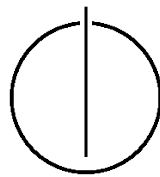# FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Framework for the Rapid Development of E-Textiles

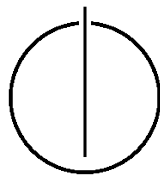Mertcan Yigin

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

A Framework for the Rapid Development of E-Textiles

Ein Framework für die schnelle Entwicklung von E-Textiles

Author:	Mertcan Yigin
Supervisor:	Professor Bernd Brügge, Ph.D.
Advisor:	Juan Haladjian, M.Sc.
Date:	October 15, 2015

I assure the single handed composition of this master's thesis only supported by declared resources.

Munich, Germany, 15.05.2015   Mertcan Yigin

Acknowledgments

I want to express my gratitude to my supervisor, Professor Bernd Brügge for making this project possible. I am especially grateful to my advisor, Juan Haladjian, for finding time and patience to discuss all the ideas that I came up with in the process of writing this thesis.

I also thank to my beloved friend Deniz Yeltekin for her support during my master study in Munich. She always encouraged me even when I lost my motivation and inspired me with her successes.

And above all, I would like to dedicate this thesis work to my parents Berrin and Hamdi Yigin. I want to thank my mother for her continual support and words of encouragement, and I want to thank my father for being my inspiration to pursue this field of work and education. I thank to dear my sister Merve Yigin for supporting me. They always believed in me even when I failed. They made a big effort for raising me as a fair and conscientious person.

Abstract

Electronic textiles (E-Textiles) are a good container for wearable technology and wearable computing specifically to produce activity recognition systems. They enable to use digital components and electronics to embed them into textiles. An activity recognition system built with e-textile can be used in a variety of ares like health monitoring, military and fashion.

Since electronic textile is a new concept, it suffers from technology related problems that have not been solved sufficiently; problems such as time consuming data collection, low-level development, unable to test different approaches in an efficient way and no good platform to experience. As a consequence, gathering, observing and testing information from e-textiles and wearable devices poses a challenge, and the development a product is slow and tedious.

In this thesis, a complete integrated development environment (IDE) to develop an e-textile product with gathering the data, observing and processing it and testing different algorithms has been proposed. The idea is to combine the playground feature for inexperienced developers and the IDE for e-textiles and wearable devices. User will not need any additional application to get the data from e-textile or wearable devices or any other applications (like WEKA or Matlab) to test and process the signal.

The focus of this thesis is on the design and the development of a framework for e-textile development. The framework is a combination of a playground and a conventional IDE. Text based development framework called TextIt that is used for gathering, observing, testing for the e-textile application. The system is developed with playground support to make it easy to enter this area as an inexperienced developer. As the system is able to show the data, have pre-defined algorithms for signal processing and visualize the different test algorithms and compare the results.

Contents

Outline of the Thesis

CHAPTER 1: INTRODUCTION

The introduction chapter starts with the definition of an e-textile, and a motivation on how it can improve human lives. This chapter describes the challenges that are faced when creating an e-textile product, and proposes a solution for overcoming these challenges.

CHAPTER 2: REQUIREMENTS ELICITATION

This chapter describes the requirements elicitation which includes the definition of scenarios that drive the development of the TextIT. Based on these scenarios, non-functional requirements are proposed and a functional model with use cases and functional requirements is elaborated.

CHAPTER 3: ANALYSIS

This chapter provides a description of an analysis model based on the requirements. The model is created to formalize the objects and information that exist in the domain of the e-textile environment. It identifies the entity, boundary and control objects of the TextIt. The created object and sequence diagrams provide a better understanding of the system requirements. As a part of the analysis object model, abstract factory were used for modeling a e-textile environment. By using aa abstract factory pattern tackles a problem of modeling an e-textile communication with multiple messaging types that may execute different set of commands, while each device can use a different communication protocol.

CHAPTER 4: SYSTEM DESIGN

This chapter gives a description of the system design model which contains the strategies and practices for designing the TextIt. As a part of the design model, the TextIt is decomposed into smaller subsystems, and the hardware/software mapping is shown. The broker pattern is used for decoupling clients from communication tasks. The broker component is decomposed into sub-components, and the role of each sub-component is shortly described. Finally, persistent data management, software control and boundary conditions are discussed.

CHAPTER 5: OBJECT DESIGN

In this chapter, a description of the components of the broker for controlling the e-textile is given. The interfaces of the components are described in detail.

CHAPTER 6: CONCLUSION

This chapter gives a summary of what was done during the course of the thesis. Afterwards, the results are reviewed critically and visions for future work are pointed out.

1. Introduction

E-textiles, also known as smart garments, smart clothing, electronic textiles, smart textiles, or smart fabrics, are fabrics that contain embedded digital components (including small computers) and electronics [1es]. Simply put, e-textiles are fabrics with electrical characteristics. These electronic devices can be worn on clothes, stored in pockets, held in the hand, or even strapped to the head. E-textiles provide an effective system to store or connect computing elements [Cho08].

Sensors and low-power processors are now small enough and inexpensive enough to allow the deployment of wireless ad hoc networks for various applications [CS03]. Recent technological developments have enabled miniaturization; smaller, lower-priced, and lighter chips; wireless sensor networks (WSN); and fiber level research. Thus, e-textiles have emerged, and the research focus has shifted from wearable devices to e-textiles. As in classical electronics, the construction of electronic capabilities in textile fibers requires the use of conducting and semiconducting materials such as conductive textiles [1es]. One of the most important issues of e-textiles is that the fibers should be made in a way that they can be insulated and completely washable, so they can be successfully used in clothing manufacture. Fortunately, as a result of fiber level research, a new class of electronic materials has emerged that helps to create e-textile products that acts like a normal clothes. That is why the new class of electronic materials is more suitable for e-textiles and solves daily use problems. It is a class of organic electronics materials that can be conducting and semiconducting, and they can be designed as inks and plastics. As a result of fiber level development, it is possible to produce woven circuits [CKTT10]. These woven circuits enable the integration of electronic components in woven e-textiles at the yarn level. Furthermore, in connection to the continuing development of wireless sensor networks, the authors in [HSB+13] state that the excitement about this technology is motivated by the several benefits associated with the long-term monitoring, low cost, rapid deployment, self-organization, and flexibility features of WSN. These different technological developments make it possible to create e-textile products with electronic devices that can be worn, bent, washed, and carried without attaching any additional devices. For example, rather than attaching electronic devices to the body with strips, the strips are now the electronic components. Thus, because of all these technological advancements, electronic components can now be integrated into clothing.

The field of e-textiles can be divided into two main categories:

- E-textiles with classical electronic devices such as conductors, integrated circuits, LEDs and conventional batteries embedded into garments.

- E-textiles with electronics integrated directly into the textile substrates.

Since textiles play an essential role in human existence, e-textiles possess the potential to be used in many aspects of our lives. Wearable textile products can be utilized in many

different areas, including sports, fashion, medicine, rehabilitation, context awareness systems, and social media integration [Cho08]. Representative applications are sports training data acquisition, health monitoring of vital signs, monitoring personnel handling of hazardous materials, tracking the position and the status of soldiers in the field, and monitoring pilot or truck driver fatigue. Furthermore, in the area of rehabilitation as stated in [HSB+13], rehabilitation requires continuous supervision during long-term rehabilitation therapy. This continuous supervision increases the workload for physical therapists and medical staff, and it is very expensive for the patients. As a result of such challenges, new sensor-based e-textile solutions have arisen from the need to develop effective, low-cost and easy to use rehabilitation supervision systems. Using e-textile solutions dramatically reduces the cost and size of systems and opens new opportunities [HSB+13].

Since the field of e-textiles has subcategories, the word e-textile is used to identify a variety of concepts. These concepts include fiber level development, textile sensors, and application development for smart garments (health/rehabilitation, fitness, fashion, maintenance). This thesis focuses on the application level development for e-textiles. In this paper, I present the design and implementation of a framework for rapid development of e-textile applications.

1.1. Problem Statement

Wearable devices and the smart garments has created a significant research interest over the last decades [HSBH14]. This research interest has occurred by the commercial interest. As a result it can be said that smart garments are highly popular among not only researchers but also end users. Despite this interest by both the developers and the end users, only a few developments have exceeded a prototypical level [HSBH14]. Moreover, problems still exist that prevent an efficient level of mass production.

The basic materials used to produce e-textiles, which are conductive threads and fabrics, have been around for over 1,000 years. However, the e-textile market is still in its infancy, and the quantity of e-textiles being produced is limited. One main reason for and resulting criticism of this limited development and production of e-textiles is the slow and complicated process currently involved.

The process of development an e-textile application is actually slower and complicated when it is compared to alternative mobile or web application development. One of the reasons for the this slow and complicated process is that there are few good high level sources of information or research data to give a practical insight into making e-textile development easier and quicker. The development must go through the following steps: signal receiving, synchronizing, signal filtering, and feature extraction. These essential steps generally need to be done in order to develop an e-textile application. However, when new developers are interested in working in this area, it is hard for them to find solid resources to aid in the development of their applications. Thus, because new developers may not have a comprehensive grasp of the essential steps in e-textile development, they often do not know where to start.

Another reason is that the knowledge that is required to develop e-textiles is complex. The development of e-textiles includes sensor interfaces, synchronization of the sensor signals, and signal filtering [LAB08]. For beginners in the field, it is first required to experiment with the signals, to then review some examples, and finally to use an interactive development tool in order to get some idea about the concept. The IDEs and libraries that are used for e-textile development will be explained in detail in the Related Work section. They focus on either students that are not professional developers or experienced developers that have previous experience with e-textiles. These IDEs and libraries do not actually support complex application development since they lack testing, simulation, visualization, and predefined algorithms for machine learning or filtering for a real e-textile devices. The problem here is the lack of supporting entire process in development of an e-textile application. Part of the steps can be achieved by using the mentioned IDEs and libraries partially if the other steps are achieved manually. In order to realize all the steps in one environment, all of these procedures should be implemented manually. So that this creates an expectation that users have a deep knowledge about e-textile development, and so they do not really support newcomers to the field.

There is a jungle of libraries and protocols and these libraries and protocols are either vendor specific or operate on the hardware level; for example, the developers must be able to handle the hardware I/0. In this situation, the developers need to start from the beginning by checking hardware level development. This low-level development is another reason for why overall e-textile development is such a slow process. Since there is no good abstractions for the development, users need to work on the hardware elements and develop them manually. This situation creates a time issue for the developers and they felt rushed the development and the problems. Since this low-level of development requires specific information about the device, most developers lack this kind of information, and it is hard to obtain this knowledge quickly and efficiently. Hence, that is the reason development is separated from device in other areas like communication, such as with servers and mobile devices. This device application separation is not available in e-textile science at this point in time.

Furthermore most developers are not experts in signal processing or activity recognition but want to realize their application concepts using a high level API [HSBH14]. Niels Henze et al argue that there is a lack of tool support for application developers and no enough defined APIs within the software and hardware stack that allows developing useful smart garment applications [HSBH14]. The hardware that is used to produce e-textile products, like raspberry-pi, Arduino or Intel, does no support for higher level development, there is not enough protocols and libraries to interact with hardware components. The problem is there is no application layer in the e-textile development yet. (Fig. 1.1). As a result, all the communications with hardware, such as reading the data from sensors or writing to actuators, are needed to be done by the developers. This situation makes building more complex systems even harder. Like in development of mobile or desktop application, in the development of e-textile also requires for APIs that support more complex commonly used functions or functionality such as data mining and machine learning. Unfortunately in the current situation there is no tool support for application developers

and no defined APIs within the software [HSBH14]. The development is on specific devices that the libraries or protocols are designed for. This makes an application is specific to that devices or vendors. All the signal-processing algorithms, signal synchronization or filtering methods are developed specific to the used device because all the input output handling are specifically designed according to the used hardware. Thus the knowledge about developing on hardware is required from the developers. Since expecting a good knowledge on hardware development is not realistic, there are some researches are still going on about creating a top development level [HSBH14], [CKTT10], [LAH$^+$14], [LAB08], [HSB$^+$13].

Most of the development environments, which are used to test different system variants, are not suitable for actually running applications [LAB08]. These environments are dependent upon custom engines or libraries. For this reason, in order to test different systems, implementing an actual e-textile application for specific device is necessary. In order to do this, developing selected algorithms in an appropriate programming language and then distributing them to specific devices are necessary. Relevant issues in this process include sensor interfaces, synchronization of the signals, and optimizing for specific devices. Although the applications are similar, when the developers want to test them on other devices, since all the implementations are done for a specific device, the same development has to be done from the beginning for any different hardware.

E-textile development is similar to distributed systems. It is hard to communicate and synchronize between different sensors. Moreover, as stated before, it is hard to simulate the application before actual implementation; therefore, it is not easy to test a system. Developers may not know which algorithms to use in their application. For example, just for filtering alone, there are more than twenty-four algorithms that can be used for smoothing the signals [10ng]. In order to find the best one, it requires several phases of testing. Also, the algorithms change according to the data and the sensor. Since there is no easy way to simulate the data and test them with different algorithms to find the best ones, the developers need to implement every algorithm that they want to compare, in order to find the suitable one. The developers need to collect the data again, apply the algorithm, see the results, and compare them. This cycle needs to be done for different algorithms to compare the results and find the appropriate ones. They cannot change the algorithm in the apply algorithm phase. If they want to test a different algorithm, they need to start the process from the beginning. This is a time-consuming process when the system is more complex.

