**Template For Final Project**

by: ChoChak Wong [ChoChack\_Wong@student.uml.edu](mailto:ChoChack_Wong@student.uml.edu)

Munkhajargal Narmandkh [Munkhjargal\_Narmandkh@student.uml.edu](mailto:Munkhjargal_Narmandkh@student.uml.edu)

Kirtan Patel [Kirtan\_Patel@student.uml.edu](mailto:Kirtan_Patel@student.uml.edu)

Mihir Patel [Mihir\_Patel@student.uml.ed](mailto:Mihir_Patel@student.uml.edu)u

Tyrone\_Turrell Tyrone\_Turrell@student.uml.edu

Adapted from IEEE documentation standards (particularly IEEE-1016 and IEEE-830)

Copyright © 1994-2015 by IEEE

*Permission is hereby granted to make and distribute verbatim copies of this document provided the copyright notice and this permission notice are preserved on all copies*.

**Table of Contents**

1. Introduction....................................................................................................... 3

1.1. Document Description......................................................................................... 3

1.1.1. Introduction.................................................................................................... 3

1.1.2. System Overview.............................................................................................. 3

2. Design Considerations..................................................................................... 4

2.1. Assumptions and Dependencies............................................................................. 4

2.2. General Constraints........................................................................................... 6

2.3. Goals and Guidelines.......................................................................................... 6

2.4. Development Methods......................................................................................... 6

3. Architectural Strategies............................................................................. 6

4. System Architecture....................................................................................... 7

5. Policies and Tactics......................................................................................... 7

6. Detailed System Design................................................................................... 8

7. System Installation and Execution………………………………………………… 9

8. Glossary............................................................................................................... 10

# **1. Introduction**

## 1.1. Document Description

The following document describes the design of the smartwatch, the components of it and how each part contributes to the overall function, how it is installed, the errors that need to be taken into account when making the product, the reliability or dependency of the product on certain systems, limitations affecting the construction and functionality, and policies or standards that affect the development.

### 1.1.1. Introduction

The purpose of the project is to discover the benefits of the smartwatch sensor in an individuals daily lives, specifically focusing on step counters. We will be discussing personal fitness and how the sensor helps its users stay active. The intended audience is all users who are fascinated by technology who want to make their lives more efficient and sports fans who want to keep track of their physical activity.

### 1.1.2. System Overview

* We will use SQL with an application with a user interface GUI. The app’s data and settings must be accessible by an administrator; such as user account, user information and user settings.
* The following functional requirements must be met by the system
* Web Interface
  1. Data Monitoring
* Collection of Data concerning user’s health
* Calculation of user’s health risk and assessment of danger
* Alerting of dangers to user and health professionals
  1. Account management
* Creation of account
* Modifying account: ie password, deletion, and status
  1. Subscription service
     1. Upgrade service
     2. downgrade service
  2. User Side
     1. Personal Account
     2. User Settings
        1. Modifying account: ie password, deletion, and status
        2. Upgrade/Downgrade service
     3. Database to store user health information
     4. Adaptable to evolution of hardware, software, and user needs
     5. Inclusion of user’s health professionals
* Other functional requirements
  1. Undecided
     1. The capability of health outcomes from collected data.
     2. The coordination with health professionals.
     3. The difference between the Health Monitor and Health Monitor Pro settings.
     4. The number of possible users per account or number of accounts per watch.
  2. Optional
     1. Real time emergency response
     2. The inclusion of advertising partners for health solutions

Non-functional requirements are unspecified at this point. The future may include diabetes.

* Technical Requirements-Feasibility
  + Server
  + Database

# **2. Design Considerations**

2.1. Assumptions and Dependencies

**Compatibility:-**  The Pebble step-counter must be backward compatible with earlier versions and future versions of the Pebble watch. The use of the C language could make it possible to use the Pebble step-counter in other devices with some modification.

