this in the radio propagation model we got significant distance errors. Therefore, we decided to follow the procedure as described by Bose and Foh [2007], where the distance is calculated, given an RSSI value when using various path loss exponents. Depicting the resulting tendency lines makes it possible to identify the path loss exponent giving the best approximation to the reference graph. Figure 13.6 depicts the result of conducting this procedure in our experiment. The graph with no markers is the actual distances for RSSI values based on the data from Figure 13.5, and hence it is the reference graph. As shown, using 0.71 did not yield a graph closely resembling the reference graph. Instead, it turned out that adjusting the path loss exponent to 0.83 resulted in a path loss model fitting best.

According to [Rappaport, 2002, c. 4], the path loss exponent indoors should be around 2 to 3. From this, along with our measured path loss exponent, we concluded that determining the path loss exponent is connected with great uncertainty. Therefore, in order to use the radio propagation model in practice we expect that a path loss exponent should be calculated for each deployed location sensor. This complicates the deployment phase.

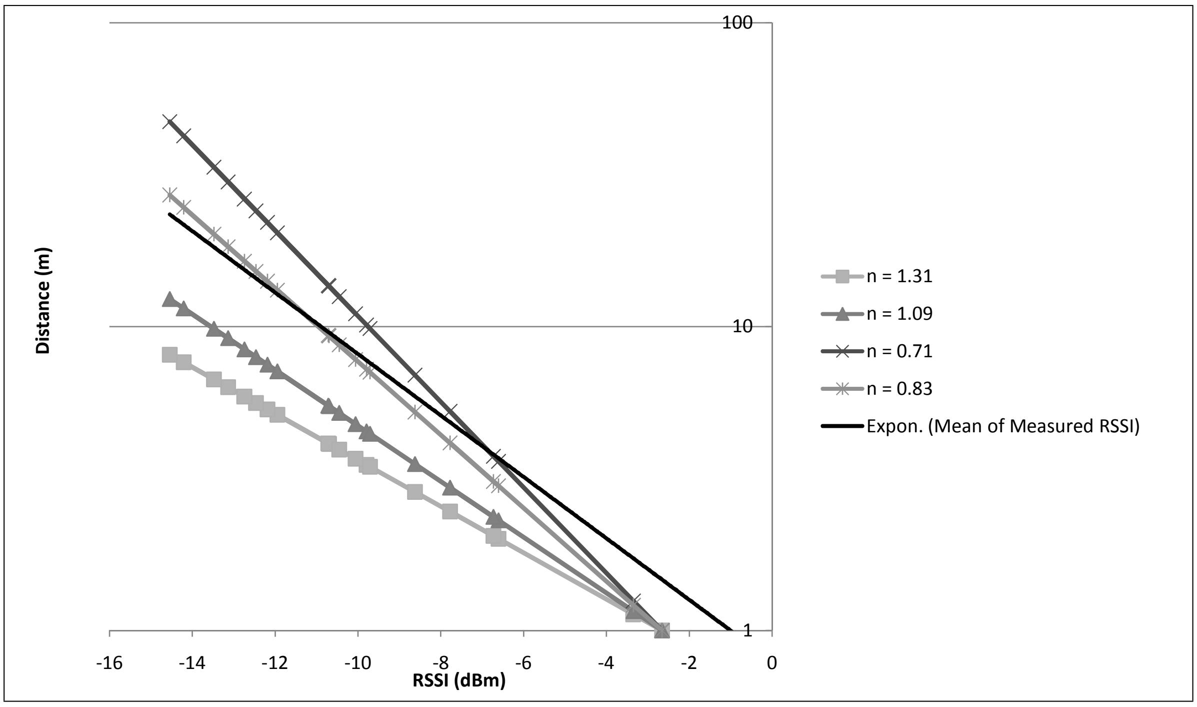


Figure 13.6: Trendlines using different path loss exponents depicted on a logarithmic scale.

13.6.4 Accuracy and Precision

Using the derived path loss exponent, we wanted to verify how often an accuracy of 5 meters was achievable.

According to Figure 13.2, we observed that the RSSI values were distributed around a mean. This meant that we could not expect to get the same RSSI values for multiple measurements at the same distance. Therefore, a way of minimising the deviations was needed. We experimented with calculating the average signal strength of 10 measurements and concluded that this was sufficient to compensate for erroneous measurements. To get a better result, more RSSI values could be taken into account, but this would be at the cost of time required to measure them. We assessed that 10 measurements for each butcher in the detecting range was possible within approximately 5 minutes for 10 butchers. Notice that since we have already shown that the RSSI values were normally distributed, it makes sense to calculate the average of a set of RSSI measurements, because they are equally distributed around the mean[Rumsey, 2003].

Figure 13.7 depicts the cumulated frequency of the accuracy calculated from measured RSSI values for distances from 1 to 20 meters. The graph was derived by calculating the difference of the calculated distance based on an RSSI measurement and comparing it with the actual distance. As shown, 87% of the calculated distances were within 5 meters accuracy.

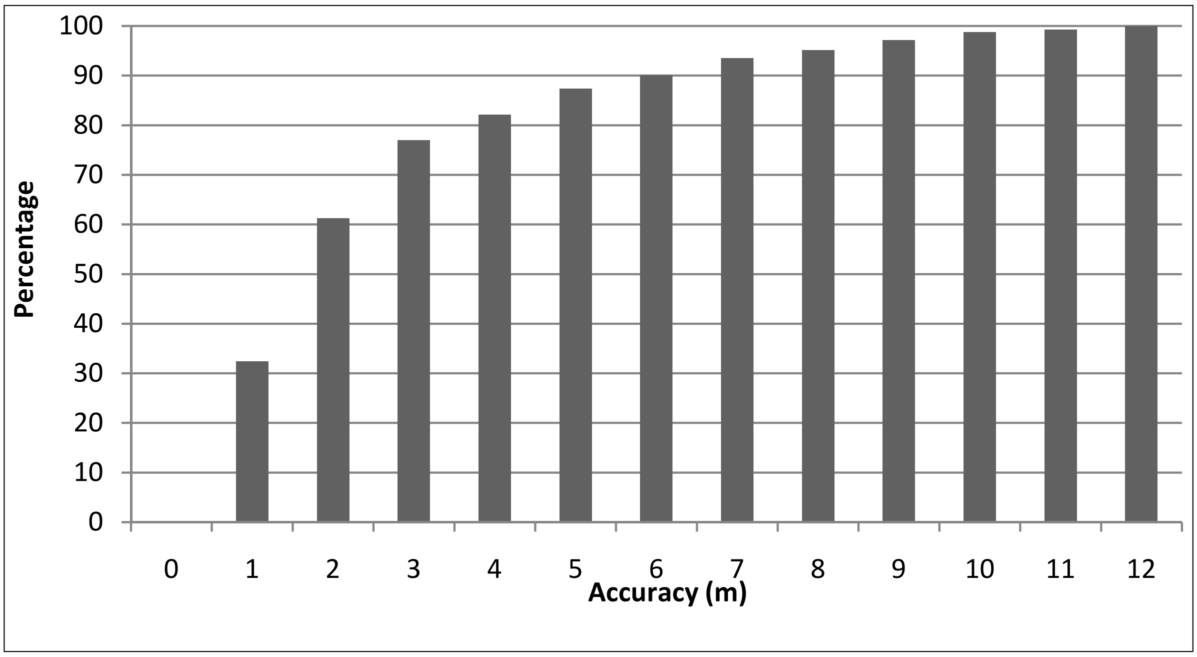


Figure 13.7: The cumulated frequency shows the percentage of distance measurements which lies within a given accuracy.

Chan et al. [2006] describes that by using Wi-Fi for positioning, an accuracy of 1 meter is achievable with a precision of 90% with noisy signals. Since we assume that these experiments have been more thoroughly conducted than ours and that filtering algorithms might have been used, we expected our precision to be lower. This is because we have not addressed issues influencing the experiment such as the propagation phenomenon called multipath fading. Multipath fading means that the radio signal takes different paths when propagating from the source. This can e.g. be caused by obstacles and humans, resulting in indistinguishable signals and hence the signal strengths.[Malik, 2009, c. 2][Bose and Foh, 2007][Tadakamadla, 2006]

In our opinion, the achieved precision is acceptable for detecting a butcher in favourable environments, but we do not expect this precision to hold in a practical deployment of Easy Clocking. In deployment environments there might be high humidity and metal surfaces which all affect the reliability of the RSSI values.[Tadakamadla, 2006] Our suspicion was confirmed by sampling measurements in other environments with more obstacles and interference, because these indicated that only a significant lower precision was achievable.

Part IV

Conclusion

14 Conclusion

Throughout this project, we have created Easy Clocking, a system for automatically clocking in and out employees based on their presence at workstations which can be used in many different of scenarios. It consists of a number of location sensors, placed at each workstation, which locates employees, and communicates their presence and location data to a web application. In our scenario, managers are capable of scheduling assignments to employees, verifying if they have worked on their assignments and resolve possible inconsistencies between the information registered by the location sensors and the assignments. The system is created with a goal of being flexible such that it can be applied to different companies.

We used a butchery as an example of a company using the Easy Clocking system, but it can be applied to any companies wanting to verify the presence of employees at specific points, in order to calculate salary, analyse work habits or track time usage on tasks and more. One example is the Danish home care, which have been criticised for their usage of clocking when visiting homes, since this process requires the employees to manually registering start and end times of tasks on a PDA[FOA, 2008]. Using Easy Clocking, this process could be automated, leaving the home care employees to spend more time on home care, and thus help the elderly, and spend less time on clocking. Another example is that statistics can be made on the work habits of the employees to plan assignments to prevent that employees conduct stagnant or repetitive work potentially having health implications.[Arbejdstilsynet, 2009]

One large advantage of Easy Clocking over manual clocking is the ease of use for employees, who do not need to interact with the system but are automatically detected. This avoids time overhead connected with manual clocking and human errors.