1.2. Solution

According to problems that are described in section 1.1, an interactive e-textile development environment, so called TextIt, is proposed as a solution in order to increase the speed of development of e-textile application. As a result more developers may join to production of e-textile application without any previous knowledge requirement about e-textiles.

The solution of development environment that is proposed is to create some kind of

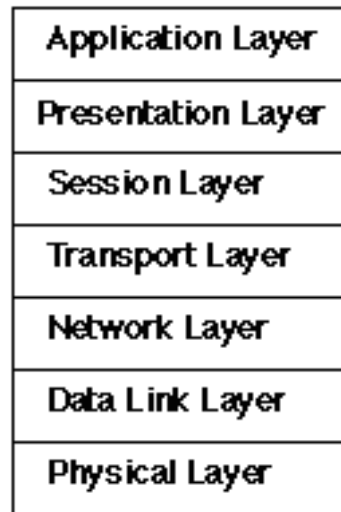| Application Layer |
| Presentation Layer |
| Session Layer |
| Transport Layer |
| Network Layer |
| Data Link Layer |
| Physical Layer |

Figure 1.1.: Presented Development Layer

an e-textile playground for the developers. In that development environment with the playground feature developers can see how a data looks like, can use the pre-defined components and see the affects, let the system handle device communication and input output handling. Developers may get some sample data or receive the real data from e-textile products and have predefined filtering, feature extracting and machine learning components, which are required to process data, so that they can try their own application ideas by using these components. These filtering, feature extracting and machine learning components have to be extracted according to current market needs. In the e-textile applications there are some common steps that are used several times like Bluetooth communication, signal processing, filtering, machine learning and etc. Since these steps are common and there are not some standardized solutions for the development, the environment should support pre-defined solutions for those common steps. Implementing those components for every application is a wasted effort and money especially they do not change over time. That is why the proposed solution comes with a pre-defined algorithms that are used in the development of an e-textile application. Moreover since the developers are required to test and experiment with different algorithms before the actual implementation, the development environment can solve this problem by providing the algorithms that the developers need and can show the result of different algorithms. By this approach the developers can see the effects of the algorithms and choose the best one for their applications and use it for actual implementation. This support for pre-defined components helps developers to explore more and be more free on choosing algorithm. With this approach rather worrying about how to implement the algorithms, the developers can spend more time on designing phase and be more imaginative and think about more complex applications. Focusing on the actual application and thinking about what can be done rather than how can be done can help e-textile to be more innovative and can address the problems of the market.

As a result, to realize the solution of a development environment for e-textiles, needed

APIs, pre-defined algorithms and components are required to be identified. The purpose to collect those requirements under one environment is to support developers to easily access the resources specifically for e-textile development. The aims of this e-textile environment, so called TextIt in this thesis, are to speed up the development of e-textiles, make it available for many developers that have different knowledge background and make the applications hardware independent to extend its usage area.

The presented system mockup can be seen in figure 1.2.

Figure 1.2.: Presented System Mockup

2. Requirements Elicitation

The requirements elicitation chapter provides a description of the overall use of the system and defines the requirements that the system has to meet. The requirements are extracted from the project KneeHapp. In order to identify the requirements the project KneeHapp is developed with the current technologies that are available to develop smart garments. After using the available e-textile development environments and frameworks, the missing pieces are identified. KneeHappproject is developed in TUM iOS praktikum with Prof. Bern Brüge and sport traumatology expert Dr. Jürgen Höher in 2014 [BHF+15]. Moreover the participants of this project were computer science students and had not previous knowledge about e-textile development. This made us also see the barriers and challenges for beginners to e-textile development. With KneeHappproject we had the chance to work on a real e-textile project and see the current status and the drawbacks of the area. The most of the requirements are extracted by our observations. The requirements are extracted from the scenarios described below. Extracted requirements are divided into two categories: **functional requirements** and **non-functional requirements**. Functional requirements describe the interaction between the system and its environment, independent of its implementation [BD03]. Non-functional requirements refer to user requirements that are not directly related to the functional behavior of the system [BD03].

2.1. Project KneeHapp

The rupture of the anterior cruciate ligament (ACL) of the knee is a severe injury. ACL ruptures are mainly the result of sports related injuries such as football, basketball, handball, and contact sports. An intense rehabilitation period is necessary to bring the individual back to physical activities and the desired sports. The rehabilitation of the knee often includes physical therapy and strength exercises. The progress and the success of the rehabilitation measures are currently not well documented. The goal of this project is to develop a system that would allow patients to record their rehabilitation progress without the need of a doctor or physiotherapist. During the Summer Semester 2014 we, six TUM students, developed a smart bandage (Kangaroo Bandage) and iPad application (Knee-Happ) that measure performance parameters while the patient performs the rehabilitation exercises. The smart bandage was consisted of two Arduino devices and the devices were put in to the knee bandage. One of the exercises that were supported was "One Leg Hop". The one Leg Hop is an exercise in which the patient jumps forward as far as he/she can with only one leg. Then the algorithm calculates the duration of the hop. The other exercise was "Side Hops". In this exercise patient should jump right and left as many times as he/she can in the given time. Then the algorithm calculates the number of the hops from accelerometer data.

The requirements are extracted by use cases and they also serve to describe the function-

ality of the system from the users point of view [BD03].

2.1.1. One Leg Hop

One Leg Hop is the exercise that the patient should jump as far as he/she can, then the smart garment should calculate the duration of the hop by processing the acceloremeter data. Figure 2.1 shows the linear accelerometer data from the y and z-axis corresponding to a one-leg hop. Even for human eyes it is not easy to read this accelerometer signal. There are one maximum peak and two minimum peaks in the signal. Thus without further observations, one can not conclude which points corresponds to jumping and landing. So that we could take the time difference between these two points. It was not really possible to implement the algorithm before really understanding the signal and knowing which points to take. We made several iterations of jumping, collecting the data, video recording of the jump, using different filtering methods, using different feature extraction methods. Unfortunately at the end we found measuring the duration of a hop from accelerometer signal challenging due to the fact that hops occur very quickly (an average hop lasts around 0.22 seconds) and there are so many noise because of the leg shaking. One possible solution we proposed was using an insole with integrated pressure sensors. We thought that it could be more easier to detect when the user leaves the ground and lands the ground by using insole with pressure sensor. Figure 2.2 shows the insole and its hardware. Figure 2.3 shows the signal produced by the pressure sensors during a one-leg hop. We could then determine whether the user is on the ground or in the air using a threshold.
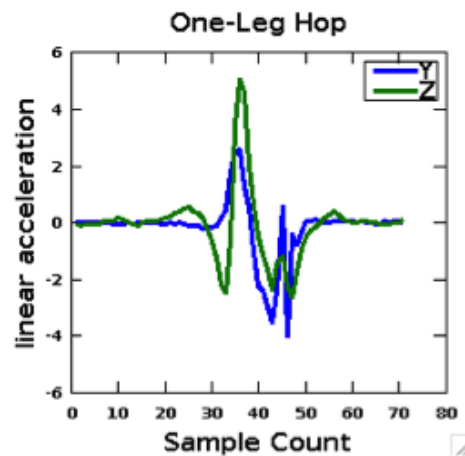

Figure 2.1.: One Leg Hop with Accelerometer Data

2.1.2. Side Hops

Side Hops is the exercise that the patient should jump right and left as many times as he/she can in the given time. Then our smart garment should calculate the number of jumps. This was rather an easier exercise than the One Leg Hop to calculate. To given an idea about how the signal looks like, the example linear acceleration on the horizontal axis for 7 side hops is given in Figure 2.4. This was rather cleaner and easier to understand
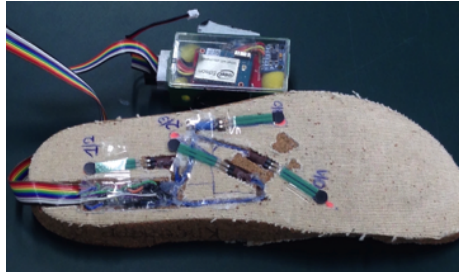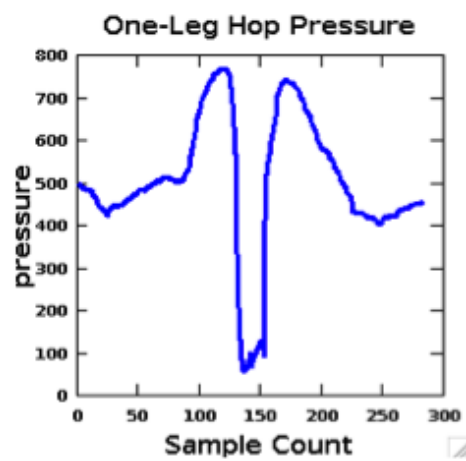
Figure 2.2.: Insole - Pressure Sensor

Figure 2.3.: One Leg Hop with Pressure Sensor

when compared to One Leg Hop exercise signal. The number of jumps are clear for human eyes, as the peaks indicates the jumps. However for the algorithm to calculate the number of the jumps, we needed to filter the signal from the noise. In order to eliminate noise in the signal, we applied several iterations of a low-pass filter until the signal's standard deviation became smaller than a specific threshold. We called this threshold low-pass filter threshold. We found the optimal low-pass filter threshold to vary depending on the algorithms used for calculating the jump duration. We tested the RC low-pass filter with a time constant T = 0.25 and a weighted moving average filter with coefficients [1/4 1/2 1/4]. After filtering the signal, we used a peak detection algorithm to count the high and low peaks in the signal. A naive peak detection algorithm counted also local maxima as hops in the signal, resulting in more side hops being counted. To overcome this issue we analyzed the surroundings of a peak. If the peak is larger enough than its surroundings, the peak is counted as a hop. We called this factor peak threshold. We found the peak threshold leading to most accurate results to depend on the filtering algorithm used. Figure 2.5 shows the same 7 hops after applying three iterations of the RC filter and running the peak detection algorithm.
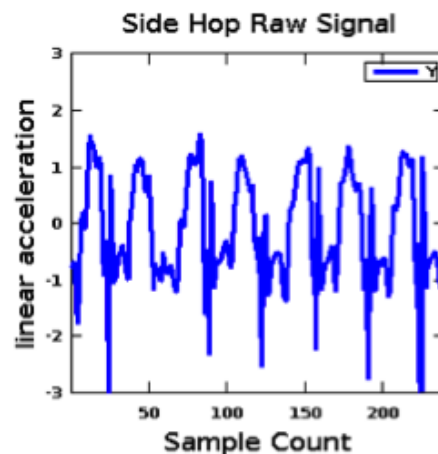

Figure 2.4.: Side Hop Raw Data

2.2. Requirements

In this section requirements are presented. Requirements are collected into 3 main parts; gathering, observing and testing the data. Paul Lukowicz et al. argues the requirements for e-textile environments can be listed as, a collection of ready-to-use algorithms (signal processing, pattern classification, and so on), synchronization, merging, and splitting of data streams. The environment should include I/O device readers and writers, filtering and classification algorithms, and components for splitting, merging, and synchronizing data streams. Also a visualizer is a must to show the data to user. Thus it would be easier to work on the data.
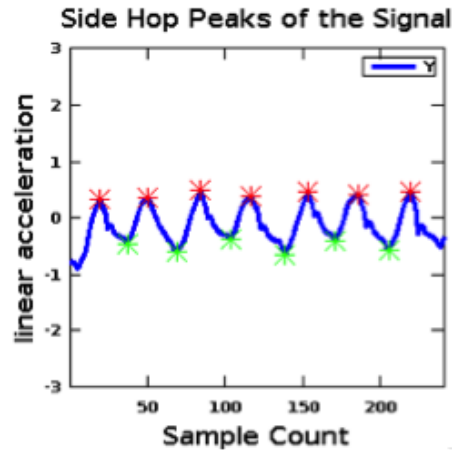
Figure 2.5.: Side Hop Filtered Data

Observation 1: Gathering

In our case since we did not know how the signal looks like, first of all we needed to collect sufficient amount of sample data to really understand the signal and discover what peaks, curves, numbers and points actually mean in the context. Collecting sufficient amount of data is the important key before the actual implementation because first we need to understand the data and the problem before proposing the solution to that. Understanding the signal by gathering data is required since the developers did not have previous knowledge about e-textile field and know how the actual data looks like. In our project for this purpose, we developed a mini application to record all the acceleration and gyroscope data. After all the collection process was done, we needed to transfer to data into an environment in which we process the signal. Although it might sound as an easy process it took several full days to only collect 10 jumps from users.