**Extensibility:-** The underlying architecture allows for new functionality to be added.

**Maintainability:-** The project would be maintained in concordance with the Pebble Watch maintenance schedule.

**Modularity:-** The Pebble Watch has limited sensors and The Pebble step-counter will only use the accelerometer. The

**Reliability:-** Under normal conditions and use the Pebble step-counter will

simply make a count of the number of steps taken by the wearer of the Pebble watch. It will work admirably in a discrete fashion; from start to a reasonable number of steps and then stop, or be stopped. I will not satisfy continuous use. Nor combine separate usages or users.

**Reusability:-** The Pebble step-counter project was simplified from a more

extensive Pebble watch app. The addition of functionality would be

easy enough along the original app’s capabilities and the open source nature of the app also expands the possibility that newer or different implementations would be likely.

**Robustness:-** The essential nature of the step-counter is not robust, in fact, the simple accelerometer does not protect the user from “ghost steps;” from many dissimilar sources. The step-counter is vulnerable to slow sampling rates and miscounts from sensor data that is incorrectly source. ( The movement of the Pebble step-counter matches a step-motion but is not in fact a step.)

**Security:-** There should be no security concerns from the code that give rise

to any form of hacking or misuse by others.

**Usability:-** The interface of the Pebble is very simple and provides the

necessary capabilities; with four buttons possible for the selection and activation of the step-counter.

**Performance:-** There is an inherent flaw in the simple accelerometer that requires an increase of the sampling rate to retain a valid count regardless of step pace. Also, motion from driving, being driven, flight, etc… also must be calibrated against, if the device is to be used in continuous fashion.

**Portability:-** The portability of the software is possible because we are using

the C language but would be dependent on how C is handled by

the software or hardware of the new device.

**Scalability:-** The initial Pebble step-counter is designed for one user, and any

extension will have to include the memory, and energy use of the

Pebble itself as a boundary condition.

## 2.2. General Constraints

The Pebble Watch and the C language were used to make a simple step-counter and as such there were no constraints necessary as the the project was very much within the boundaries of normal use.

## 2.3. Goals and Guidelines

The Pebble step-counter should not require the user to know how to operate it, that is its interface should be intuitive. The user presses start and the step-counter calculates the number of steps taken by the user. The user presses stop and the total number of steps taken is displayed.

**2.4. Development Methods**

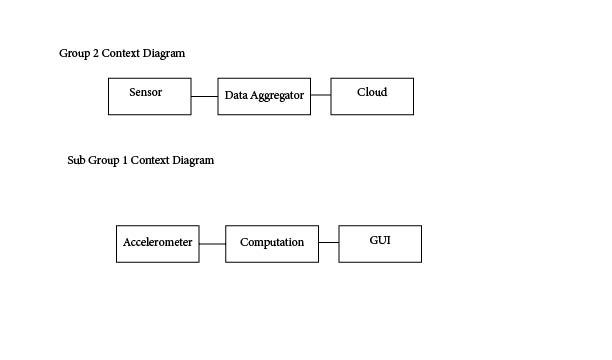
The scrum agile method was used with a reuse process. The scrum method was used for the initial design features of the Pebble step-counter and the actual code was modified from an actual Pebble Watch activities app. The easiest method to document, having the least requirements for documentation, the scrum method is highly recommended for any evolution of the Pebble Watch step-counter; until any actual functionality requires a real effort, in which case an incremental plan-driven process that more efficiently aligns proficient designers with the capacity to perform the tasks; quickly and professionally.

# **3. Architectural Strategies**

The simplicity of the Pebble step-counter is quickly designed using the existing application as a model. The MVC design strategy is already in place and no other considerations need be concerns. How the MVC is actually implemented and performs its functions is outside of the conditions of this project as formulated. The interface acts as view with the Pebble Watch handling control of a very limited model for the user. Memory, power, and all other boundaries for performance and use are limited within the range provided by the device. The architecture is entirely given and the C language is already in use with an open source organization. There are no other systems interacting with the step-counter, no further planned extensions or external databases and many of the architectural strategy questions are not applicable to this project.