Using the butchery example, we created a set of functional and non-functional requirements for Easy Clocking, which we fulfilled during the project.

The driving force of the project was a set of learning goals which stated what we wanted to achieve and learn from the project while ensuring we covered the required topics from the study regulation. In the following each of the learning goals are concluded upon.

Gain knowledge of Internet technologies We have demonstrated knowledge of Internet, Internet services and Internet technologies by developing a web application aimed at being hosted on an Intranet of a company. The web application makes use of various technologies, namely: RESTful web services, XML, HTML, CSS, HTTP protocol, and sessions. Throughout the implementation, we documented our knowledge of these.

Usability The user interface of the web application was designed using various principles and techniques known from the field of Human Computer Interaction. Specifically, the Gestalt laws was harnessed by using the proximity and closure effects which enhance the usability of the user interface. Also, the user interface reflected usage of design principles such as consistency and feedback.

In order to verify the usability of Easy Clocking we made a usability test in which we discovered a number of usability problems with our initial implementation. We were able to recognise these from the usability principles, and correct a number of them in the final implementation.

New language and framework We gained experience in using a framework along with a new programming language for developing web applications. This was achieved by using the Ruby programming language and the Ruby on Rails framework. These had a steep learning curve and required more time to learn than we had anticipated. Regardless, the separation of concerns through MVC in the framework showed to be beneficial because it provided a good overview of the application.

Localisation Technologies and Techniques We gained theoretical knowledge of different localisation technologies and techniques. After conducting an analysis of these, we were able to compare them and determine the most suitable for Easy clocking. We based our choice on a set of criteria, such as price and detecting range, our requirements and scenario.

We chose to use the Bluetooth technology using the Received Signal Strength Indicator (RSSI) technique for determining distance from a location sensor to a Bluetooth device. We gained practical knowledge in using both of these in our implementation of Easy Clocking.

For RSSI, we showed that using a radio propagation model to calculate the range was applicable on our hardware under favourable conditions in the environment. However, it was later discovered that the distance measurements fluctuated in environments with great interference and obstacles. This indicated that this should be solved in future development, e.g. by using another ranging technique, filtering the measured values or by using a positioning technique.

Development Method As development method, we chose to tailor the agile development method Scrum which was a success. We gained a lot of experience in agile methods by using Scrum, and will in later projects use parts of it such as burn down charts, user stories and time estimations, while altering some parts of the process to better fit the requirement of conducting an academic report.

Real customer To make the product even more realistic, we wanted to get in contact with a butchery to analyse the work flow of the clocking procedure further. Also, we wanted to present our evolving product to them to get feedback to form the development in the direction of their possible needs. We ensured that the project did not depend on this contact, by defining our understanding of the clocking procedure and creating personas to define the target group. This meant that the only loss was the feedback. This could have been provided by other students, but we did not do this since the main purpose of talking to a butchery was to make a product that could be used in real-life.

We contacted the person from the Danish butchery Danish Crown, dealing with student contacts. Unfortunately, after a few conversations our phone calls and e-mails were not replied to. Furthermore, we contacted them through their website, but similarly, we got no response. From this, we learned that in upcoming projects, we should be more persistent when contacting a potential customer and it should be clarified early in the project whether the customer will participate or not. Also, we learned that later projects should not be dependent on customer contact.

Embedded software A large part of the project constituted developing the embedded software. Beforehand, none of us had experience with this type of development and this resulted in many observations which we had not anticipated. Specifically, we observed that the realisation of the location sensor was relatively difficult compared to the same development for a regular PC. The firmware used on the location sensors, OpenWRT, provided limited hardware support and lacked documentation which made hardware support for our Bluetooth device a difficult process, since we needed to install the needed packages and configure the software ourselves.

Moreover, many of the needed software packages, for instance libCURL, were over a year older than the latest releases. This gave rise to some confusion because the documentation did not conform to the version of the package available and hence the software utilisation was a time consuming process. From these experiences, we can conclude that one must thoroughly examine the possibilities and limitations the embedded device offers in terms of both hardware support and additional needed software.

Plug-in The web application used a plug-in architecture which was designed in such a way that Easy Clocking could be used in many different companies. The plug-in architecture makes it possible for a company to implement their own business logic which utilises the data made available by the kernel of the system. However, due to prioritisation, we did not explore all solutions for implementing a plug-in architecture but instead focussed on relatively few. Moreover, we did not include considerations in our plug-in architecture about the procedure for actually deploying Easy Clocking in a company. For instance, there are considerations such as how the source code of Easy Clocking is prevented from being directly read after being deployed. In this relation there are also considerations about how to limit access to the entire source code when extending it with customised plug-ins. Because these considerations have not been included, we have not successfully achieved the learning goal concerning making the application flexible to accommodate the various needs of different companies.

An important lesson learned through this project was that the number of areas of concern should be reduced when they comprise great uncertainty in terms of knowledge. In this project, we have touched upon many different areas without having prior knowledge and in addition, we chose to work in a small group for the first time. Due to these factors, we had difficulties in correctly assessing the workload. In the future, a better approach would be to focus on relatively few areas and structure the project such that the scope of the project could be changed more easily. Additionally, conducting a risk analysis identifying potential issues, dependencies among different components and the magnitude they negatively impact the project if not being completed could be convenient to aid identifying problems ahead of time.

The next step for Easy Clocking is to focus on improving the capabilities of the localisation component. A solution to this problem, will significantly improve the applicability of Easy Clocking as an automatic clocking system. Furthermore, some considerations remain to be addressed with regards to the implemented plug-in architecture.

15 Discussion

The purpose of this chapter is to comment on some of the aspects in the development which could be made differently. Also, we assess how these aspects would have affected the project. Furthermore, the chapter clarifies the aspects that have not been taken into account but are essential if Easy Clocking was to be deployed in a production environment. Finally, the purpose is to describe some of the problems encountered during the development process.

Ruby on Rails We expected some difficulties in using Ruby on Rails since the framework with corresponding programming language was new to us. We chose it because we wanted a challenge in web development while learning a new programming language and framework. After reading some documentation, we were able to use the scaffolding concept, which made it possible to get a working prototype of the web application early in the project. In our opinion, using scaffolding is a great advantage since a lot of trivial code is auto generated. However, in the beginning of the project, it was difficult to understand how the auto generated code worked. Also, the policy of convention over configuration made learning difficult since some of the conventions were not intuitive.

Ruby on Rails uses pluralisation of nouns, e.g. to indicate if an attribute refers to one or more models. We had a lot of trouble getting used to its semantics, which in our opinion conflicted with the Ruby on Rails principle of least surprise, described in Section 7.2.2. Also, we discovered that iterating over entities in models could be done using different syntax which again conflicted with this principle.

In some sense, Ruby on Rails 2.3.4 seems immature, because we at several occasions wrote code, primarily interacting with the models, which logically should be correct but was not. In finding solutions to these problems, we found the documentation insufficient, and used great effort in searching for alternative solutions.

Despite the previous critique, we do not want to leave out Ruby on Rails as a candidate for web development in later projects, because we now know some of its peculiarities.

Localisation Techniques We selected to use ranging as localisation technique to determine if an employee is present at a workstation. Even though this approach was possible in theory, it proved to be problematic in practice. We observed that the measurements were more dependent on the environment than expected. If we were to continue working on localisation of employees, we need to find a better way of doing this. A possible solution would be to address the issue of fluctuations by filtering the measurements. For this, a filtering algorithm such as the Kalman Filter can be applied[Helen et al., 2009]. Also, we could use other approaches for measuring distance such as Time of Arrival, or use positioning techniques in order to determine the employee's position with greater confidence.

Scrum Development Method Generally, using Scrum turned out to be a positive experience, even though there were parts of the method we would like to change or not use at all in future projects. One element we want to remove is the user roles, and most notably the Scrum master. In our project group, we have a more cooperative approach of managing the process which works flawlessly.

The parts we really liked, and were a big help in our project, were user stories, time estimates and burn down charts. Using these tools, we were able to more easily assess progress in contrast to previous projects where more traditional methods were used such as the incremental model. In the future, we hope to improve this part as we improve our time estimates, which we had some trouble getting right. We base these problems on lacking experience.

Our greatest problems with Scrum were its policy of not planning ahead when implementing user stories and our choice of not including tasks related to the creation of the report. Planning a bit further than only one user story at a time would have saved some time by reducing the amount of code written and later discarded in the project. We also observed that some conceptual problems were left unresolved until late in the project. These would have been revealed if we had designed more ahead of time. The separation of Scrum and the report was problematic, since we had difficulties in planning our sprints without taking the tasks related to the report into account. In the future, we should track both of these tasks in Scrum, treating them collectively as the final product.

The policy of not planning ahead also had its positive effects. It was easier to priorities user stories, concentrating on implementing the ones important to us, and discarding or reducing magnitude of user stories which we were unable to schedule in the time allocated for developing the software. These were discarded without having used any of our development time or having incorporated their design into the current software.

Concentrating on getting the most important and fundamental user stories ready first resulted in having a working and functional product early in the project. This was reassuring, since we had something finished in contrast to projects where we used a more traditional waterfall model, and the product was finished very late in the project.