Observation 2: Observing

After the collection of the data, the signal was passed to another environment in which it was processed. As mentioned before, since the developers had not had any previous knowledge about the data, it was necessary to show how actually the signal looks to give an overview about the problem. Then we could understand the signal better and decide what was needed to process the signal (filtering, feature extraction, data mining, machine learning). In our case we used different environments and frameworks ,like WEKA, Matlab, Octave as desktop applications and JBChartView, iOS Chart etc. for iOS integration, to visualize the exercise signal. To try every different framework we had implemented them in our project to see which ones gave the better solution for our application. Because every framework is suitable for different purposes, it took quite a time to find the best one for our needs. It was necessary to show not only the original signal to developer but also how it looked like after every manipulation the developer made to process the data and make it more understandable. With this approach we could see what effects every algorithm had. Also we had the same approach when deciding the filtering and feature extraction

methods to use. In our project we implemented different filtering and feature extraction methods and applied them to data sequentially and noted the results of the methods.

Observation 3: Testing

As every algorithm has different effects on the signal, we wanted to see which one had the best results for our project. In our case we manually took the result of different methods and compared them with our references. Then we created a table to see which ones we should use. To find the appropriate methods, it required several iterations of gathering, observing and testing. Since only one iteration takes significant amount of time, repeating this loop was a time consuming process. It would be easier if we could have defined the different test cases to run and the references to compare with to the development environment, then the result returned as an excel like table to present the final results of different test cases and the comparison of the accuracies.

2.2.1. Functional Requirements

The following subsection defines functional requirements identified from the previously shown use cases.

FR1: Data Gathering

The system should receive data from an e-textile.

FR2: Data Observing with Graph

The system should be able to visualize the incoming data from an e-textile as a graph.

FR3: Data Observing with Number

The system should be able to visualize the incoming data from an e-textile as a number.

FR4: Data Observing with Text

The system should be able to visualize the incoming data from an e-textile as a text.

FR5: Data Observing with Video

The system should be able to visualize the incoming data from an e-textile as a video.

FR6: Creating of different Algorithm Test Cases

The system should be able to create different algorithm test cases to apply on the same data.

FR7: Applying of different Algorithm Test Cases

The system should be able to apply different algorithm test cases on the same data.

FR8: Testing of Different Algorithms

The system should show the results of different algorithms applied on data and the comparison of algorithms accuracy.

FR9: Data Manipulation

The system should allow the user to manipulate and apply different algorithms to the data.

FR10: Test Case Management

The system should allow the user to create and to run different test cases.

FR11: Filter

The system should support to filter the data.

FR12: Feature Extract

The system should support to feature extract from the data.

FR13: Learn

The system should support to learn from the data.

FR14: Live Coding

The system should interpret the user input live with REPL (read eval print loop) approach.

2.2.2. Non-functional Requirements

In the following subsection, the non-functional requirements of TextIt will be described.

NF1: Vendor Independent

The system should provide support to communicate with different e-textile products.

NF2: Allowed Vendor List

The system should communicate with allowed e-textile products.

NF3: Extendibility with Respect to Algorithms

The system should support the addition of new algorithms.

NF4: Response Time

The system should provide a feedback to user inputs, within 5 seconds (an error message in case of an error or a message about a successful execution of a command).

NF5: Pre-defined Re-usable Library

The user should be able to use pre-defined library that are defined in Functional Requirements.

NF6: Robustness

Invalid developer input should not crash TextIt.

NF7: Operating System

The system should run on MAC OS.

NF8: Interactex Integrity

The system should be able to communicate with Interactex (Graphical e-textile development app for iPads) and send the code that is defined by a developer.

NF9: Interpreter

The system should support an interpreted language to allow developers to code.

2.3. Related Work

The following sections provide the summary of proposed related framework to work on e-textile development.

2.3.1. E-textile Environments

In this section the generic environments, that are created to create an abstraction for e-textile systems, are explained in detail.

The CRN Toolbox enables fast implementation of activity and context recognition systems, featuring mechanisms for distributed processing and support for mobile and wearable devices. It is a tool set specifically optimized for implementing multimodal, distributed activity and context recognition systems running on Posix operating systems. The CRN Toolbox contains a collection of ready-to-use algorithms (signal processing, pattern classification, and so on). Another important feature is the ability to interface conventional simulation environments such as WEKA (Waikato Environment for Knowledge Analysis, www.cs.waikato. ac.nz/ ml). The concepts the CRN Toolbox uses are graphical programming, data driven computation, parameterizable libraries, and distribution [LAB08].

TinyOS is an open-source operating system designed for low-power wireless devices, such a sensor networks, ubiquitous computing, personal area networks, smart buildings and smart meters. TinyOS provides useful software abstractions of the underlying device hardware. TinyOS is especially useful for microcontroller-based devices that have sensors

and/or networking capabilities. It is developed to make a layer on top of the hardware so that the Os separately runs the applications from hardware [LMP$^+$05].

RUNES (Reconfigurable Ubiquitous Networked Embedded Systems), is a project to enable the creation of large scale, widely distributed, heterogeneous networked embedded e-textile systems that interoperate and adapt to their environments. RUNES aims to provide an adaptive middleware platform and application development tools that allow programmers the flexibility to interact with the environment where necessary. It also offers a level of abstraction that facilitates ease of application construction and use. This will allow for a dramatic cut in the cost of new application development and a much faster time to market [PGM$^+$05].

VisualRDK is a high-level programming language for prototyping pervasive applications. Pervasive applications are inherently distributed. Logical control flow jumps between sensors, context servers, PDAs, and any other kind of networking appliance. The control flow is extremely difficult to follow using classical debugging tools because a debugger is attached to a single process running on a single device. the Visual Robotic Development Kit (VisualRDK) that can generate two implementations from one source: a prototype version and a debug version [WKU$^+$07].

BASE is a flexible middleware for Pervasive Computing environments. It provides adaptation support on the communication level by dynamically (re-) selecting communication protocol stacks, even for currently running interactions. It is created for an extensible middleware platform covering small embedded systems to full-fledged desktop computers [BSGR03].

PCOM: a component system for Pervasive Computing. PCOM offers application programmers a high-level programming abstraction which captures the dependencies between components using contracts. BASE offers no support for adaptation at higher levels, e.g. by automatically reselecting services and devices. Therefore, we have designed and developed PCOM, a light-weight component system on top of BASE [BHSR04].

2.3.2. Data Manipulation Frameworks and Platforms

In this section the platforms and frameworks with collection of different algorithms to work on a data.

Matlab is a High-level language for numerical computation, visualization, and application development. It has predefined mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations. It supports Built-in graphics for visualizing data and tools for creating custom plots [oao].

Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a data-set or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. Advantages of Weka include, free availability under the GNU General Public License, portability, since it is fully implemented in the Java programming language and

thus runs on almost any modern computing platform, a comprehensive collection of data pre-processing and modeling techniques, ease of use due to its graphical user interfaces [oaf].

2.3.3. E-textile Development Environments

The following section provides the summary of related development environments to create e-textiles.

There are different models in the market right now to produce wearable or e-textile devices. As an example there are; Arduino Lilypad, which is cheap board and has low energy consumption [oaa]. It is designed to build soft interactive textiles. Raspberry-pi is cheap, bigger than Arduino Lilypad, powerful and has high-energy consumption [oai]. Intel Edison has very powerful microprocessor, Wi-Fi and Bluetooth low-level energy but bulky design [oam]. Beaglebone is very similar to Raspberry-pi, has a size of a credit card [oab].

There are different development environments specifically for almost every other hardware that is listed above. Arduino IDE is a text based programming language for Arduino hardware. Its language is similar to C++. It has 1-click deployment feature but no debugging.

In Gadgeteer hardware structure is defined visually (components and their connections). Automatic code is generated based on visually defined structure. It has debugging feature and also text based programming in C# [oan].

Espruino IDE is Google Chrome plug-in. It is a script and block based programming language. It can upload automatically to the Espruino microprocessor [oak].

Cloud9 is another browser based collaborative development environment. Version management integrated in the environment. It is also text based programming language in Nodejs and Javascript. Node-Red is visual flow based programming environment based on Nodejs. It runs as a Web Server. Programs can be created from any browser and offers components with higher-level features such as Tweeting, Mail. It can be used to connect the beans to the cloud and define behaviors such as alerting over email when temperature drops below some level [oad].

ICATch is a hybrid with visual and textual programming environment for Arduino based micro-controllers. Its targets are children and novices. It is based on bus architecture. And it is available to plug and play programming blocks and connect them to denoted program flow.

Fritzing is open source software for documenting and prototyping hardware involving micro-controllers sensors and actuators. Their purpose is to help hobbyists, designer and artists to produce their designs. It supports many components including micro-controllers, sensors, actuators and smaller parts such as resistors and connectivity modules. It has

functionality to export the design in order to share it with other users. SAM is visual flow based programming environment for the SAM Kit. It can translate visual representation into code that can be uploaded wirelessly to the boards [oac].

2.3.4. Interactive Development Environments for Teaching

Swift playground is introduced by Apple to make swift learning more interactive, effective and fun. Playground makes writing Swift code incredibly simple and fun [Vicng]. Whenever the users enter a line of code, the result appears immediately. For immediate result the Swift playground uses REPL (read-eval-print-loop) approach. Thus the playground does not require any compiling and keep listening the input from the users. The playground can be used for designing a new algorithm by watching its results every step, create new tests by verifying the work before applying into the test suite and experimenting with new APIs [oae].

Khan Academy has also launched an online development environment for teaching purpose. This online environment contains a set of tutorials based on the JavaScript and Processing languages and supports live coding environment like Swift playground, where the program updates the output as the programmers write [oah].

Processing is an open source programming language built for teaching the fundamentals of computer programming in a visual context and to serve as the foundation for electronic sketchbooks. One of the stated aims of Processing is to act as a tool to get non-programmers started with programming, through the instant gratification of visual feedback [oag].

Greenfoot is an interactive Java development environment for also educational purposes. Its main focuses are two-dimensional graphical applications like simulations and games. The aim of Greenfoot is to give programming education (object-oriented programming) at high school and early university level [oal].

Bluej is another Java environment specifically for first year college students. It is developed to eliminate some of Java's complex syntax and represents the object/class relationship visually. Special emphasis is placed in Bluej on visualization and interaction techniques to create an interactive environment to encourage experimentation and exploration [oaj].

Bret Victor, the owner of the work that was cited as an inspiration for the Khan system, emphasizes that; the goals of a programming system should be:

- to support and encourage powerful ways of thinking

- to enable programmers to see and understand the execution of their programs

As a live coding example Processing do not address any of these goals [Vicng]. Bret Victor further criticizes JavaScript and Processing as a poorly-designed languages since

they do not support ways of thinking and ignores learning process. Without these goals a live coding environments are worthless [Vicng].

3. Analysis

In this chapter, an analysis model based on the requirements elicitation will be presented. The main purpose of the analysis model is to structure and formalize the elicited requirements and to identify the objects that exist in the application domain. An analysis model is composed of three individual models: a **functional model**, represented by the use cases and scenarios, an **analysis object model**, represented by the class and object diagrams and a **dynamic model**, represented by the state machine and sequence diagrams [BD03]. The functional model was presented in the previous chapter. To complete the system requirements analysis model, object and sequence diagrams will be presented.

3.1. Analysis Object Model

According to system requirements that are presented in previous section the system should contain component view, code view, debug view, data view, and video view. The mockup of the system can be seen in figure 3.1.

The application consists of 3 stages; gathering, observing, testing. For simplicity and easy to read these categories will be explained separately. Firstly, the analysis object model of the gathering, which includes communication with the hardware and collecting the data, is shown in the UML class diagrams and is developed iteratively in the following section. The initial model is shown in figure 3.2.

In this model the software can connect to different kinds of e-textile components. Also in an e-textile component there can be different sub-components which have different functionality (actuators, sensors), addresses, registers and pins. The communication can be different between the actual component and the sub-components so the correct command should be sent. Even the communication through these different e-textile components can vary depending on the manufacturers. In addition to this, the two same devices can send the information in a different way, so that it has to be parsed differently. As it is stated in the requirements elicitation the system should be extendable, it should connect to different devices and the communication shouldn't be device dependent.

Moreover, the initial model is designed to only get acceloremeter and gyroscope values (I2C values). However one e-textile component can have not only I2C messages but also Analog and Digital sensors. In order to tackle this issue and make the communication device independent, the well-known and widely used Firmata protocol is used in this project.