# **4. System Architecture**

The system architecture was provided by the Pebble Watch device. No decisions were made to decompose or understand the workings of the system. The system is a black box and all components are hidden. The only view possible of the architecture is that input is entered and used to compute the output. The entire question of the system architecture is not applicable because it was not designed and it need not be understood for the development of this app.



The project functions as a sensor and permits data aggregation to be used by a cloud

app. The step-counter collects data from the accelerometer, computes the step count

and displays it on the watch face. Ideally, a MVC pattern would be used to handle the

different conditions for accelerometer data collection, how it is computed, and then

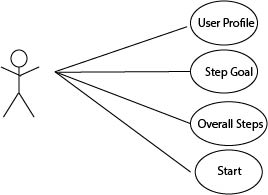
displayed.

# **5. Policies and Tactics**

There were no policies and tactics used or considered in this project, outside of

using the Pebble Watch and its open source code in the C language. However, the

code should be commented on and as functional as possible.



The idea is to simplify the user experience and app functionality even more than the

diagram shows, while retaining the ability to re-establish these functions and expand

capabilities further.

# **6. Detailed System Design**

The system is adequately documented through the Pebble Watch distributor for

open source use.

#### [Foundation](http://developer.getpebble.com/docs/c/Foundation/)

The core of the Pebble SDK

.<http://developer.getpebble.com/docs/c/Foundation/>

#### [Graphics](http://developer.getpebble.com/docs/c/Graphics/)

Low-level drawing routines.

<http://developer.getpebble.com/docs/c/Graphics/>

#### [User Interface](http://developer.getpebble.com/docs/c/User_Interface/)

Everything related to the user interface.

<http://developer.getpebble.com/docs/c/User_Interface/>

#### [Worker](http://developer.getpebble.com/docs/c/Worker/)

<http://developer.getpebble.com/docs/c/Worker/>

#### 

#### [Standard C](http://developer.getpebble.com/docs/c/Standard_C/)

Standard C types, functions, constants, etc.

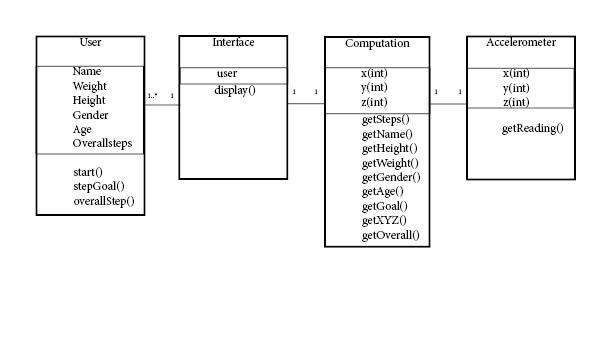
<http://developer.getpebble.com/docs/c/Standard_C/>

Simplification of the project reduces much of the functionality of the classes shown

below, especially when concerning the User class; which has all profile ( name, weight,

etc…) removed. A start function and a stop function, which displays the overall steps is

all that remains.



# **7. System Installation and Execution**

The Pebble Watch owner would download the app from a repository. The menu

screen would direct them to hit the start button and the app would begin.

# **8. Glossary**

**Accelerometer** - a device that measures [proper acceleration](http://en.wikipedia.org/wiki/Proper_acceleration) ("[g-force](http://en.wikipedia.org/wiki/G-force)"). Proper

acceleration is not the same as coordinate acceleration (rate of change of velocity).

For example, an accelerometer at rest on the surface of the Earth will measure an acceleration [g= 9.81 m/s2](http://en.wikipedia.org/wiki/Standard_gravity) straight upwards. By contrast, accelerometers in [free fall](http://en.wikipedia.org/wiki/Free_fall)

orbiting and accelerating due to the gravity of Earth will measure zero.