Having a working product each iteration and showing it to a customer, collecting feedback on the product, would have given some of the agile processes more meaning, because a customer would very likely give us feedback prompting us to change our development plan. The lack of a customer resulted in a feature set nearly unchanged throughout the project since we had a fairly good view of our goals and desired features.

Usability and Usability Tests We tried to conduct an informal usability test on our product in the later stages of development. Here, we observed that with a small number of testers, we would get a lot of useful information about the usability of our product.

We also observed how a small number of quick changes to the user interface results in a substantial improvement in usability. A problem with only conducting a single usability test, was that we could not verify if changes made to the interface solved the problems.

If we were to focus on usability in a later project, we would conduct the same type of usability test multiple times throughout the project. This would give us even more feedback, allowing us to verify the impact of our changes and allow us to change the user interface conventions before the full user interface has been developed. It would also be possible that we could gather feedback on the product itself, and discover new features to implement.

Security Considerations There are multiple security mechanisms which undoubtedly would have been a requirement if the system was set into a production environment.

Currently, the Bluetooth devices are naively identified by their MAC-addresses without some supporting security mechanism. Hence, the system is vulnerable to exploits such as MAC-spoofing which is a relatively easy operation to conduct on a networked device. By using MAC-spoofing, an employee would be capable of stealing the identity of another employee which in the worst case could affect the salary system if this is based on the information generated by Easy Clocking. To address this issue, the MAC identification system should be supported by an authentication mechanism such as certificates.

In addition, the system lacks security in the communication paths. Currently, data sent via the communication path between the location sensor and the web application is sent in clear text, thus potentially being a victim of packet interception or injection for instance. This issue could be addressed by forcing encryption on all data sent over the communication path. Similarly, in relation to using the web application, an encryption mechanism such as SSL could be implemented to avoid similar problems.

Privacy Aspects Roy Want and Gibbons [1992] mention several considerations to be taken into account when deploying a localisation system. They describe how employees sometimes do not want to be tracked and that they should be given the possibility of not being registered. In Easy Clocking, this kind of freedom would not be possible since the purpose of the system is to replace the manual clocking procedures.

Another privacy aspect is to whom the localisation data is available. This means that the butchers must be assured that only the managers can access their data and that this data will not be used against them. This can be achieved by having certain management policies that must be fulfilled when handling the data[Roy Want and Gibbons, 1992].

If, when deployed, a great effort is put into introducing the advantages for each user group of the system, we expect that some of the privacy issues can be handled before they actually occur.

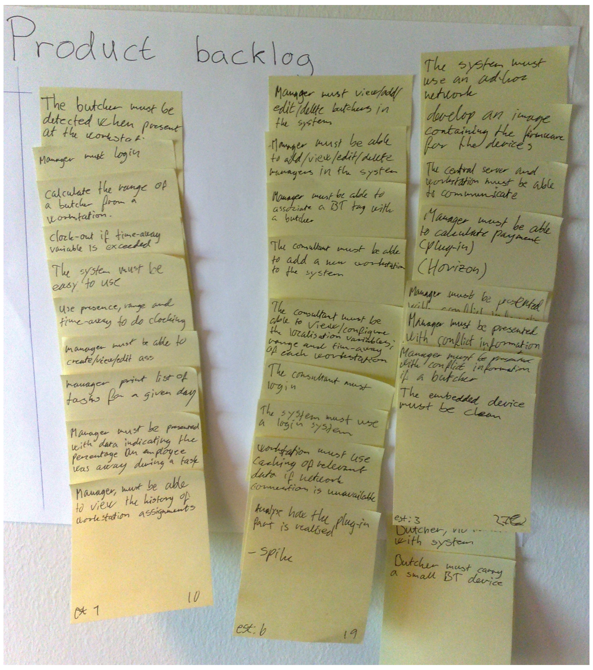
Part V

Appendices

Appendix Contents

A
Pregame

Figure A.1(a) shows our product backlog used in the developing process and Figure A.1(b) shows an example of a user story.


Product backlog

The butcher must be detected when present at the workstat.

Manager must login

Calculate the range of a butcher from a workstation.

Clock-out if time-away variable is exceeded

The system must be easy to use

Use presence, range and time-away to do clocking

Manager must be able to create/view/edit ass

Manager print list of tasks for a given day

Manager must be presented with data indicating the percentage an employee was away during a task

Manager, must be able to view the history of workstation assignments

est 1 10

Manager must view/add/edit/delete butchers in the system

Manager must be able to add/view/edit/delete managers in the system

Manager must be able to associate a BT tag with a butcher

The consultant must be able to add a new workstation to the system

The consultant must be able to view/configure the localisation variables; range and time-away of each workstation

The consultant must login

The system must use a login system

Workstation must use caching of relevant data if network connection is unavailable

Analyse how the plug-in part is realised

- Spike

est: 6 19

The system must use an ad-hoc network

develop an image containing the firmware for the devices

The central server and workstation must be able to communicate

Manager must be able to calculate payment (plug-in) (Horizon)

Manager must be presented

Manager must be presented with conflict information

Manager must be presented with conflict information if a butcher

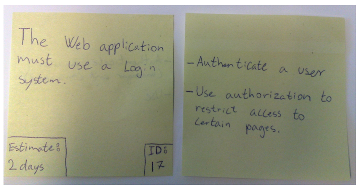
The embedded device must be clean

est: 3 27

Butcher, with system

Butcher must carry a small BT device

(a) Product backlog.


The Web application must use a Login system.

Estimate: 2 days

ID: 17

- Authenticate a user

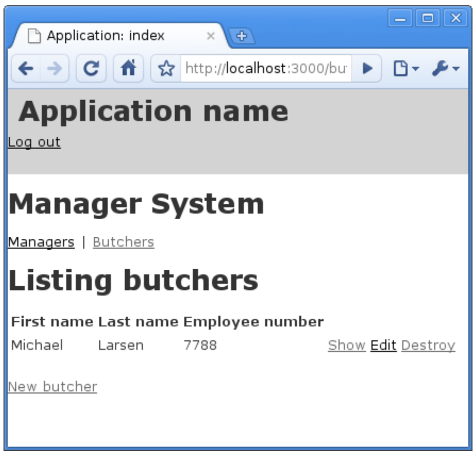
- Use authorization to restrict access to certain pages.

(b) Three sensors pinpointing a single position.

Figure A.1: User story defining the need of a login system, which is fulfilled when the acceptance tests are passed.

B
First Sprint

Figure B.1(b) shows a screenshot of the login page of the web application after the first sprint and Figure B.1(a) shows a screenshot of the page listing the butchers in the system. This page is viewable by managers.


Application: index

http://localhost:3000/bu

Application name

Log out

Manager System

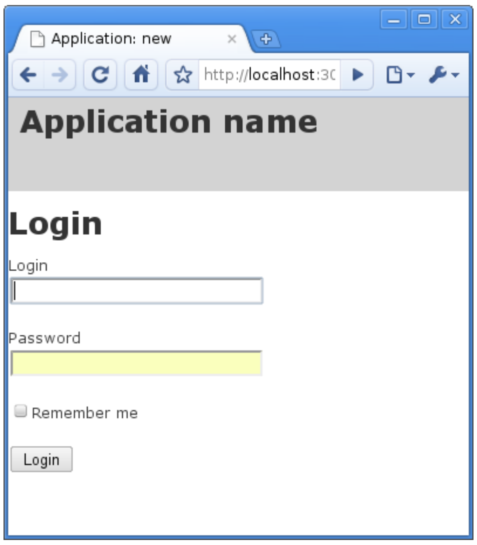
Managers | Butchers

Listing butchers

First name	Last name	Employee number			
Michael	Larsen	7788	Show	Edit	Destroy

New butcher

(a) The listing of Butchers after the first sprint.


Application: new

http://localhost:3C

Application name

Login

Login

Password

Remember me

Login

(b) View presenting the butchers of the system.

Figure B.1: Screenshots of the web application after first sprint.

C Third Sprint

Figure C.1 depicts the burn down chart for the third sprint.

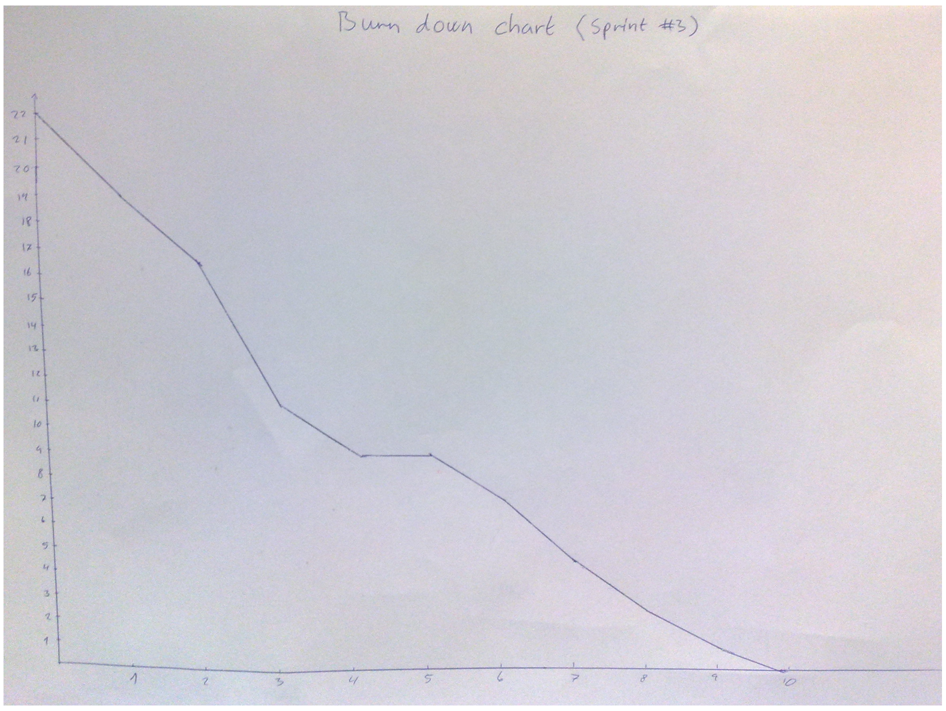


Figure C.1: Burn down chart for the third sprint.