Firmata protocol allows the ability to communicate with the devices without knowing the model of the device and it ensures that the message that is received will be in the same format. The Firmata protocol proposes three different communication ways. For analog sensors it has Analog messages, for digital sensors there are Digital messages and for ac-
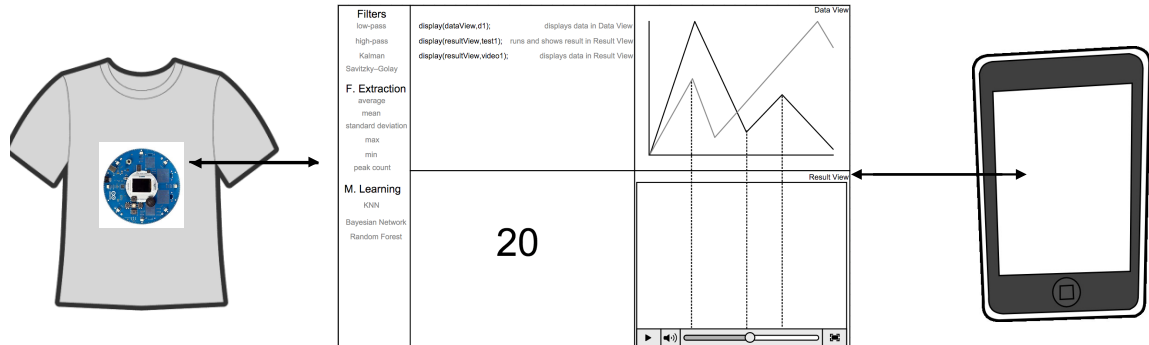
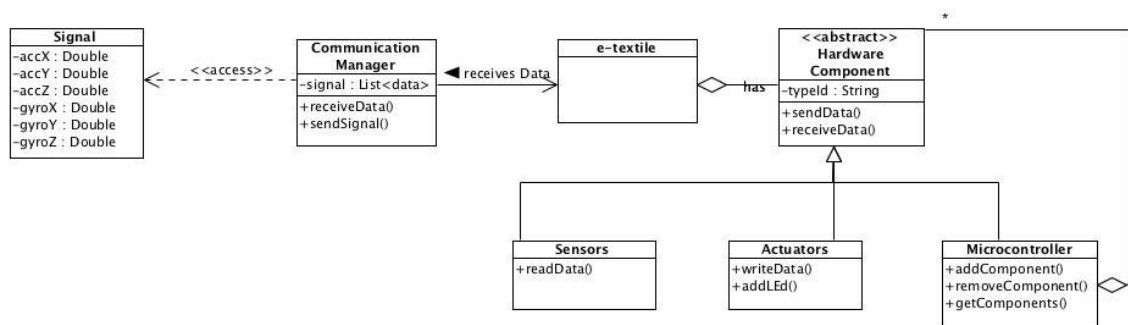Figure 3.1.: The representation of the System TextIt

Figure 3.2.: Old Analysis Object Model

celoremeter and gyroscope value it has I2C messages. The way to communicate and the response received from this communication include different meanings and different information. Yet in all communication type there needs to be a name of the data to call it later in the code, the default name is given according to message types. Because the data type changes based on the communication type in abstract class the data is shown as a generic type AnyObject. Later this will be implemented accordingly to the communication type. Although it is an array of numbers in Digital and Analog messages, it is a double array of numbers in I2C. The reason of being a double array in I2C is the e-textile component returns different information from different sub-components at the same time. In every message that is received has an acceleration x,y,z and gyroscope values of x,y,z.

The data is saved to a file after collecting data is done. In order to save the messages and their data to a file the abstract class implements Printable class. This interface enables to have a description method to return a string representation of information the class has. Since the types have different data types and different variables the description of the classes should be also different. The template pattern is used to solve that problem by allowing having different implementations of description method [GJHV03]. So that the description algorithm can be overridden by subclass to have a different behaviors by still following the overall flow.

To get the data or the description of every messages one by one can be time consuming and redundant. In those cases those different sub classes of messages can be treated as a single instance of an object since they all support the same variables. The composite pattern is used for this issue. The composite patterns allows users to treat group of different messages as a single object [GJHV03].

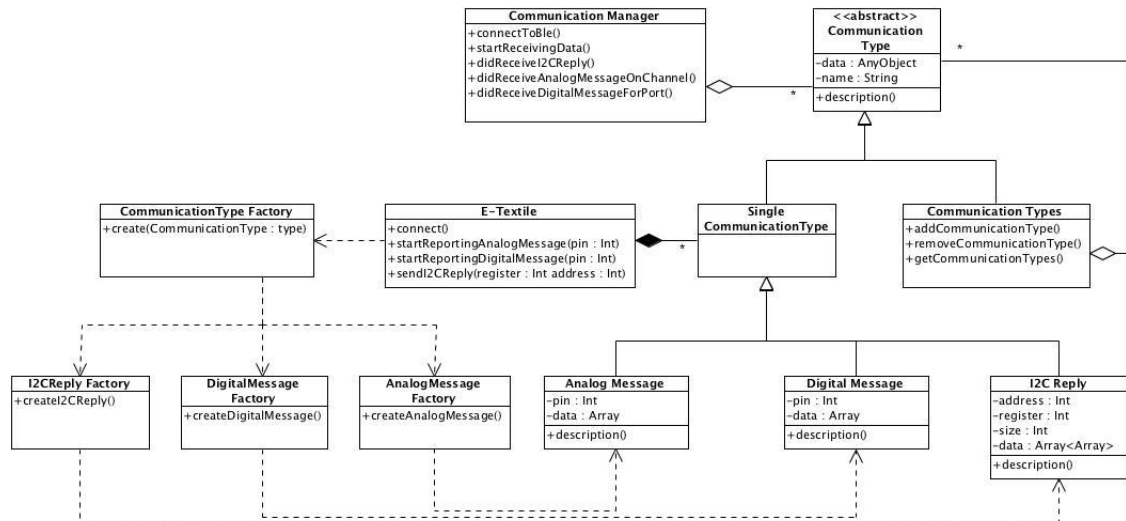The second iteration is presented in figure 3.3

Figure 3.3.: Analysis Object Model

The 3 different communication types use different protocols to communicate with the e-textile component. They send different messages and different inputs like pin number, address number or register number. There is a problem of having multiple communication types and they all contain different variables with different protocols. To achieve the problem an abstract factory pattern is used [GJHV03]. In figure in figure 3.3the abstract factory object is Communication Type Factory object. The 3 different communication types are the concrete objects that contain the necessary information to success the communication.

Another functionality of the system is observing data. The meaning of observing data is visualizing the data, manipulating it, applying filter methods, feature extraction methods or machine learning methods and visualizing it again. So that the users can extract the necessary information that they need and in every step they can follow the process and see how the data is changed.

To support users when they want to manipulate the data or apply some machine learning methods, they are pre-defined in the application so that these components are ready to use. The user can create an actual object from the components and can use it by giving the necessary inputs. Here in this model there are multiple components that are added to application. The components is the abstract object for the feature extraction, filter and machine learning methods. In deed they are also abstract objects and they include the actual implementation of the methods like; peak-detection algorithm, low-pass filter algorithm or k-nearest-neighbor algorithm 3.4.

In this model using an Abstract Factory pattern is considered, but for the simplicity to the user it is not used. Since the users are mostly developers; it is left as an Object-Oriented programming. The user can create the objects by using new() methods and start using them. The application itself is an IDE and the users are actually programming. To give freedom to the users and for the simplicity, abstract factory pattern is not used here.
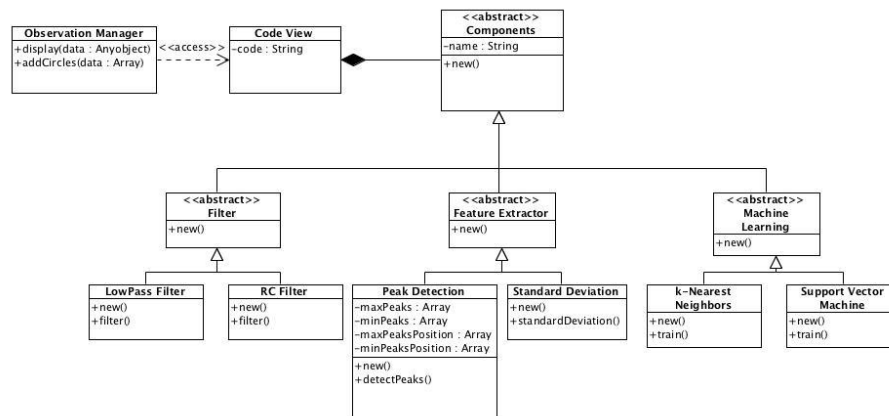

Figure 3.4.: Component Diagram

At the final stage there is testing part to see how different algorithms effect the data and

gives the best result. Here to get the accuracy the user should have a reference to compare. For example a jump count in Side-hops use case. The each component algorithms have different effects on the data. So here the user can create multiple use cases with different component algorithms and get the result of the each test cases and see the final accuracy in a table format. Easily the user should compare and see the best test case by comparing the accuracy of the use cases on the table 3.5.
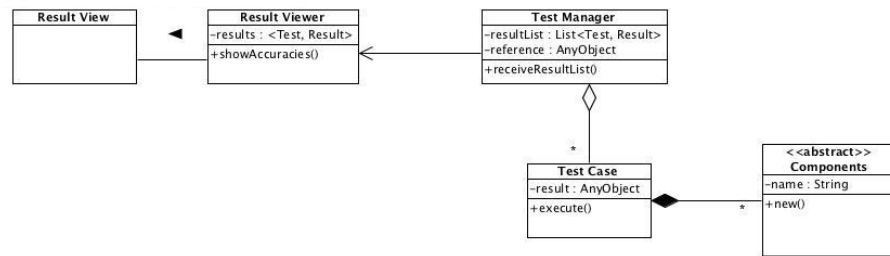

Figure 3.5.: Testing Diagram

4. System Design

The system design chapter focuses on transforming the previously described requirements analysis model into a system design model. The system design model contains a description of design goals, subsystem decomposition and strategies (hardware/software mapping, persistent data management, access control policy) for building the system. The goal of this chapter is to describe the system decomposition and to give a clear description of the strategies that are to be used for building the system.

4.1. Design Goals

The system design of TextIt was based on the following design goals:

Support of Different E-Textile Hardware

Different hardware manufacturers have different communication protocols. Even on the same hardware the messages getting values from different analog, digital or I2C messages can differentiate. To overcome the communication problem and to make it more device independent, the system has been developed with Firmata protocol. This Firmata protocol is implemented in Communication Manager. The component contains different messaging implementations while providing a uniform interface for the interactions. More information about this component is given in *Object Design* chapter.

   The Firmata protocol helps to overcome lack of uniform communication problem. In addition to this in order to connect to hardware at the first place, the BLE (Bluetooth Low Energy) library has been used. The BLE library helps to connect to hardware and start communication via Bluetooth 4.0. So that the system can connect to any hardware ,that is supporting BLE feature, wirelessly.
   To Separate system communication functionality from the main application functionality, the communication manager is used as a broker that isolates communication-related concerns (Broker Pattern).

Selection of Different Communication Message Types

The system should enable adding, removing, modification of different message types to communicate with the connected hardware. The Firmata protocol supports 3 different message types. The users should be able select which message types they want to use and how many different hardware components they want to communicate. For example there might be a case where the user wants to communicate via Analog Message with 3 different components on the hardware. This is accomplished by introducing E-textile Communication Selector Controller. In this controller user can add or remove by dragging

and dropping the message types and identifies their pin, register, address number. So that the user can connect to any component on the e-textile hardware. More information about this component is given in *Object Design* chapter.

Saving ad Displaying the Incoming Data From E-Textile

TextIt should help to users to understand which data, users are receiving. It is stated that the users can get 3 different messages (data) from many different hardware components. In order to identify which message type sends what kind of data, the system displays the message data to users while receiving the data. So that the users can see the data, that is currently receiving. This may help users to see whether it is the data that they want to get and whether there is a problem with the data before continuing working on the data.

Moreover to use the data later on and keep it persistently, users can save their data under the given file. The system enables to enter the file name and an option to save the data. This is accomplished by introducing Gathering Window Controller right after the message types are selected. More information about this component is given in *Object Design* chapter.

Supporting Scripting Language

The system should interpret a scripting language to enable its user to code their own program and use the tools. To achieve interpreting the user input live with REPL (read eval print loop) approach, using a light scripting language is decided. REPL (read eval print loop) method is chosen to support playground feature for beginner programmers, so that the system is more interactive and return the result of the code immediately. This encourages the inexperienced users when they see their result or problems immediately. Moreover in a result the system is light, easy to use, returns immediate results, and has a modern look rather than having an old way of compiler approach. For this feature JavaScript Runner component is added to the system. The reason choosing JavaScript and more information will be explained in *Object Design* chapter.

Should Connect and Push User Defined Code to Interactex Program

Interactex is the visual programming version of TextIt, and developed by my supervisor Juan Haladjian, M.Sc.. They target the similar goals for development of e-textile programming. TextIt can support the Interactex where it has some drawbacks because of the visual programming. In TextIt users can program their own custom algorithms and push their own methods to Interactex. The system supports this communication with Interactex and push the user defined methods. To achieve this feature plug-in pattern is used.

Supporting Basic Features of Integrated Development Environment

The system is designed to be an Integrated Development Environment(IDE) for e-textile development. Thus there are some common features that IDEs are supporting. As they can be found in many IDEs, the left panel is reserved for folder view. So that users can go

through different files or folders in the project folder. The system also supports creating a new file/folder.

In the toolbar not only to the classic functions are added like text size, font and color buttons but also some new functions that are specific to the system is like record button for connecting to e-textile hardware and getting the data from it and push button for sending the user defined code to other platforms like Interactex.

Since it is an IDE and allows users to program the debugger view is attached at the bottom of the view traditionally. Whenever there is a result of user code or an exception it is immediately printed to debugger view.

In order to keep integrity between models and view the widely used Model View Controller architectural design pattern is used. So that Model subsystem is responsible for application domain knowledge, View subsystem is Responsible for displaying data to the user and Controller subsystem is responsible for sequence of interactions with the user and notifying views of changes in the model.

Ready to Use E-Textile Development Components and Pre-defined Functions

The system contains some ready to use components to hep users to develop their e-textile program in an efficient way. These components are the methods that are widely used in e-textile development like signal smoothing, feature extraction or machine learning tools. So that users can gain more time and focus on their program in a more abstract way rather than focusing on the algorithm implementing.

The design of TextIt should also support addition of new components. Users can also define their own components. The feature is achieved by supporting JavaScript language so that the user input is directed to JavaScript context without making any changes on it. Thus the user defined components will be persistent on JavaScript context and users can use them as much as they want.

In addition to the components there are some pre-defined functions to realize some features in the system. These behaviors are the ones that should be supported inside of the system and let the user call those functions without concerning the implementations. For example display function to visualize the data or connect function to e-textile hardware or pre-defined parser methods are examples of pre-defined functions. These functions are system defined functions meaning they are written is swift and are ready to use when the system is opened. It is realized by using Repository Architectural Style. All the pre-defined functions are in the TextIt repository, so that subsystems are decoupled.

Portability

Textit is designed to run on Mac Os, that is why it is implemented in Swift programming language.

In addition to this, the system supports JavaScript language to interpret. However nothing on the system is developed specific to JavaScript. Later when another scripting language is wanted to use, only changing the interpreter is enough to evaluate the user input.

4.2. Subsystem Decomposition

The system is made up of three distinct subsystems which can be seen in figure 4.1:

1. **E-Textile subsystem** - consists of a different e-textile hardware to connect and gather the necessary data.

2. **TextIt subsystem** - consists of a number of components and pre-defined functions to support features that users need when they develop an e-textile application. Also it contains an interface to make communication possible by interpreting user inputs.

3. **Interactex subsystem** - is a visual programming version of e-textile development environment, and the system can communicate and transfer user defined methods to be used in Interactex.
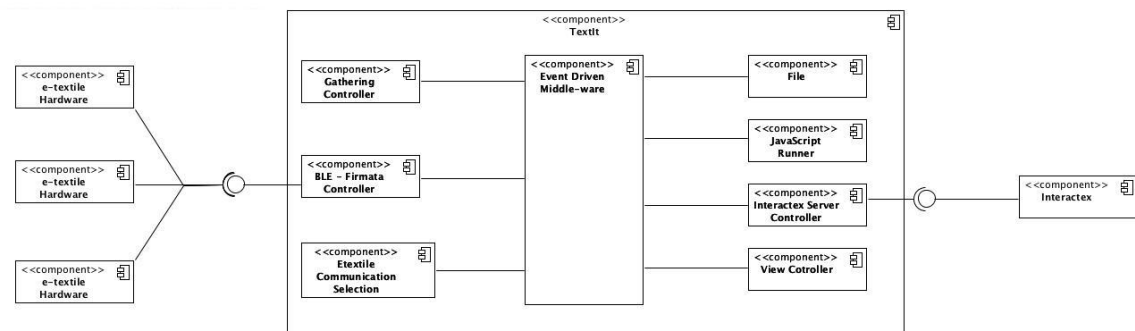

Figure 4.1.: TextIt system decomposition diagram

The focus of this thesis is on the TextIt component, thus the e-textile hardware and Interactex applications will not be explained.

4.2.1. Event Driven Middle-ware Architecture

To support low coupling of the TextIt sub-components, event-driven architecture is implemented. A center middle-ware controller has the control of the flow and calls the necessary sub-components to do their job. As the sub-components finishes their job and sends their events to the middle-ware, the middle-ware assigns the next task to other components or return feedback to the user.

4.2.2. Repository Pattern

There are pre-defined methods that are used often and make changes on the data. The users and the other components do not know the implementation but the purpose of the methods. The reason is that there are some functionality that TextIt supports, like filterin, feature extraction and machine learning, but the user actually do not know where it is implemented. To decouple these two different subsystems a repository pattern is used. The Repository pattern adds a separation layer between the data and domain layers of an

application. All the pre-defined functions are added to repository so that users can call them whenever they need from the repository. Later if new functions are added to the system again they can be added to repository to make them available to the users.

4.2.3. Broker Pattern

For the implementation of communication between the system (TextIt) and the hardware (e-textile) a Broker Pattern is implemented. The purpose of using this pattern is there are two distributed system, one is the TextIt that runs on MacOs and the other is an e-textile device, that needs to communicate and work on synchronously. So the communication is given to specific component and isolated from the system. Only one component is responsible to handle to communication. This component is responsible for coordinating communication, such as forwarding requests, as well as for transmitting results and exceptions. Moreover, this patterns helps to separate system communication functionality from the main application functionality, also isolates communication-related concerns.

4.2.4. Plug-in Pattern

When it comes to connecting and communicating with Interactex, plug-in pattern is used. In this functionality, TextIt works as an plug in to Interactex. A new components is defined by the user in TextIt and later this custom component is sent to Interactex, and Interactex systems uses it as a new custom component without knowing the implementation. So that a new element is added to the system by using another compatible system.

4.2.5. TextIt

The purpose of TextIt in this design model is to hide the complexity of different components and also make them communicate between each other so that the user input is fulfilled as designed. In order to fulfill the functional requirements of the system, TextIt has the JavaScript part to enable to users implement their own code and Swift part to help users to achieve their goals. The component diagram of TextIt can be seen in figure 4.2.
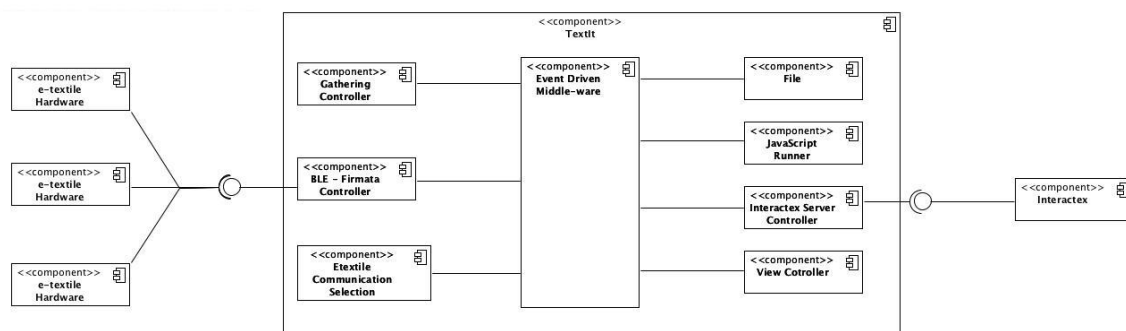

Figure 4.2.: Component diagram of TextIt

TextIt is decomposed into eight sub-components:

1. **BLE - Firmata Controller** - responsible for connecting to hardware, which UUID is in the allowed list.

2. **E-Textile Communication Selection** - responsible for starting a data flow from specific address or pin on the connected hardware.

3. **Gathering Controller** - responsible for receiving the data from the hardware, display it to the user.

4. **JavaScript Runner** - responsible for interpreting the user input in REPL (read-eval-print-loop) approach.

5. **Interactex Server Controller** - responsible for connecting to Interactex program and sends the user defined custom component.

6. **View Controller** - responsible for interacting with the user. It receives the input and shows the result as a GUI to the users. It has folder view, debug view, display view and text input view.

7. **File** - responsible for handling all the file functionality; reading, writing, creating etc.

8. **Event Driven Middle-ware** - responsible for handling the events and the flow between different components.

Smart Environment Integration Framework architecture can be represented as a four layered architecture which is shown in figure 4.3.

1. **Interface Layer** manages user interactions. It receives the user inputs and forwards the request to appropriate components. Interface layer also forwards the system response to the user.

2. **Application Logic Layer** is responsible for translating user's requests to appropriate commands. This layer also contains a component for the TextIt controllers.

3. **Storage Layer** persists the data that the data layer provides to the upper layers.

4.3. Identify Concurrency

TextIt has several threads to success the different task concurrently:

- **Main thread** is the main thread where application runs and executes the remaining functionalities.

- **JavaScript Thread** is used to run user code. Whenever the user inputs a new line of code it is activated o that the user input is interpreted.

- **GUI Thread** is used to make a change and visualize the data on GUI.

- **File Thread** is used to store user inputs. This thread is in a loop and whenever there is a change of the status of user input it saves the user code to a file.
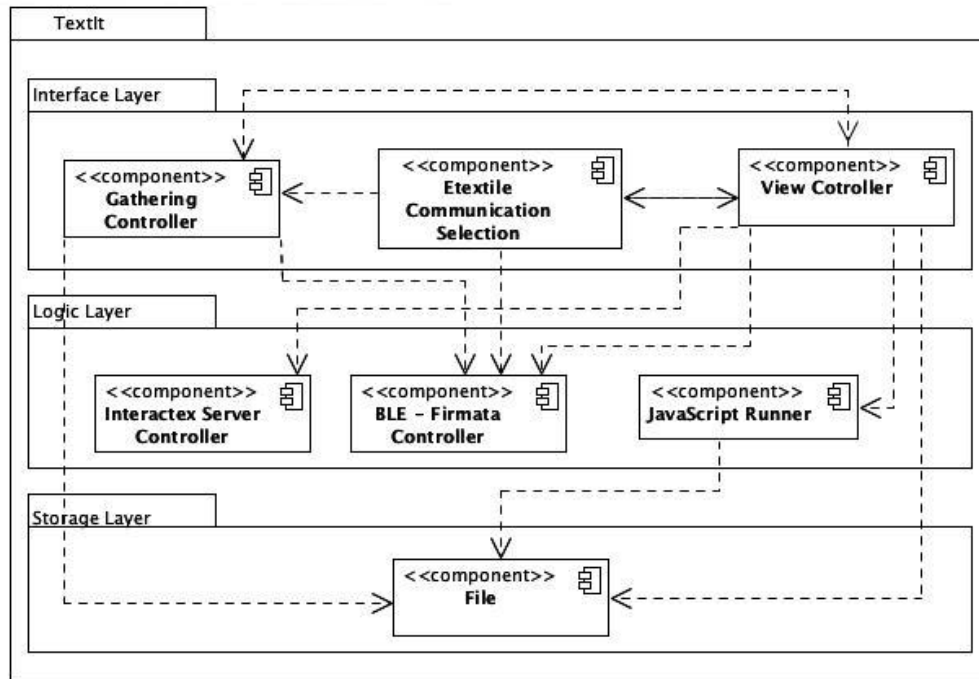
Figure 4.3.: Layered Architecture Representation of TextIt

4.4. HW/SW Mapping

The following section describes the assignment of the subsystems from figure 4.3 to hardware components.

Using the following HW/SW mapping, decoupling of client applications from the hardware is accomplished. This enables the creation of an interface to communicate and manipulate the e-textile components. Also thanks to design of TextIt and Interactex they can communicate with each other and can transform custom component. In this way the user has more freedom and can choose or use both of the visual and textual programming for e-textile development. Figure 4.4 shows TextIt deployment diagram. On the left side we can see different types of e-textile devices. There can be any number of types of e-textile devices, as long as they use an appropriate BLE and Firmata based protocol for communication with TextIt. The Interactex on the right contain modules for communication using Multipeer Connectivity Framework.

Because the communication is done via Bluetooth, this can be bottleneck and can be connected to one device because of the hardware limitations.
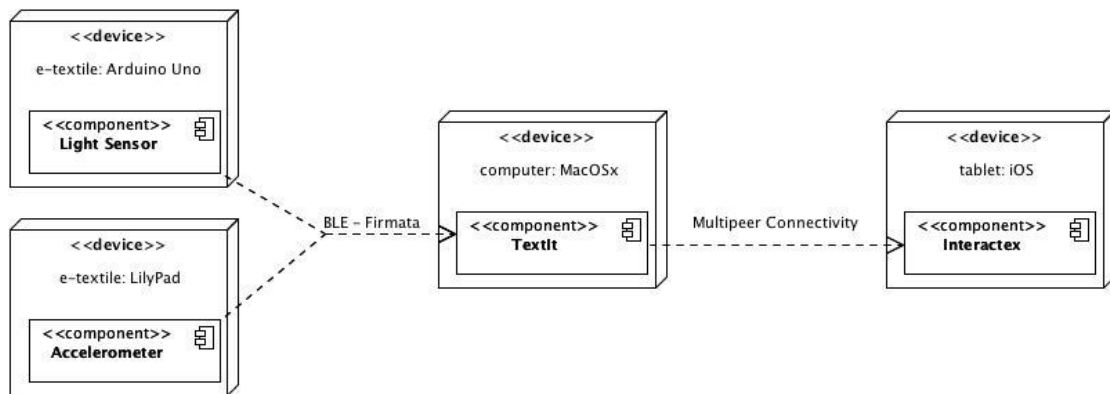
Figure 4.4.: Deployment diagram of the e-textile system

4.5. Persistent Data Management

TextIt uses files for storing persistent data. Files stores information about user code and the data received from e-textile components.
JavaSctipt context can also be stored in a file so that, the user defined components in the JavaScript can be persistent and can be used again after closing the application.

4.6. Software Control

The flow of the system is controlled by center controller using event driven architecture. All the components report to center controller by sending the finished event when they are done doing their task. After the center controller gets the event from one controller, it can assign another task to another component.