D
Technology Evaluation Criteria

We have used the following criteria in the evaluation of the different technologies. The criteria were based on requirements we see as critical in order to accomplish our goals in the project and in order to support a company.

Interference When using wireless technologies with multiple location sensors in a given area, one must consider how interference of the location sensors might affect the localisation and potentially preventing positive identification of the tag. In general, a butchery is an indoor environment containing various electrical powered machines with motors. These generate electrical noise which potentially may interfere with the radio signals of the chosen technology. Additionally, many surfaces are made of flat metal which may result in radio signals interference. It must therefore be possible to achieve the other criteria under these conditions.

Power Consumption Power consumption should generally be a concern in modern projects since there is a public focus on environmentally friendly solutions. Furthermore, in our project where the employees need a remote device, such as a tag, it is important that the power consumption is low, such that the device does not require recharging during a work day.

Distance The reading distance from the tag to the sensor is of significant importance. If the reading distance is as low as a few inches or feet, it may not be usable because it can be difficult to detect the employees. To fulfil the requirements it must be possible to for a location sensor to read tags in a range of 10 meter.

Accuracy It is important that the range of a tag can be determined with a reasonable accuracy in order to do the clocking for the right workstation. The accuracy must be 5 meter. With this accuracy it is possible to tell which workstation is nearest to an employee. This accuracy is sufficient in order to achieve the system requirements.

Price Our solution for automatic clocking must be affordable for companies, meaning that the cost of deploying the system must not be too high. Therefore the choice of technology among others depends on value for the customer in respect to the cost. The hardware used at each workstation must not cost more than 1000 DK/Kr. And the tags used for identification must not cost more than 200 DK/Kr. With these prices, it would be possible to deploy the system on 10 workstations with 20 employees for 14.000 DK/Kr which in our opinion is affordable for a company.

Reliability The localisation of the employees must be built on a reliable technology in order to satisfy both the employees and the company. In an unreliable system an employee might have been working a given period of time without being registered.

E Quality Factor Definitions

Product Operations	
Correctness	Extent to which a program satisfies its specifications and fulfils the user's mission objectives.
Reliability	Extent to which a program can be expected to perform its intended function with required precision.
Efficiency	The amount of computing resources and code required by a program to perform a function.
Usability	Effort required to learn, operate, prepare input, and interpret output of a program.
Integrity	Extent to which access to software or data by unauthorised persons can be controlled.
Availability	The extent to which the program is operable in respect to the total running time.
Durability	The extent to which data is persistent when first stored in the program.
Product Revision	
Flexibility	Effort required to modify an operational program.
Maintainability	Effort required to locate and fix an error in an operational program.
Testability	Effort required to test a program to ensure it performs its intended function.
Scalability	Extent to which the program can handle information when the program is scaled according to some factor.
Product Transition	
Portability	Effort required to transfer a program from one hardware configuration and/or environment to another.
Reusability	Extent to which a program can be used in other applications related to the packaging and scope of the functions that the program performs.
Interoperability	Effort required to couple one system with another.

Table E.1: Quality factor definitions.[van Vliet, 2008, c. 6] Availability, durability and scalability are defined according to our understanding of them.

F Persona

F.1 Butcher

Personal Information Michael is 45 years old. He has short dark hair, brown eyes and a moustache. He is close to being overweight but is not concerned with it because he does not want to change dietary habits and start exercising. He lives about five kilometres away from the butchery where he works from 7:00 AM to 3:00 PM five days a week. Although he could use his bicycle for getting to work, he uses the car of the family. His family consists of his wife, Ann, 45 years old, whom Michael has known since his childhood. They have to daughters, Elisa and Sofie, aged 20 and 22, respectively. Elisa has just finished her education as a cook but due to limited job offerings, she has not found a job yet. Sofie is currently attending her fourth semester in philosophy at the nearby university. Both daughters are still living with their parents. Ann is working as a cashier at a grocery store.

When Michael gets home from work he enjoys watching TV. He has a passion for crime series. Besides, he also enjoys tinkering with small projects at home. He has recently built a new terrace in their backyard and is currently considering tearing down a wall to expand the space of their living room.

Technical Information Occasionally, Michael uses the stationary family computer. He primarily uses it for paying bills through home banking and also checks his email account through Microsoft Outlook. He only rarely surfs the Internet, but when he wants specific information, he knows how to make use of search engines. Once in a while, he also enjoys playing some of the card games that came pre-installed with the computer. Finally, he knows the basic usage of Word. However, it is primarily used when he needs to write letters, hence, he seldom uses it.

In relation to his technical skills at work, he has many years of experience with a variety of the workstations at the butchery. Specifically, he knows how to operate many of the industrial machines in terms of pushing the right buttons and configuring some of the machines according to the particular task they have to perform. When he is at a piecework task, he uses a laminated card with a barcode and his name on. The barcode is scanned every time he is assigned a piecework task both at the beginning and at the end. Additionally he needs to clock-in and -out every morning and evening, respectively. In this relation he has frequently made errors. For instance, some days the traffic can be chaotic because he lives in the city and this may result in that he forgets to clock-in because he needs to hurry in order to not be late at work. When he annoyingly realises that he forgot to clock-in, he needs to spend some of his lunch-break at the office where a secretary needs to correct his error manually.

When he is doing a piecework task, it also may happen that he forgets to clock-in correctly. In order to clock-in he needs to find the correct card with a barcode and name corresponding to the task he is assigned to do. He thinks this can be annoying because he first needs to locate this card and scan it and afterwards scan his own personal card. Besides this, it may sometimes occur that the particular task he performs is located relatively far from one of the clock-in machines. Hence, in some cases it may be that he needs to walk about 100 meters for reaching the closest clock-in machine.

He hopes that the new system will be much better than the old approach of doing clock-in manually. He hopes the introduction of the new system will remove some of his annoyances. However, he hopes that the adoption of the system will only require him minimal education because he does not want anything that may be more complex than the existing system.

F.2 Manager

Personal Information Bo is 52 years old, 170 cm tall with short blond hair. Straight after public school he went to get a butcher education and has therefore been working at the butchery most of his work life. He started by working as a normal butcher at the workstations, doing piecework. But an accident at work gave Bo certain back injuries. This ended his dream of being a butcher since it was impossible for him to handle the demanding job at the workstations for a whole work day. In the month after the accident Bo worked part time but later he advanced to manager.

Bo does not have a wife or any children, he only has a dog which he has named King. His hobby is hunting which he uses almost all his spare time on. When having shot a deer or the like he enjoys cutting it up, as it reminds him, of the old days at the butchery.

Technical Information At home Bo has no computer, but at work he is used to working with a computer for e-mailing internally at the butchery. The mail system is used through a web interface. Also since he has become a manager he has been attending several courses where e.g. he has been trained to use a computer to help him at his work.

Bo's job is to distribute the butchers at the different workstations according to which tasks need to done. He does this by looking up a task list on the Intranet, made by his boss, and then by remembering where the butchers normally work, and at which workstations they are certified to work, he distributes the tasks and writes them on a piece of paper. This paper is then put on a notice board in the butchery such that the butchers can check it when they come in for work.

F.3 Easy Clocking consultant

Personal Information Henrik is 33 years old. He is athletically built but is not overweight. He is 175 cm tall, has medium long hair and green eyes. 7 years ago he received a bachelor degree in computer science. He wanted a break between taking his bachelor degree and master degree but he has not yet pulled himself together to start studying again. Instead, since he left university, he has been focused on gaining experience in industry. He has been employed by various companies and has primarily worked with the programming part of the projects. Today he is employed as a consultant by the company offering the Easy Clocking system for automatic clocking at industrial companies. He has no children and is not married yet but has been seeing a girl, Joanna, frequently for the last 3 years. They are currently considering moving together in Henrik's apartment. Henrik lives about 20 kilometres away from the butchery and therefore uses his car every day.

In his spare time he enjoys tinkering with small gadgets. He enjoys multifunctional devices and one of his main passions is to write small programs for his telephone. He especially finds it amusing utilising some of the in-built sensors such as the accelerometer. Besides tinkering with gadgets, he is also very passionate about movies. Especially, he enjoys movies that fit into either the science-fiction genre or action. He owns a fascinating amount of movies some of which need to be stored in the basement simply because there is not enough room in his little apartment.

Technical Information Henrik has experience with computers that dates back to his childhood. Ever since he started using a computer, he has been working with it practically every day. Therefore, he is acquainted with many programs such as various word-processing tools such as Word but also with more basic text editors needed for formatting LATEXdocument such as Vi. Because of his deep passion with computers, he is also familiar with putting together the components that make up a computer and setting up SOHO networks.

One of Henrik's inabilities is however programming in what is considered low-level languages such as assembly and C. In fact, he is not fascinated with the low level details of programming. He is more focused on being given a set of high level functionality which abstracts the low level details. If it is possible to utilise tools or drag and drop-like code generation, he prefers that instead of writing it himself.