File thread used in the system is working isolated from other thread. But JavaScript, main and GUI threads have interactions between each other. JavaScript thread gets the user inputs from the user and if necessary may call main thread function or GUI functions. In this situation the functions in the main and GUI threads always tries to get them selves to the main thread or GUI thread to do their tasks.

4.7. Boundary Conditions

4.7.1. Initialization

TextIt is composed of eight components as already have been described. And there are some components that are initialized at the start of the system like Middle-ware Component and the View Controller, and others are not initialized until the user command has arrived.

At the starting of TextIt, the system checks the User/Document folder to see if there is already created a folder for the project. If there is an existing TextIt folder then it reads the main folder to load all the previous code of the user. If the system can not find any existing TextIt folder it creates a folder and an empty main folder to enable user to put some input.

JavaScript Runner components is being initialized as a singleton. Since all the user input should be interpreted in the same context and to achieve this there is only one JavaScript runner, thus there is only one JavaScript context.

Gathering Controller and Etextile Communication Selection components are lazy variables. At the first place there is no need for those components. When the user explicitly wants to use those components then the components are initialized. The same approach is used for the Interactex Server Controller, it is not started until the user starts it.

On the other hand, the BLE - Firmata Controller is started at the start of the system. It is initialized and started so that it is ready to connect when the user wants to. It searches nearby e-textile components and if the found peripheral UUID is on the allowed list, it connects to the peripheral and wait for the communication types, addresses, registers or pin from the user.

4.7.2. Failure

In case of the failure of one of task that is needed to be done, the system will provide an error message containing information to the debug view of the system. Then TextIt will wait for the next task from the user. In case of the failure of communication with the external components, the communication controllers have to be restarted.

4.7.3. Termination

For the full functioning of TextIt, all components do not have to be running. The components are temporary, they become active when they are assigned a task, then are terminated after their job is done. The only 2 components are need to be active for the all time is the ViewController to keep interaction with the user and the middle-ware to continue to flow of events.

All the objects in the TextIt components that are required to be persistent are stored into files whenever there is change or when a task is assigned explicitly. Thus, no special sequence of actions is needed for a safe termination of TextIt.

5. Object Design

The following chapter describes the object models of the components that make up the TextIt.

Although the previous chapters describe the system from the application domain, this chapter represents the transition to the solution domain. The main components of the TextIt are already defined, but in this chapter, they are refined in more detail. This chapter provides the definition of components and describes the interaction between components.

5.1. Specifying Interfaces

5.1.1. BLE - Firmata Controller

The BLE - Firmata controller are used for two actions. One is to find and connect to nearby peripherals and two is for starting messaging and asking for the wanted data. The controller takes the connection requests and the communication types, process them, constructs the appropriate BLE and firmata requests and passes them to BLE or Firmata libraries to communicate with the e-textile device. This controlller is used as a broker to take care of all the communication concerns with the e-textile devices. This broker class is a bridge between the communication libraries and the system. Other than taking control of the communication there is no other logic going on on this component. The interface of the component can be seen in figure 5.1.
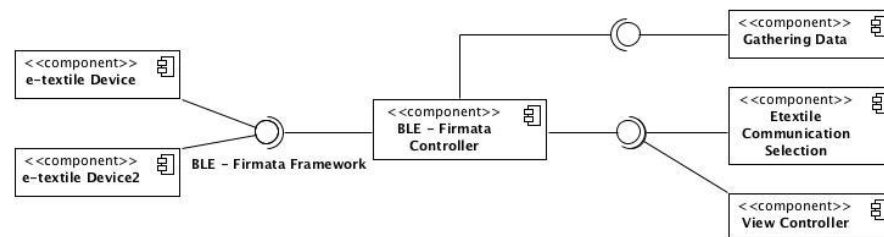

Figure 5.1.: The BLE Firmata Controller Interface Specification

To communicate through Firmata protocol, there 3 different messaging types:

1. **Analog Messages** - for retrieving information from Analog sensors

2. **Digital Messages** - for retrieving information from Digital sensors

3. **I2C Replies** - for retrieving information from I2C sensors like accelerometer and gyroscope

To initiate the messaging communication between the system and the device there are three different methods for three different message types.

To retrieve an Analog or Digital messages, the system asks to the device to get report from specific pin address on the e-textile device. However, for I2C replies, the address and register number have to be specified to tell which sub-component should be listened that is connected to actual device and the size of the incoming message.

1. **Digital Messages** - sendReportRequestsForDigitalPin:(NSInteger) pin reports:(BOOL) reports;

2. **Analog Messages** - sendReportRequestForAnalogPin:(NSInteger) pin reports:(BOOL) reports

3. **I2C Replies** - sendI2CStartReadingAddress:(NSInteger) address reg:(NSInteger) reg size:(NSInteger) size;

When the connected device start to sending replies to the system, Firmata protocol delegates are used and according to the message type the data is parsed if necessary and forwarded to next component. Here to give an example, all I2C replies are need to be parsed before any other actions. The reason for this is I2C replies are sent by bytes and all the actual values are divided into two bytes by the peripheral device before sending them. That is why a simple function is used to shift the byte array.

1. **Digital Messages** - didReceiveDigitalMessageForPort:(NSInteger) port value:(NSInteger) value;

2. **Analog Messages** - didReceiveAnalogMessageOnChannel:(NSInteger) channel value:(NSInteger) value;

3. **I2C Replies** - didReceiveI2CReply:(uint8_t*) buffer length:(NSInteger) length;

There are different delegates supported by the Firmata protocol to retrieve the message information from the devices.

The input for the messaging types are coming from user commands. E-textile communication selection component takes the messaging types to use and forwards them to the BLE - Firmata controller component. So this component can create necessary quarries and communicate through the protocols.

In this current version of TextIt, connecting to the peripheral command is coming from users. Whenever users wants to connect to available device, they can send a connect command through the View Controller interface.

When there is a connected device and the messaging types are decided, the incoming data is forwarded to Gathering Controller to show it to user and let them be saved in a file.

5.1.2. ViewController

This is the controller class of the main view. The view can be seen in figure 5.2. The component includes five sub components.

1. **Text View** - is the middle area for users to program their own code. The area is designed to let the users enter their JavaScript code. The entered input is sent to JavaScript Runner to interpret the code and according to type of a result, the result will be shown in a debug area as a feedback or in data view to visualize the given data.

2. **Debug View** - is the lower area to give exceptions or feedback about the user input. This debug view is designed to give feedback about the user code and show the exceptions, problems to warn the users.

3. **Folder View** - is the left area to show the folder structure of the project. The folder view shows the files and folders that are in the Documents/TextIt folder. This folder is the main folder for the TextIt system and all the read and write functions are done on this folder.

4. **Data View** - is the right area to visualize the any data that is assigned by the user. The user can specify the data to visualize on this area. So that users can first see the original data and then can see the affects of the manipulations that have been done to data.

5. **Toolbar View** - is the bar to give shortcuts of the pre-defined functions in the system. This is the bar view on top of the main view. Some of the pre-defined functions can be added here to make them easy to access. The reason for this feature is that users will not need to call those functions from their code and their code will stay pure user defined JavaScript code.

The interface specification of this component can be seen in figure 5.3.

5.1.3. Gathering Data

This component is a controller class of Gathering Data view. In this component the gathered data from e-textile device is shown to users in a view. Later than users can save the gathered data in a file by giving the file name. When the users are done with the view, the controller send an event indicating the component is done and regarding the user choice the gathered data is saved to file by using File Component. The view can be seen in figure 5.4. The interface specification of the component can be seen in figure 5.5.

5.1.4. Etextile Communication Selection

This component is a controller class of Etextile Communication Selection View class. This component is connected to GUI for users to select which communication types they want to choose. the component is basically allows users a list of communication types and let them add or remove via drag - drop as much as they want to choose and specify the name,
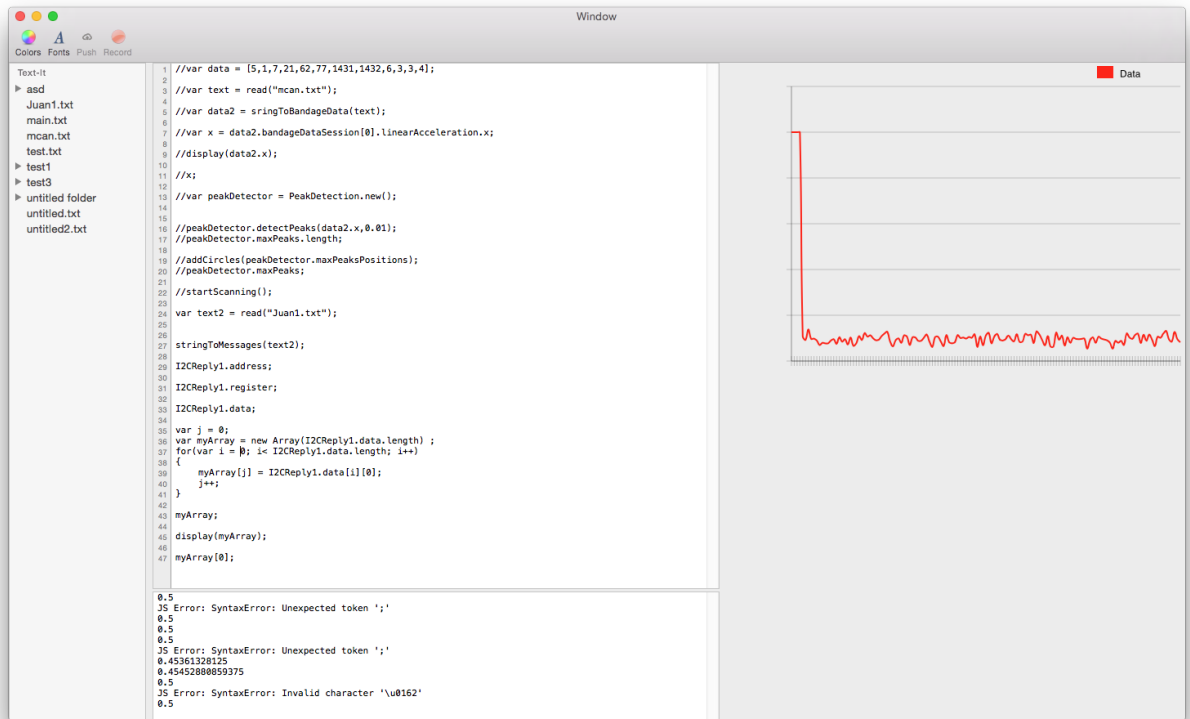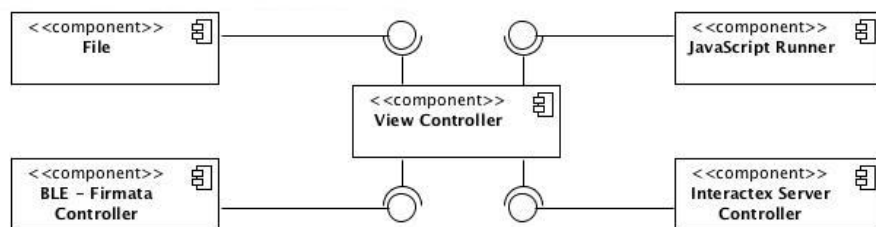
Figure 5.2.: Main Window of TextIt

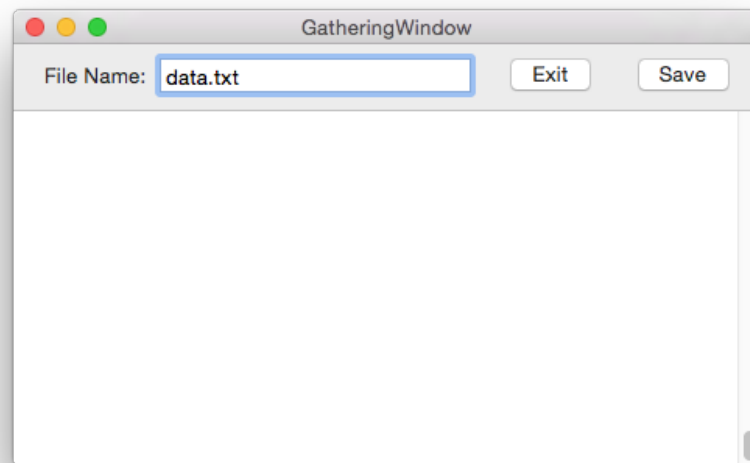Figure 5.3.: ViewController Interface Specification
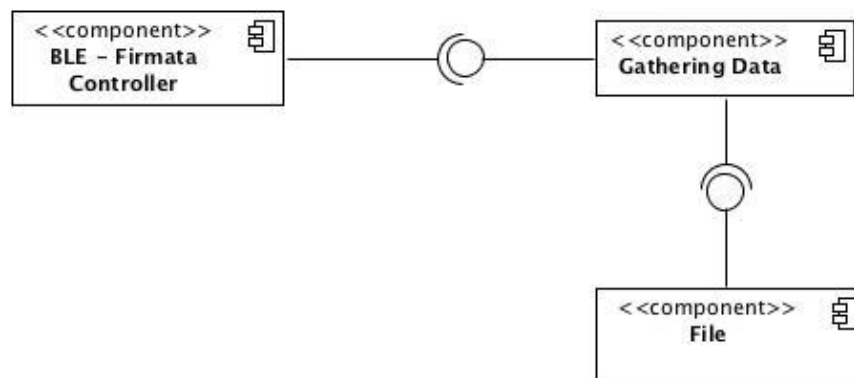
Figure 5.4.: GatheringData View of TextIt

Figure 5.5.: GatheringData Interface Speficitation

pin, address or register number to success the communication. After the selection is done by users the list of communication types are sent to BLE - Firmata controller by the center controller to start the data flow. The view can be seen in figure 5.6. And the interface specification of the ETExtile Communication Selection can be seen in figure 5.7.
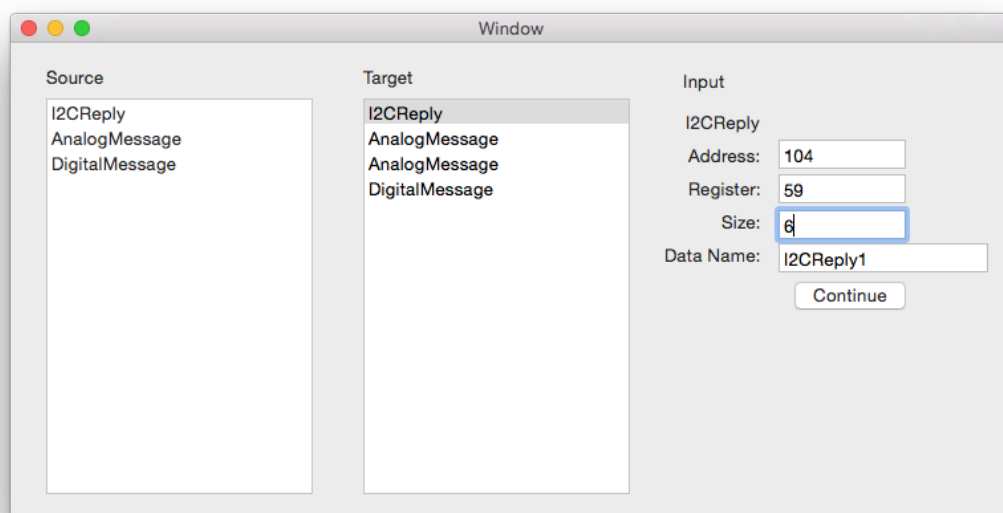

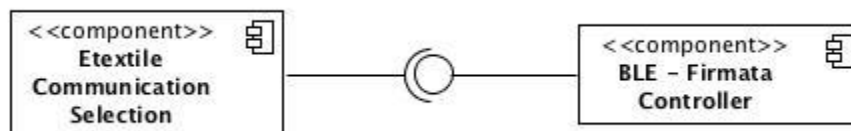Figure 5.6.: E-textile Communication Selection Window of TextIt

Figure 5.7.: Etextile Communication Selection Interface Speficitation

5.1.5. JavaScript Runner

To interpret all the user input, there is a JavaScript Runner component. This component is designed as a singleton so that there is only one responsible object that is controlling the interpreting tasks. Moreover as it is singleton there will be only one JavaScript context that contains all the information that is given by the user. In this component, also the defined filter, feature extraction and machine learning models are added to JavaScript context in this component. The interface specification of the JavaScript Runner is shown in figure 5.8.

In this component there is executeLoop() method that is running in a loop with three conditions to realize the user inputs. The loop either calls itself for every two seconds or

when the previous execution is done the loop starts again. Besides one of these two conditions there has to be a change that is done by the user to make sure the same code will not be executed.

In addition the execution of user input is done in another thread to make sure the main thread is free to do other jobs and not to freeze the GUI. So that user can continue writing a code, while the execution is done on the background.
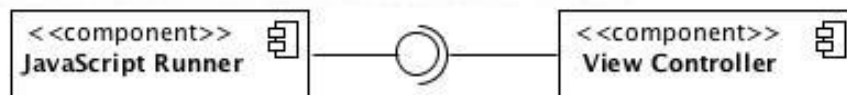

Figure 5.8.: JavaScript Runner Interface Speficitation

5.1.6. File

File Component is responsible for storing two different information. One job is to store the data that is gathered from e-textile device, the other job is storing the user code. The stored user code later can be retrieved when the system is started, so that user does not loose any information between coding sessions. With the same reason data is also stored to use the same data again in other programming sessions and can be manipulated without loosing the original data. The interface of the File component is shown in figureÃ 5.9.

The stored user code and data are sent to the ViewController to visualize and let the user to manipulate them. Indeed, ViewController sends the user code to File Component to make them stored in a file. However Gathering Data components is the component to send the gathered data to File component to store them. The gathered data is saved with their original message types alongside with the received values.

5.1.7. Interactex Server Controller

In order to connect to another development environment Interactex and work as a plug-in to Interactex, Interactex Server Controller component is used. This component is responsible for connection between these two systems. To achieve this goal, Multipeer Connectivity Framework is used and the systems are designed with a plug-in pattern. The interface specification of the Interactex Server Controller is shown in figure 5.10.
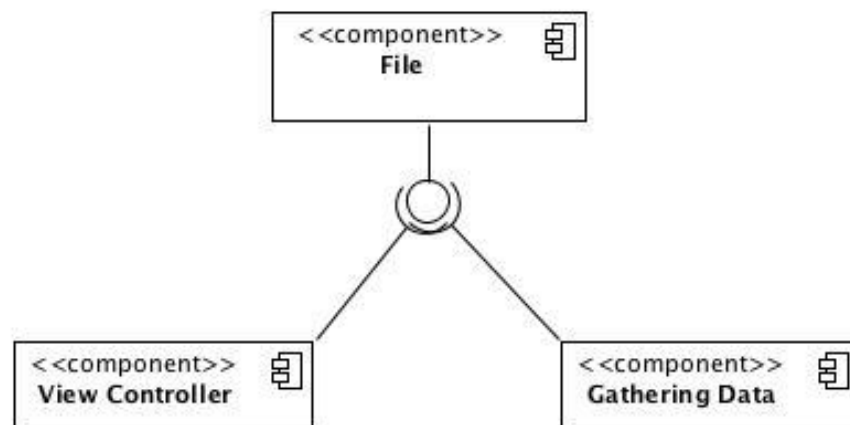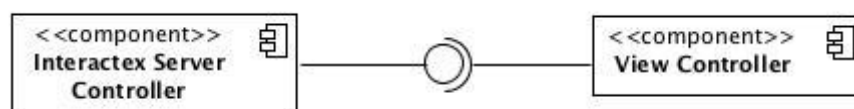
Figure 5.9.: File Component Interface Speficitation

Figure 5.10.: Interactex Server Controller Interface Speficitation

6. Conclusion

In the following chapter, a discussion about the design of TextIt is given, along with the future work and summary of what has been done during the process of writing this thesis. The discussion section describes the advantages and disadvantages of the framework, in addition to the suggestions given on how the system can be improved to have a better user experience. Future work would increase the usability of the system, and make the development of e-textile application easier and faster.

6.1. Discussion

TextIt is decomposed into seven components (Gathering Controller, Etextile Communication Selection, View Controller, Interactex Server Controller, BLE-Firmata Controller, JavaScript Runner, File), where every component is responsible for one aspect of the system (e-textile communication, user input handling, data receiving and manipulating).

An advantage of this design is that the implementation of every component can be done independently, as long as the interface of the component does not change. The components are implemented using the design-by-contract approach. The key concept of design-by-contract is viewing the relationship between a class and its clients as a formal agreement, expressing each party's rights and obligations. The existence of a contract also means that every component can be developed by a different person or a team, without knowing anything about the functioning of other components.

Another advantage of this design is that the components and the user code can easily be shared with other devices and applications (like Interactex) without the need for changing the components themselves. The advantage that emerges from the ease of distribution and the fact that the interpreted language context are stateless, is that all the components and user code can be replicated and deployed on multiple applications and devices.

A disadvantage of this design is that the user code is executed at the background in another thread in every 2 seconds. User input or data traffic load influences the response time of the system. The bigger the data load is or the more the user input is, the longer the response time of the system will be. Currently, for the use cases the response time of the system is around 1 second which is considered as a normal waiting time. Another main reason for the possible late response time is that for the simplicity of the thesis in every user change the whole user input is executed from the beginning except the visualization codes. If a more complex system is designed then because of the computation time and the re-execution of the user input the response time will be increased.

The response time is expected to decrease by understanding the change in the user code and execute the rest of the code instead of executing the whole code. Using this approach, the overload on the background thread and the computation load will be minimized. For the proposed solution there is no architecture change is required. There might be the case that the new components can be added but that does not require a change in the architecture. The proposed two architecture can continue to be used.
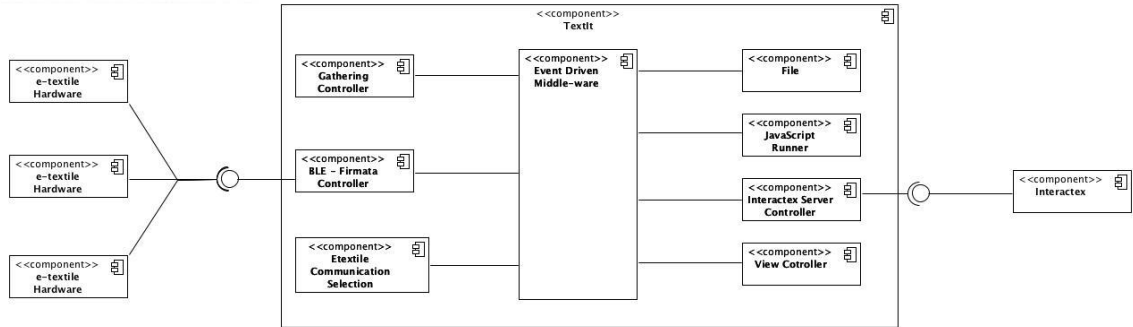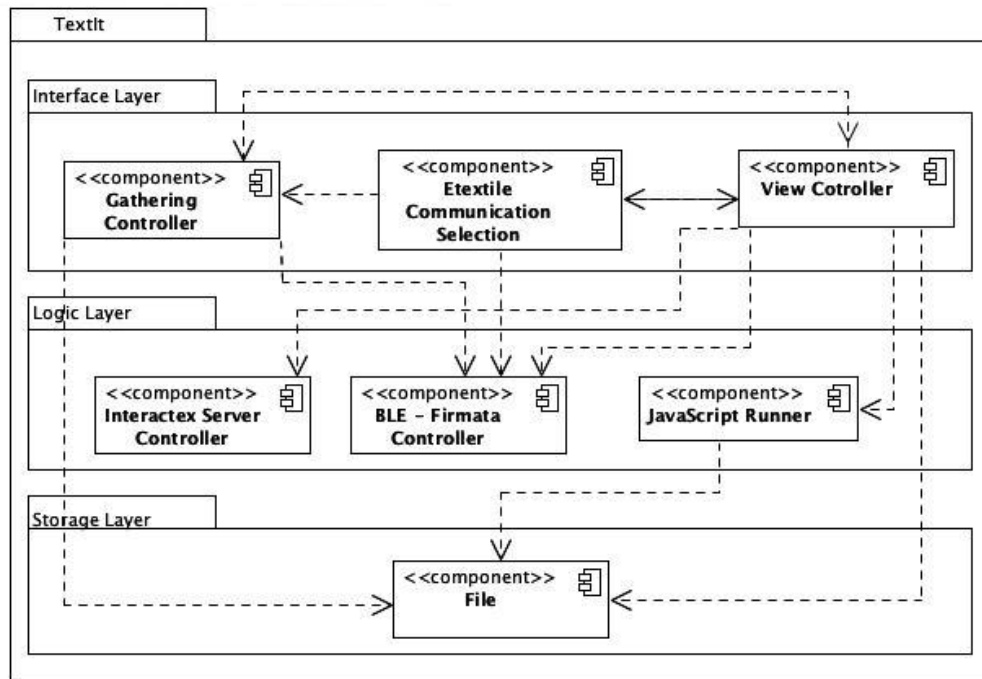

Figure 6.1.: Middleware Architecture of TextIt

Figure 6.2.: Layered Architecture of TextIt

6.2. Future Work

The emphasis of the future work should be on increasing the usability of the system and user experience of the e-textile developers. The usability of the system and comfort of the inhabitants can be increased by the following activities:

- implementing the missing functionality

- adding more pre-defined algorithms

- adding more communication protocols to support more e-textile devices

- full integration with Interactex or with other applications

- adding data visualization with the reference video

6.2.1. Implementing the Missing Functionality

Adding, removing and changing the name or the folder of the files from the system is not fully functional. In order to give full experience of the development environment these functionality should be implemented.

Also as discussed in Discussion section, instead of executing the whole code in every change, the user change should be identified and execute the code respectfully. This should increase the efficiency and decrease the response time. So that the behavior would be more convenient.

6.2.2. Adding More Pre-defined Algorithms

Right now there are three different main algorithm sections are included in the system (Filter, Feature Extraction and Machine Learning). These are the ones that are extracted from the use cases. However this sections can later be increased with new groups. So that the developer can find whatever he needs in the system and does not have to think about how to implement it.