Henrik hopes that the new Easy Clocking system will be easy to customise and configure. According to configuring the individual location sensors, he hopes that this can be done graphically instead of using a terminal and execute a series of commands. For the plug-in part of the system he would be pleased to construct these and harness the location based data through high level and easy-to-use features.

G Role Models

When developing a new system it is a good idea to study existing systems. This makes it possible to draw inspiration or reuse modules from other systems and thereby possible reduce the effort required to create ones current system.[Design, 2000, c. 2]

We are not aware of other systems which have implemented an automatic clocking based on the location of the butchers. However, we know that there are systems which to some extent make use of some of the same ideas that will be central in the development of our location system. In the activity of finding potential role models, which may give inspiration to our location system, we have come across StreamSpin [AAU, 2009], LANDMARC [Lionel M. Ni and Patil, 2004] and The Active Badge Location System [Roy Want and Gibbons, 1992].

StreamSpin

StreamSpin is a location-aware system which main concept is *"...delivering and receiving mobile services. The services can be either explicitly subscribed to, or based on user context and meta-data."* [AAU, 2009] In some sense this can be related to our project where we want to do the clocking based on the context which the butchers are in.

The advantage of extending StreamSpin to our needs is that we have experience with its architecture from a previous project. However, there are also certain disadvantages in combining our project and StreamSpin. The main concept in StreamSpin is that the mobile devices calculate their location and give this to the server, which then pushes some context aware information to the clients. This concept differs from ours where it is the embedded units at the workstations that give the location of the mobile device to the server. Because of this difference in the main concepts, we believe the result of extending StreamSpin would require too many modifications with respect to developing our own system. Furthermore the architecture of StreamSpin is characterised by its scalability in case of many concurrent users, which in our case is not required to that degree. Our thoughts of StreamSpin have been confirmed by one of the developers at StreamSpin.

Even though we not are going to base our location system on StreamSpin we will still, to some extent, use it as a role model in the sense that it is a system which uses the location of its clients to take action.

LANDMARC

LANDMARC[Lionel M. Ni and Patil, 2004] is a location sensing prototype which utilises Radio Frequency Identification (RFID) for locating objects in indoor environments. The prototype environment consists of a sensing network whose responsibility is locating objects. These sensors are RFID readers and support communication over the IEEE 802.11 protocol. This enables wireless communication to a central server responsible for processing the incoming data and calculating locations of the objects carrying detectable tags. Using a wireless protocol as the fundamental framework of all communication in the infrastructure eases the deployment of the location system because cabling is not needed. Each movable object carries an RFID tag. Because LANDMARC targets indoor localisation, the developers try to accommodate problems with the tag being under conditions that may influence the accuracy of localisation. Specifically, various obstacles and materials may affect the accuracy of indoor localisation. To accommodate these issues without making use of extra RFID readers which are relatively expensive, LANDMARC uses the approach of setting up reference RFID tags at known locations from the readers. By determining how the signal strength between reader and the reference tag is influenced by the particular environment this knowledge can be transferred to the calculation of the movable objects carrying the tags thus increasing the accuracy significantly.

The LANDMARC approach gives inspiration to the system we are to develop. Initially, we could adopt the idea of letting the sensors wirelessly communicate with the central server for easing the deployment and possibly maintenance of the system. The idea of using RFID for location determination is also attractive because of feasible accuracy and relatively low-cost hardware. In the conclusion of the article about LANDMARC, they state that they are currently experimenting with a similar system in which they have substituted RFID with Bluetooth.

Active Badge Location System

The Active Badge Location System[Roy Want and Gibbons, 1992] is an early system for locating people and has been a role model for several other localisation systems. LANDMARC has drawn inspiration from The Active Badge Location System.

The Active Badge Location System makes use of automatically determining the location of individuals carrying a tag referred to as the Active Badge. Periodically the Active Badge emits a unique code every 15 seconds by using infrared signals. The signals are picked up by a network of sensors placed in the ceilings which, when receiving the signal forwards it to a central computer which displays information about where the individual is located. Because The Active Badge System makes use of infrared communication a requirement is line-of-sight between the badge and the sensors. Also, the signals only operate within approximately 6 meters of range.

The Active Badge System also gives inspiration for the system we are about to develop. Initially, the use of infrared technology for location determination may not suit our scenario because a butchery contains large industrial machines and some areas may be crowded with many obstacles thus requiring a significant amount of sensors to accommodate the requirement of line-of-sight.

The article about The Active Badge System focuses on privacy issues which turned out to be a major concern when the system was deployed at various institutions. Specifically, of major concern is how the location data about individuals is used and who is allowed to access it. Some sort of mechanism or policy must ensure that the information is not somehow misused. Also, there may be areas where it is inappropriate to locate people such as in the locker rooms. This is an interesting area which we have to take into account.[Roy Want and Gibbons, 1992]

H Configuration of a Location Sensor

When a new location sensor must be deployed at the butchery the following must be followed.

- The location sensor must be added to Easy Clocking using the web application. During this process a unique IP address must be specified.
- The firmware on the location sensor must be updated with the customised firmware for Easy Clocking. This can be done by putting the location sensor into diagnostic mode, hold reset while powering it on, and then issuing aftp as shown in Listing H.1. After 5 the location sensor can be restarted. The location sensor can then be reached through the LAN port on its default IP, 192.168.1.1, using telnet.
- To install the OLSRD daemon the command shown in Listing H.2 must be issued.
- Issuing passwd, on the location sensor, sets a defined password and enables SSH access.
- The network configuration file must then be similar to the configuration shown in Listing H.3. Where the <WIRELESSIP>, under Wi-Fi, is substituted with a unique IP address which is used by the OLSRD for the mesh network.
- The wireless configuration must then be changed to the one shown in Listing H.4. Note that <KEY> must be set to the appropriate key for the network.
- To configure OLSRD the line "option config_file '/etc/olsrd.conf"' in /etc/config/olsrd must be uncommented. Next /etc/olsrd.conf must be similar to H.5. Where <IP> is substituted with the IP defined for the location sensor in the web application.
- The firewall must be configured by placing script in Listing H.6 at /etc/firewall.user.[OpenWRT, 2009a]
- The localisation application can then be copied to the location sensor and set to initialise on reboot by inserting the list in Listing H.7 at /etc/init.d/easyclocking and adding the script to the list of upstart scripts.

```
atftp --trace --option "timeout 1" --option "mode octet" --put --local-file
    openwrt-EasyClocking.trx 192.168.1.1
```

Listing H.1: Command for updating the firmware of the location sensor.

```
opkg install olsrd
```

Listing H.2: Command for updating the firmware of the location sensor.

```
#### VLAN configuration
config switch eth0
       option vlan0   "0 1 2 3 5*"
       option vlan1   "4 5"

#### Loopback configuration
config interface loopback
       option ifname  "lo"
       option proto   static
       option ipaddr  127.0.0.1
       option netmask 255.0.0.0

#### LAN configuration
config interface lan
       option ifname  "eth0.0"
       option proto   static
       option ipaddr  192.168.1.1
       option netmask 255.255.255.0

#### WAN configuration
config interface     wan
       option ifname  "eth0.1"
       option proto   dhcp

config 'interface' 'wifi'
       option 'proto' 'static'
       option 'ifname' 'wl0'
       option 'ipaddr' <WIRELESSIP>
       option 'netmask' '255.255.255.0'
       option 'defaultroute' '0'
       option 'peerdns' '0'
```

Listing H.3: /etc/config/network

```
config wifi-device wl0
       option type    broadcom
       option channel 5
       # REMOVE THIS LINE TO ENABLE WIFI:
       option disabled 0
```

```
config wifi-iface
       option device  wl0
       option mode    adhoc
       option ssid    'EASYCLOCKING'
       option encryption wep
       option key     <KEY>
```

Listing H.4: /etc/config/wireless.

```
DebugLevel 0
IpVersion  4
ClearScreen   yes
Hna4
{
   #Virtual IP - To receive data from server
   <IP> 255.255.255.255
}

AllowNoInt yes
UseHysteresis yes

# Hysteresis parameters
HystScaling 0.50
HystThrHigh 0.80
HystThrLow 0.30

LinkQualityLevel  0
Pollrate   0.05
NicChgsPollInt 3.0

Interface "wl0"
{
   AutoDetectChanges        yes
}
```

Listing H.5: /etc/olsrd.conf.

```
#!/bin/sh
#Copyright (C) 2006 OpenWrt.org

iptables -F input_rule
iptables -F output_rule
iptables -F forwarding_rule
iptables -t nat -F prerouting_rule
iptables -t nat -F postrouting_rule

# The following chains are for traffic directed at the IP of the WAN interface
```

```
iptables -F input_wan
iptables -F forwarding_wan
iptables -t nat -F prerouting_wan

WIFI=wl0
LAN=eth0.0
WAN=eth0.1

iptables -A input_wan -p tcp --dport 22 -j ACCEPT

# Allow connections to olsr info port.
iptables -A input_wan -p tcp --dport 1979 -j ACCEPT

# OLSR needs port 698 to transmit state messages.
iptables -A input_rule -p udp --dport 698 -j ACCEPT
iptables -A forwarding_rule -i $WAN -o $WIFI -j ACCEPT
iptables -A forwarding_rule -i $WIFI -o $WAN -j ACCEPT

# For forwarding LAN & WIFI in nodes
iptables -A forwarding_rule -i $LAN -o $WIFI -j ACCEPT

# For WIFI clients to connect to nodes.
iptables -A forwarding_rule -i $WIFI -o $WIFI -j ACCEPT

# For connecting a wired lan client of node 1 to wired lan client of node 2
iptables -A forwarding_rule -i $LAN -o $LAN -j ACCEPT

# WIFI needs to go to LAN ports, too!
iptables -A forwarding_rule -i $WIFI -o $LAN -j ACCEPT
```

Listing H.6: /etc/firewall.user script.[OpenWRT, 2009a]

```
#!/bin/sh /etc/rc.common
                                                  START=90
STOP=15

start() {
cd /usr/sbin
   /usr/sbin/clocking_application &

 }

stop() {
       killall clocking_application
}
```

Listing H.7: /etc/init.d/easyclocking.

XML Schema and DTD

In Listing I.1, the XML Schema defining the requirements of an XML document representing a workstation is given.

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

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified
    ">
<xs:element name="workstation">
    <xs:complexType>
        <xs:all>
            <xs:element name="created-at" type="date"/>
            <xs:element name="id" type="intpos"/>
            <xs:element name="ip">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                       <xs:pattern value="((1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5]).)
                            {3}(1?[0-9]?[0-9]|2[0-4][0-9]|25[0-5])"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:element>
            <xs:element name="is-propagated">
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:boolean">
                            <xs:attribute name="type" type="xs:string" fixed="boolean
                                "/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="last-heartbeat" type="date"/>
            <xs:element name="movementthreshold" type="intpos"/>
            <xs:element name="timeawaythreshold" type="intpos"/>
            <xs:element name="detectingthreshold" type="intpos"/>
            <xs:element name="updated-at" type="date"/>
        </xs:all>
    </xs:complexType>
</xs:element>
<xs:complexType name="date">
    <xs:simpleContent>
        <xs:extension base="xs:dateTime">
            <xs:attribute name="type" type="xs:string" fixed="datetime"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="intpos">
    <xs:simpleContent>
        <xs:extension base="xs:integer">
            <xs:attribute name="type" type="xs:string" fixed="integer"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
</xs:schema>
```

Listing I.1: XML Schema describing the permitted contents of an XML document describing a workstation.

As shown, the XML Schema is self-describing and does not require much explanation. It defines all the elements and attributes that must be present in the XML document. Furthermore, it restricts the allowable types of the individual element values.

Listing I.2 shows the DTD dictating the contents an xml document describing a workstation must contain in order to be valid.

```
<!ELEMENT workstation (created-at, id, ip, is-propagated, last-heartbeat,
    movementthreshold, timeawaythreshold, detectingthreshold, updated-at)>
<!ELEMENT created-at (#PCDATA)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT ip (#PCDATA)>
<!ELEMENT is-propagated (#PCDATA)>
<!ELEMENT last-heartbeat (#PCDATA)>
<!ELEMENT movementthreshold (#PCDATA)>
<!ELEMENT timeawaythreshold (#PCDATA)>
<!ELEMENT detectingthreshold (#PCDATA)>
<!ELEMENT updated-at (#PCDATA)>