Moreover even for the three algorithm sections, not every algorithms are implemented. For example for filtering low-pass filter, high-pass filter and RC filter are implemented. There can be find many other filtering algorithms for the data. Same applies for the feature extraction and machine learning algorithms. For machine learning section since the uses cases do not need and for the time pressure no sample algorithms are added. For the feature extraction section; peak detection, standard deviation, median, mean and fast Fourier transform algorithms are added.

6.2.3. New Communication Protocols for New Devices

As stated in the definition of an e-textile development environment in the introduction of this thesis, the system has to be able to communicate with different e-textile devices. In

order to do this BLE and Firmata protocols are used.

For the expansion of supported e-textile devices, the addition of new communication types is inevitable. By implementing more communication protocols, the more e-textile device can be connected to the system. This gives developers freedom when they choose a device.

Addition of every new communication type requires adding new parsing method if the protocols are using different representation for the data.

6.2.4. Full Integration with Interactex or with Other Applications

In this current situation TextIt can be able to send a sample code to Interactex as a custom component. As the system gets improved and added new components the counter components should also be added to Interactex. Moreover, the code should be cleaned from custom code that are used to visualize or to connect to e-textile device before sending to the Interactex. The custom visualization code or connection code are defined in the system so that they can be identified and cleaned from the code easily. All the communication and uploading features are already added to the system the missing part is the cleaning the code from custom methods before uploading it.

The integration with Interactex can only be an example that the system can be used with other e-textile development environments if needed. The new interactions can be added to the system independently thanks to architecture.

6.2.5. Data Visualization with the Reference Video

This reference video section can be also found in the mock up design. The purpose of this section is that when the data is recorded from an e-textile device, the users also record the activity. When they visualize the data on the TextIt, they can also give a video as a reference so that which point corresponds to which part of the activity. To give an example in one of use case called "One Leg Hop", the time of the jump is required. In order to calculate the duration of the jump the staring point of the jump and the landing should be identified. This is not an easy task especially if what peaks mean in the data. To give a user better insight about the data, the user can upload a video and the data is synced with the video. After that the user can see which point corresponds to which action in the video. With that approach the full information about the data is given to users. The area for that window is left empty for future development.
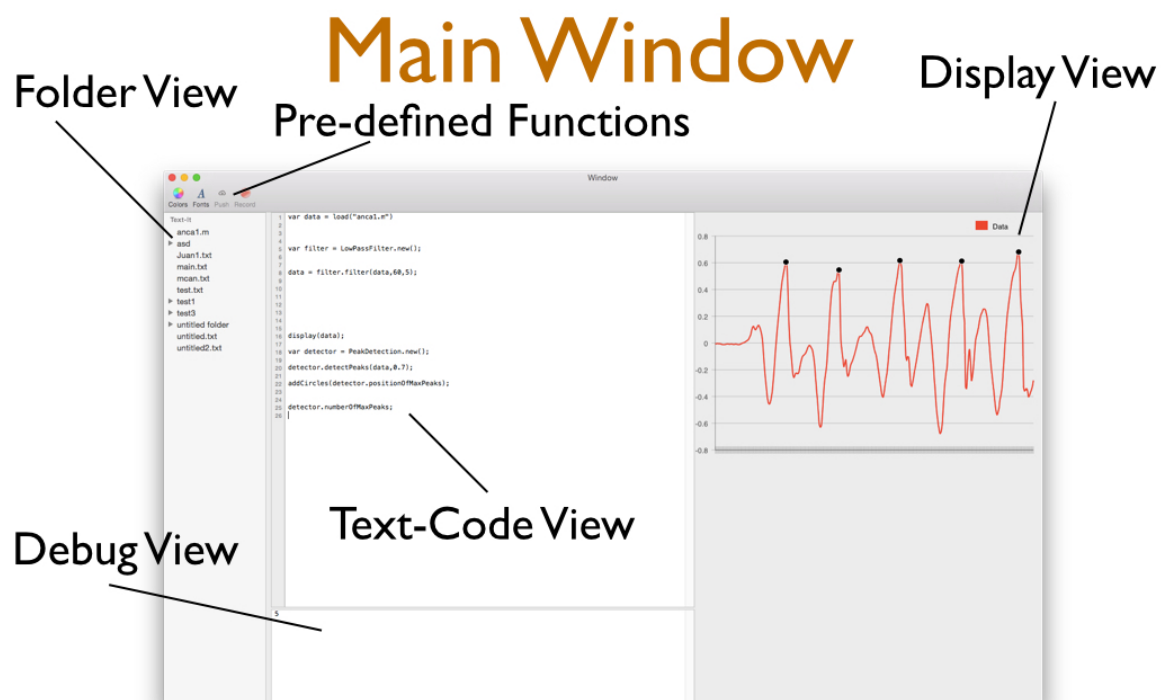
6.3. Summary

During the course of writing this thesis, an integrated development environment TextIt for developing an application for e-textile devices was designed and implemented. The emphasis was placed on designing and implementing an environment which would provide a data gathering, visualization and manipulation with help of pre-defined algorithms.

Also, the emphasis was on enabling communication with different e-textilde devices and connection with other independent environment such as Interactex. The TextIt is made up of six different components, with every component performing a different task in the system.

The TextIt enables the user to develop an application for e-textiles on an Application Layer with the abstractions. Besides that, the TextIt helps new beginners with a playground feature to make them understand the data better and use pre-defined algorithms for easy development. The systems helps experienced developers to build an application with an easy and efficient way without any vendor specification.

TextIt does not yet support the full feature of an development environment with some missing feature. Nevertheless, the design of the TextIt supports development of components that are discussed in the *Future Work* section that will add this kind of functionality to the environment. The data gathered by the TextIt along with the ability provided by the TextIt to control the data in the environment, enables the easy and fast development in the e-textile application areas. As a result of this research, new applications, which would add more value and interest to e-textile area, can be achieved with the help of TextIt 6.3.

Figure 6.3.: TextIt

Appendix

A. Glossary

Broker

The Broker architectural pattern can be used to structure distributed software systems with decoupled components that interact by remote service invocations. A broker component is responsible for coordinating communication, such as forwarding requests, as well as for transmitting results and exceptions [BMR$^+$99].

Layered Architecture

Partition the system into multiple interacting Layers with each layer representing a specific responsibility or concern of relevance and comprising all functionality that addresses that concern [BMR$^+$99].

Plug-in Pattern

The Plug-in Pattern is a software pattern for extending the behavior of a class with a clean interface. Often behavior of classes is extended by class inheritance, where the derived class overwrites some of the virtual methods of the class. A problem with this solution is that it conflicts with implementation hiding. It also leads to situations where derived class become a gathering places of unrelated behavior extensions. Another way of extending the behavior of classes is by means of configuration. In this solution the class is given a set of primitives with which the user of the class can configure the behavior of the class. The problem with this solution is that the behavior extensions are limited to the configuration abilities that the root class provides, and to the chosen implementation. In a sense the root class contain unnecessary information. Both solutions make it difficult to dynamically switch between different behavior extensions.

E-textiles

E-textiles, also known as smart garments, smart clothing, electronic textiles, smart textiles, or smart fabrics, are fabrics that enable digital components (including small computers), and electronics to be embedded in them.Smart textiles are fabrics that have been developed with new technologies that provide added value to the wearer. It is stated that "What makes smart fabrics revolutionary is that they have the ability to do many things that traditional fabrics cannot, including communicate, transform, conduct energy and even grow."

Repository Pattern

The Repository Pattern is a common construct to avoid duplication of data access logic throughout our application. This includes direct access to a database, ORM, WCF dataservices, xml files and so on. The sole purpose of the repository is to hide the nitty gritty details of accessing the data. We can easily query the repository for data objects, without having to know how to provide things like a connection string.

The repository behaves like a freely available in-memory data collection to which we can add, delete and update objects.

Event-driven architecture

Event-driven architecture (EDA) is a software architecture pattern promoting the production, detection, consumption of, and reaction to events. An event can be defined as "a significant change in state". This architectural pattern may be applied by the design and implementation of applications and systems which transmit events among loosely coupled software components and services.

Bibliography

[10ng]   Smoothing.   Website, 2015. Available online at`https://en.wikipedia.`
            `org/wiki/Smoothing.`

[1es]   E-textiles. Website, 2015. Available online at `https://en.wikipedia.org/`
            `wiki/E-textiles.`

[BD03]   Bernd Bruegge and Allen H. Dutoit. *Object-Oriented Software Engineering: Us-*
            ing UML, Patterns and Java. Prentice Hall, 2003.

[BHF+15]   Bernd Bruegge, Juergen Hoeher, Markus Fink, Juan Haladjian, Zardosht Ho-
            daie, Mertcan Yigin, and Han Xu. Kneehapp: a smart bandage for rehabilita-
            tion of knee injuries. *ISWC*, 2015.

[BHSR04]   Christian Becker, Marcus Handte, Gregor Schiele, and Kurt Rothermel. Pcom
            – a component system for pervasive computing. *IEEE*, 2004.

[BMR+99]   F. Buschmann, R. Meunier, H. Rohnert, P. Sommerland, and M. Stal. *Pattern*
            oriented software architecture: a system of patters. Wiley, 1999.

[BSGR03]   Christian Becker, Gregor Schiele, Holger Gubbels, and Kurt Rothermel. Base -
            a micro-broker-based middleware for pervasive computing. *IEEE*, 2003.

[Cho08]   Justin B. Chong. Activity recognition processing in a self-contained wearable
            system. Master's thesis, Virginia Polytechnic Institute and State University, 9
            2008.

[CKTT10]   Zysset C., Cherenack K., Kinkeldei T., and G Troster. Weaving integrated cir-
            cuits into textiles. *ISWC*, 2010.

[CS03]   Chee-Yee Chong and Kumar S.P. Sensor networks: Evolution, opportunities,
            and challenges. *IEEE*, 91(8), 8 2003.

[GJHV03]   Erich Gamma, Ralph Johnson, Richard Helm, and John Vlissides. *Design Pat-*
            terns: Elements of Reusable Object-Oriented Software. Addison Wesely, 2003.

[HSB+13]   Abdelkrim Hadjidj, Marion Souil, Abdelmadjid Bouabdallah, Yacine Challal,
            and Henry Owen. Wireless sensor networks for rehabilitation applications:
            Challenges and opportunities. *Network and Computer Applications*, 2013.

[HSBH14]   Niels Henze, Stefan Schneegass, Tobias Birmili, and Mariam Hassib. Towards
            a garment os: Supporting application development for smart garments. *ISWC*,
            2014.

[LAB08]   Paul Lukowicz, Oliver Amft, and David Bannach. Rapid prototyping of activity recognition applications. 2008.

[LAH⁺14]   Paul Lukowicz, Oliver Amft, Marco Hirsch, Jingyuan Cheng, Attila Reiss, and Mathias Sundholm. Hands-free gesture control with a capacitive textile neckband. *ISWC*, 2014.

[LMP⁺05]   P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. Tinyos: An operating system for sensor networks. *IEEE*, 2005.

[oaa]   Available   online   at`http://arduino.cc/en/Main/ArduinoBoardLilyPadSimple`. Arduino. Website.

[oab]   Available online at`http://beagleboard.org/bone`. Beaglebone. Website.

[oac]   Available online at`http://fritzing.org/home/`. Fritzing. Website.

[oad]   Available online at`https://c9.io/`. Cloud 9. Website.

[oae]   Available online at`https://developer.apple.com/swift/`. Swift. Website.

[oaf]   Available   online   at`https://en.wikipedia.org/wiki/Weka_(machine_learning)`. Weka. Website.

[oag]   Available online at`https://processing.org/`. Processing. Website.

[oah]   Available online at`https://www.khanacademy.org/`. Khanacademy. Website.

[oai]   Available online at`https://www.raspberrypi.org/`. Rasberry-pie. Website.

[oaj]   Available online at`http://www.bluej.org/`. Bluej. Website.

[oak]   Available online at`http://www.espruino.com/`. Espruino. Website.

[oal]   Available online at`http://www.greenfoot.org/home`. Greenfoot. Website.

[oam]   Available   online   at`http://www.intel.com/content/www/us/en/do-it-yourself/edison.html`. Intel edison. Website.

[oan]   Available online at`http://www.netmf.com/gadgeteer/`. Gadgeteer. Website.

[oao]   Available online at`www.mathworks.com/products/matlab/`. Matlab. Website.

[PGM⁺05]   PaoloCosta, GeoffCoulson, Cecilia Mascolo, GianPietroPicco, and StefanosZachariadis. Therunes middleware:a reconfigurable component-based approach tonetworked embedded systems. *IEEE*, 2005.

[Vicng]   Bret Victor.   Learnable programming: Designing a programming sys-
            tem for understanding programs.   Website, Available online at`http://`
            `worrydream.com/#!/LearnableProgramming`.

[WKU+07] Torben Weis, Mirko Knoll, Andreas Ulbrich, Gero Mühl, and Alexander Brän-
            dle. Rapid prototyping for pervasive applications. *IEEE*, 2007.