<!ATTLIST created-at type CDATA #FIXED "datetime">
<!ATTLIST id type CDATA #FIXED "integer">
<!ATTLIST is-propagated type CDATA #FIXED "boolean">
<!ATTLIST last-heartbeat type CDATA #FIXED "datetime">
<!ATTLIST movementthreshold type CDATA #FIXED "integer">
<!ATTLIST timeawaythreshold type CDATA #FIXED "integer">
<!ATTLIST detectingthreshold type CDATA #FIXED "integer">
<!ATTLIST updated-at type CDATA #FIXED "datetime">
```

Listing I.2: XML document of a location sensor configuration.

J Pitfalls in Using Pure Plug-in Architecture

The pure plug-in architecture has some pitfalls, as described by Birsan [2005]. We have added notes regarding these pitfalls and how they apply to the plug-in framework for Ruby on Rails.

Installing and Updating Because of the number of possible provided plug-ins and third-party plug-ins it can be problematic to test every possible software configuration and compatibility between the plug-ins.

A possible solution to this problem would be to use system and integration testing in order to verify the interoperability between the plug-ins.

Security The application and underlying system must be protected from malicious code in third-party plug-ins and malicious activity caused by bugs in the plug-ins.

One solution to this problem is to run each plug-in in a sandbox, and define precisely the APIs between the individual plug-ins. Unfortunately, there does not, at this time, exist a maintained library to sandbox Ruby code.

Concurrent Plug-in Version Support Plug-ins, as other software, are changed over time with the creation of new versions. This introduces some scenarios which are problematic, such as what to do if multiple versions of the same plug-in are added to the application, or if two plug-ins depend on different versions of the same plug-in. This can be problematic if the plug-in adds new elements to the user interface, resulting in multiple instances of the same button or menu.

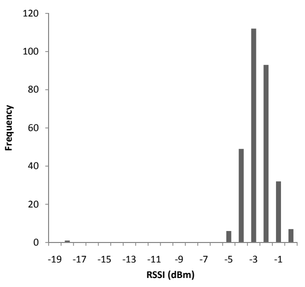
The Ruby on Rails plug-in architecture does not directly support the definition of versions and dependencies between plug-ins. As an alternative, one can use the gem packaging system in Ruby which supports versions and dependencies between packages, containing Ruby modules and even Ruby on Rails plug-ins. But the pitfalls mentioned above still remain. One possible solution, adopted by the Eclipse project, is to divide plug-ins into two groups, one adding new visual elements and functionality to the application and one providing libraries used by other plug-ins. Different versions of the library plug-ins may be loaded multiple times while the functionality plug-ins may only be loaded once, where they load the newest version.

Scalability, Up and Down If needed, the plug-in engine must be able to scale up to large numbers of plug-ins or scale down to small numbers whilst supporting quick start-up times and low memory usage. As an example some Eclipse installations contain thousands of plug-ins which depend on each other.

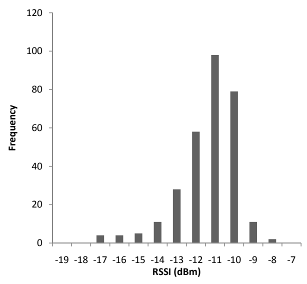
Each plug-in in Ruby on Rails is imported at initialisation, such that they function as if they were all provided as a normal application without plug-ins. Thus, by using a large amount of plug-ins, the initialization time would possible be extended, since the plug-ins needs to be imported, but then the application runs as if no plug-ins were used. Shifting the question regarding scalability to Ruby on Rails ability to run large web applications with many controllers, models, ect. Unfortunately, no documentation is available describing how Ruby on Rails reacts in this regard.

K

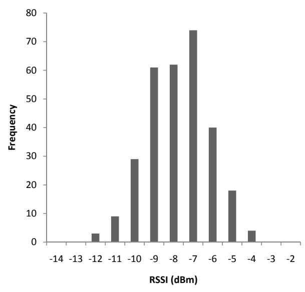
RSSI Frequency



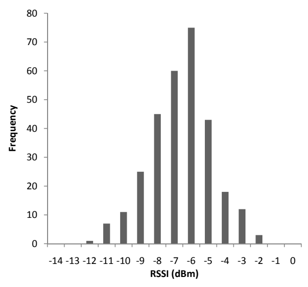
(a) 1 meter.



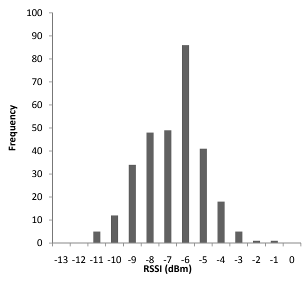
(b) 2 meters.



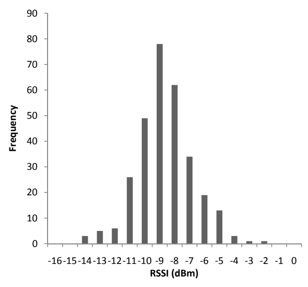
(c) 3 meters.



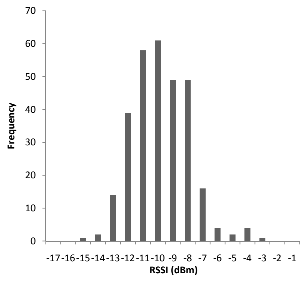
(d) 4 meters.



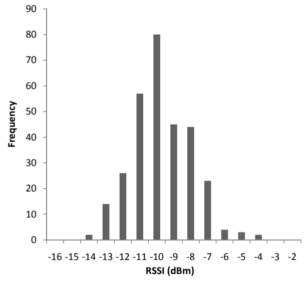
(e) 5 meters.



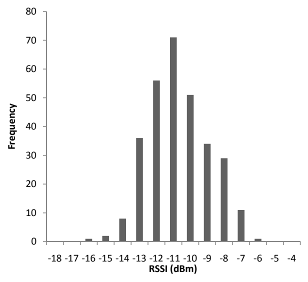
(f) 6 meters.



(g) 7 meters.

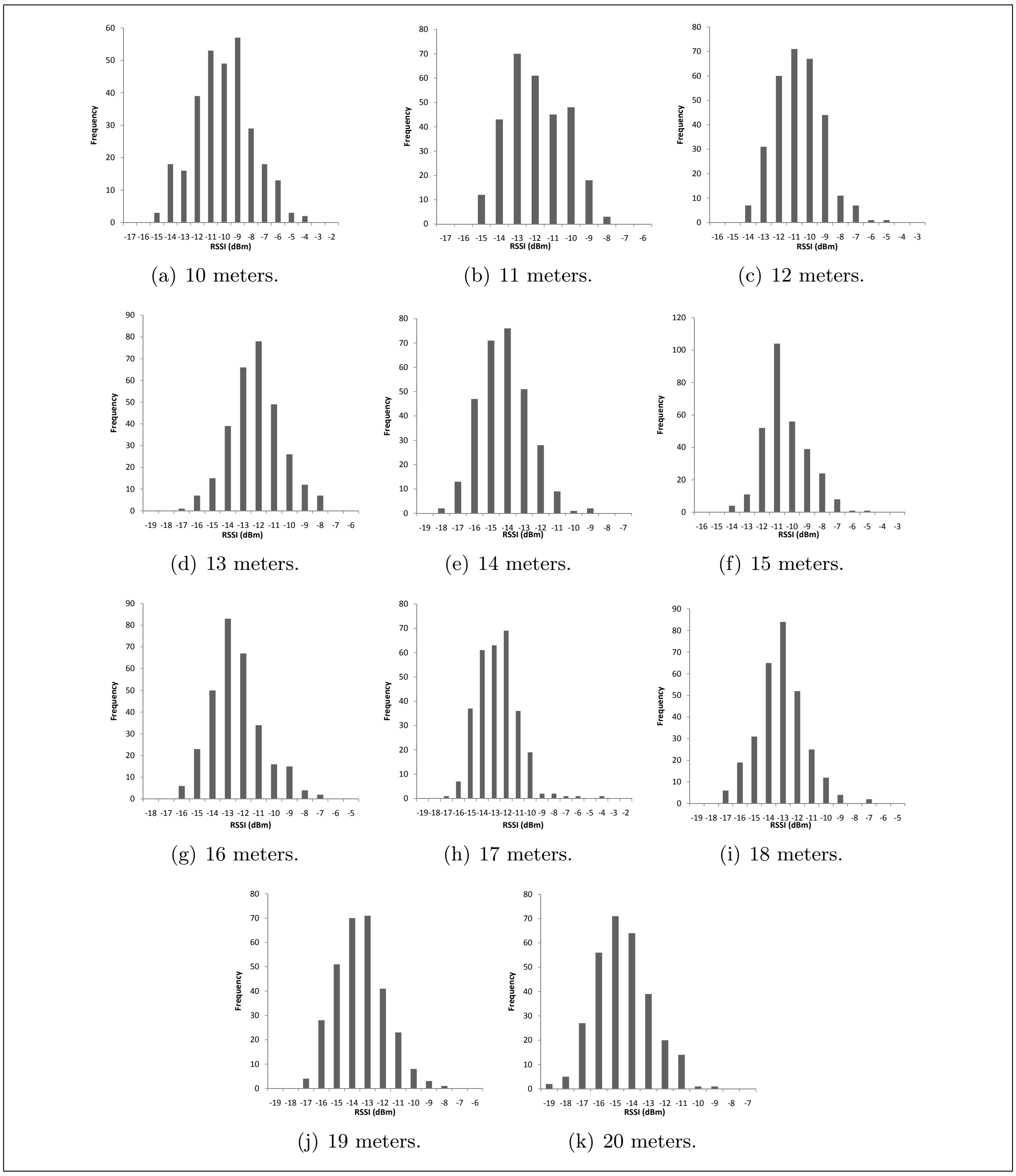


(h) 8 meters.



(i) 9 meters.

Figure K.1: Frequency of RSSI values for distances 1 to 9.



(a) 10 meters. (b) 11 meters. (c) 12 meters.

(d) 13 meters. (e) 14 meters. (f) 15 meters.

(g) 16 meters. (h) 17 meters. (i) 18 meters.

(j) 19 meters. (k) 20 meters.

Figure K.2: Frequency of RSSI values for distances 10 to 20.

L Usability Test Plan

The introduction read for the test users, in the usability test, is written below.[Krug, 2005, c. 9]

Usability Testing Introduction Welcome to this test session. During the tests you must remember that we are testing the system, not you. In order to improve the system, we want you to think aloud and honestly let us know what you think.

Easy Clocking is a system whose purpose is to automate the process of a clocking procedure in a butchery. It works by using sensors to detect butchers at specific workstations. A web application is used to assign assignments to butchers and solve conflicts in the gathered data from the localisation sensors.

The localisation of the butchers gives rise to conflicts in three cases. First, if the butcher is detected at multiple workstations at the same time. Second, if a butcher has not completed all his tasks at the specific day. Third, if a butcher has worked a task which he should not have.

Afterwards, we asked them to solve the following tasks one by one.

- Login to the Easy Clocking system, using “manager” as username and “1234” as password.
- Solve conflicts for the 23th of November.
- Determine which conflict is present for butcher1. Butcher1 is pre-configured with a time conflict. Try to solve the identified conflicts.
- The same test as the previous one is conducted on butcher2. Butcher2 has been pre-configured with a conflict regarding not having visited all workstations corresponding to his assigned tasks that particular day.
- Again, the same test as the previous one is conducted on butcher3. (Butcher3 has been pre-configured with a conflict regarding having visited more workstations than originally assigned.)

- The conflict page is visited and the test user must tell what functionality the page provides.
- The final test comprises a mixture of the three previous test cases to test how well a user handles multiple types of conflicts at the same time.

Bibliography

Aalborg University Denmark AAU. Streamspin, 2009. `http://streamspin.com/` (15. September 2009).

Arbejdstilsynet. Ensidigt, belastende arbejde og ensidigt gentagende arbejde., 2009. `http://www.at.dk/REGLER/At-vejledninger-mv/Arbejdets-udforelse/At-vejledninger-om-arbejdets-udforelse/D3-Ergonomi/RLOIA-D32-Ensidigt-belast-og-ensid-genta.aspx?sc_lang=da` (12. December 2009).

Atira. Scrum and agile methods the real world, 2009. `https://intranet.cs.aau.dk/fileadmin/user_upload/Education/Courses/2009/SOE/Slides/lecture14_atira.pdf` (15. October 2009).

Michael Barr and Anthony Massa. *Programming Embedded Systems: With C and GNU Development Tools*. O'Reilly Media, Inc., 2006. ISBN 0596009836.

Dorian Birsan. On plug-ins and extensible architectures. *Queue*, March 2005.

David A. Black. *Ruby for Rails*. Manning, 2006. ISBN 1932394699.

Andrew J. Blauch and Paul D. Johnson. Structured design using flowcharts. 2001.

BluetoothSIG. Bluetooth.com | the official bluetooth® technology info site, 2009. `http://www.bluetooth.com/` (22. September 2009).

A. Bose and Chuan Heng Foh. A practical path loss model for indoor wifi positioning enhancement. 2007.

Barry Burd. *Ruby on rails for dummies®*. John Wiley & Sons, Inc., 2007. ISBN 9780470081204.

chalermlab. Bluetooth not working, 2009. `https://forum.openwrt.org/viewtopic.php?id=19820` (19. October 2009).

Liwei Chan, Ji rung Chiang, Yi chao Chen, Chia nan Ke, Jane Hsu, and Hao hua Chu. Collaborative localization: Enhancing wifi-based position estimation with neighborhood links in clusters. 2006.

M. Ciurana, F. Barcelo-Arroyo, and F.Izquierdo. A ranging method with ieee 802.11 data frames for indoor localization. *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, March 2007.

Frank Cohen. Misunderstanding: Aren't xpath and xquery the same thing?, 2009. `http://www.datadirect.com/developer/data-integration/whitepapers/xquery-myths/xpath-vs-xquery/index.ssp` (13. November 2009).

Mike Cohn. *User Stories Applied: For Agile Software Development*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2004. ISBN 0321205685.

The European Commission. The european commission's decision on ultra-wideband technologies: Frequently asked questions, 2007. `http://europa.eu/rapid/pressReleasesAction.do?reference=MEMO/07/72&format=HTML&aged=0&language=EN&guiLanguage=fr` (24. September 2009).

Alan Cooper. *Inmates Are Running the Asylum.* Sams, 2004. ISBN 0-672-32614-0.

Mani Srivastava David Culler, Deborah Estrin. *Overview of Sensor Networks.* IEEE Computer, Special Issue in Sensor Networks, 2004.

Yukihiro Matsumoto David Flanagan. *The Ruby Programming Language.* O'Reilly, 2008. ISBN 978-0-59-651617-8.

DD-WRT. Wl-500g premium v2, 2009. `http://www.dd-wrt.com/wiki/index.php/WL500G_Premium_v2` (13. October 2009).

Object Oriented Analysis & Design. *Andreas Munk-Madsen and Lars Mathiassen and Peter Axel Nielsen and Jan Stage.* Marko, 2000. ISBN 87-7751-150-6.

Peter Dolog, Maristella Matera, Florian Daniel, and Sven Casteleyn. *Engineering Web Applications.* Springer, 2009. ISBN 978-3-540-92200-1.

Silke Feldmann, Kyandoghere Kyamakya, Ana Zapater, and Zighuo Lue. An indoor bluetooth-based positioning system: concept, implementation and experimental evaluation. *Institute of Communications Engineering, Hanover*, 2003. `http://projekte.l3s.uni-hannover.de/pub/bscw.cgi/d27118/An%20Indoor%20Bluetooth-based%20positioning%20system%3a%20concept,%20Implementation%20and%20experimental%20evaluation.pdf` (8. Oktober 2009).

R.E. Filman, T. Elrad, S. Clarke, and Prof.dr.ir. M. Aksit. *Aspect Oriented Software Development.* Addison-Wesley, Boston, 2004.

Gunter Fischer, Burkhart Dietrich, and Frank Winkler. Bluetooth indoor localization system. *Proceedings of the 1st Workshop on Positioning, Navigation and Communication (WPNC'04)*, 2004.

FOA. Registration and documentation in home care (danish), 2008. `http://www.foa.dk/Forbund/Presse/Undersoegelser/Kommune/2008/Dokumentation%20og%20registrering%20i%20hjemmeplejen.aspx` (4. December 2009).

Martin Fowler. *Patterns of Enterprise Application Architecture.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002. ISBN 0321127420.

FreeWRT. Documentation/targetsystems, 2009. `http://www.freewrt.org/trac/wiki/Documentation/TargetSystems` (13. October 2009).

Stuart A. Golden and Steve S. Bateman. Sensor measurements for wi-fi location with emphasis on time-of-arrival ranging. *IEEE Transactions on Mobile Computing*, 6(10), October 2007.

Mike Gunderloy. Getting started with rails, 2009. `http://guides.rubyonrails.org/getting_started.html` (13. November 2009).

Marko Helen, Juha Latvala, Hannu Ikonen, and Jarkko Niittylahti. Using calibration in rssi-base location tracking system, 2009. `http://www.cs.tut.fi/sgn/arg/heln/Publications/CSCC2001_Helen.pdf` (2. November 2009).

Rick Hesse. Normal probability plots. 1998.

Matti Hiltunen. Rfid reliability. January 2007.

Keith W. Ross James F. Kurose. *Networking: A Top-Down Approach 4th Edition*. Addison-Wesley, 2007. ISBN 9780321497703.

James Kalbach. *Designing Web Navigation*. O'Reilly, 2007. ISBN 0596528108.

Jesper Kjeldskov, Mikael B. Skov, and Jan Stage. Instant data analysis: Conducting usability evaluations in a day. `http://www.cs.aau.dk/~jesper/pdf/presentations/NordiCHI-04-Tampere-IDA.pdf` (30. November 2009).

Jesper Kjeldskov, Mikael B. Skov, and Jan Stage. Does time heal?: a longitudinal study of usability. pages 1–10, 2005.

Antti Kotanen, Marko Hännikäinen, Helena Leppäkoski, and Timo D. Hämäläinen. Experiments on local positioning with bluetooth. *Proceedings of the International Conference on Information Technology: Computers and Communications, p.297*, April 2003.

Steve Krug. *Don't Make Me Think: A Common Sense Approach to Web Usability, 2nd Edition*. New Riders Press, 2005. ISBN 0321344758.

Craig Larman. *Agile and Iterative Development: A Manager's Guide*. Addison Wesley, 2003. ISBN 0-13-111155-8.

Yiu Cho Lau Lionel M. Ni, Yunhao Liu and Abhishek Patil. Landmarc: Indoor location sensing using active rfid. *Online Document*, 2004. `www.cs.ust.hk/~liu/Landmarc.pdf`.

A. Malekpour, T. C. Ling, and W. C. Lim. Location determination using radio frequency rssi and deterministic algorithm. *Communication Networks and Services Research, Annual Conference on*, 0:488–495, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/CNSR.2008.32.

Ajay Malik. *RTLS For Dummies*. Wiley Publishing, Inc., 2009. ISBN 978-0-470-39868-5.

Guoqiang Mao, Brian D. O. Anderson, and Barış Fidan. Path loss exponent estimation for wireless sensor network localization. *Comput. Netw.*, 51(10):2467–2483, 2007. ISSN 1389-1286. doi: http://dx.doi.org/10.1016/j.comnet.2006.11.007.

mitEDB. mitedb. `http://mitedb.dk/shop` (1. October 2009).

Anders Møller and Michael Schwartzbach. *An Introducion to XML and Web Technologies.* Addison-Wesley, 2006. ISBN 9780321269669.

Netcraft. Web server survey archives - netcraft, 2009. `http://news.netcraft.com/archives/web_server_survey.html` (3. November 2009).

NetworkWorkingGroup. Optimized link state routing protocol - rfc 3626, 2009. `http://www.ietf.org/rfc/rfc3626.txt` (2. November 2009).

Jakob Nielsen. Why you only need to test with 5 users, 2000. `http://www.useit.com/alertbox/20000319.html` (30. November 2009).

Oleg. Bluetooth support in oleg's firmware?, 2009. `http://wl500g.info/archive/index.php/t-5148.html` (19. October 2009).

Open-Wrt. Openwrt buildroot - usage and documentation, 2006. `http://downloads.openwrt.org/docs/buildroot-documentation.html` (20. October 2009).

OpenPCD-Shop. Openpcd shop. `https://shop.openpcd.de` (1. October 2009).

OpenWRT. oldwiki:olsrmeshhowto - openwrt wiki, 2009a. `http://wiki.openwrt.org/oldwiki/olsrmeshhowto` (26. October 2009).

OpenWRT. Hardware-asus, 2009b. `http://oldwiki.openwrt.org/Hardware(2f)Asus.html` (13. October 2009).

Charalampos Papamanthou, Franco P. Preparata, and Roberto Tamassia. Algorithms for location estimation based on rssi sampling. pages 72–86, 2008. doi: http://dx.doi.org/10.1007/978-3-540-92862-1_7.

polarcloud. Tomato faq, 2009. `http://www.polarcloud.com/tomatofaq#what_will_this_run_on` (13. October 2009).

Jennifer Preece, Yvonne Rogers, and Helen Sharp. *Interaction Design, beyond human-computer interaction.* John Wiley & Sons, 2002. ISBN 0-471-49278-7.

Theodore S. Rappaport. *Wireless Communications: Principles and Practice (2nd Edition).* Prentice Hall PTR, 2 edition, January 2002. ISBN 0130422320.

Erik T. Ray. *Learning XML, 2nd Edition.* O'Reilly, 2003. ISBN 0-596-00420-6.

RFID-specialisten. Rfid. `http://www.rfid-specialisten.dk` (1. October 2009).

Bonnie Rind. The power of the persona, 2007. `http://www.pragmaticmarketing.com/publications/magazine/5/4/vol5iss4.pdf` (2. Oktober 2009).

Kay Romer and Friedemann Mattern. *The Design Space of Wireless Sensor Networks.* Institute for Pervasive Computing, ETH Zurich, 2004.

Veronica Falcao Roy Want, Andy Hopper and Jonathan Gibbons. The active badge location system. *Online Document*, 1992. `www.cs.ust.hk/~liu/Landmarc.pdf`.

Wiki RubyOnRails. Authentication and authorization in ruby on rails, 2009. `http://wiki.rubyonrails.org/howtos/authentication-authorization` (19. Oktober 2009).

Deborah Rumsey. *Statistics for Dummies*. Wiley Publishing, Inc., 2003. ISBN 978-0764554230.

Jeff Sharkey. Coding for life – battery life, that is, May 2009. `http://dl.google.com/io/2009/pres/W_0300_CodingforLife-BatteryLifeThatIs.pdf` (1. October 2009).

Mikael B. Skov. Understanding and conceptualizing interaction, 2009. `https://intranet.cs.aau.dk/fileadmin/user_upload/Education/Courses/2009/DIEB/Lektion02.pdf` (25. November 2009).

Ian Sommerville. *Software Engineering, Sixth Edition*. Addison-Wesley, 2001. ISBN 978-0201398151.

Tom Stafford and Matt Webb. *Mind Hacks: Tips & Tools for Using Your Brain*. O'Reilly, 2004. ISBN 0596007795.

John C. Stein. Indoor radio wlan performance, part ii: Range performance in a dense office environment. *Online Document*, 2000. `http://www.sparcotech.com/WLANs-in-offices.pdf`.

Jeffrey Stylos. Plug-in architectures, 2009. Lecture on Plug-in Architectures, `http://www.cs.cmu.edu/~bam/uicourse/830fall04/JeffStylosPlug-inArchitectures.ppt` (17. November 2009).

Shashank Tadakamadla. Indoor local positioning system for zigbee, based on rssi. 2006.

Kiran Thapa and Steven Case. An indoor positioning service for bluetooth ad hoc networks. *In MISC 2003: The 36th Annual Midwest Instruction and Computing Symposium, Duluth, MN, USA*, April 2003.

Jonathan Thatcher, Tom Coughlin, Jim Handy, and Neal Ekker. Nand flash solid state storage for the enterprise, April 2009. Whitepaper from SSI Group `http://www.snia.org/forums/sssi/knowledge/education/SSSI_NAND_Reliability_White_Paper.pdf` (30. November 2009).

Dave Thomas, David Hansson, Leon Breedt, Mike Clark, James Duncan Davidson, Justin Gehtland, and Andreas Schwarz. *Agile Web Development with Rails Second Edition*. Pragmatic Bookshelf, 2006. ISBN 0977616630.

Jr. Thomas A. Ryan and Brian L. Joiner. Normal probability plots and tests for normality. 1976.

TIOBE-Software. Tiobe programming community index for september 2009, 2009. `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html` (1. Oktober 2009).

Hans van Vliet. *Software Engineering: Principles and Practice.* John Wiley & Sons, June 2008. ISBN 978-0-470-03146-9.

Bill Venners. The philosophy of ruby, 2003. `http://www.artima.com/intv/ruby4.html` (28. September 2009).

W3C. Why validate?, 2009. `http://validator.w3.org/docs/why.html` (30. November 2009).

Eric W. Weisstein. Correlation coefficient. `http://mathworld.wolfram.com/CorrelationCoefficient.html` (10. December 2009).

Xirrus. High performance wi-fi networks, March 2007. `www.xirrus.com/library/pdf/Xirrus_High_Performance_Wi-Fi_Networks.pdf` (1. October 2009).