LogMe - Where have I been, what have I done?

João Augusto Curto Leiria Dias
joao.dias@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2014

Abstract

Lifelogging is the process of tracking personal data generated by yourself and your life, through the use of some tools. Lifelogging and self-tracking tools are an expanding concept, which has become increasingly popular in the last years, in part because technology nowadays allows doing that, but also because people want to learn more about themselves, thus they can improve their lives. Today we have a good variety of devices that register and analyse the data collected, such as FitBit, Narrative Clip, etc. In the beginning of lifelogging activities, people had faced some difficulties, due to the fact that computers were the only tool available for that, which was a huge limitation, as they had to wait till being near a computer to register the data. Although this is no longer a problem, there are still other problems nowadays, like which is the best tool to use, lack of time, lack of motivation and, specially, different data sources, which makes very hard to gather all the information we are logging. Therefore, we have created a system that allows both manual and automatic data collection, thus facilitating the collection process and also running it efficiently.

Keywords: Lifelogging, Self-tracking, Personal informatics, Quantified self

1. Introduction

Personal Informatics refers to a set of tools that allow users to collect relevant personal information for self-monitoring and self-reflection, for example, weight, distance walked, geographic location (tracks), computer activities, etc. Self-reflection is also referred as Reflective-Learning, which means returning to and evaluating past experiences in order to promote continuous learning and improve future experiences. These tools help people to have a better knowledge about their behaviors and habits.

The fact that a person self-tracks a particular activity doesn’t mean that will automatically improve some aspect relatively to it. However it will definitely provide previously unknown insights. These will allow that person to take the necessary steps to perform important changes for a possible improvement. That is, thus, one of the main advantages of self-tracking. This type of information collection is associated to names like lifelogging, living by numbers, personal analysis, quantified self and self-tracking.

Until recently this kind of tools were only available for traditional computers, which brought some problems, as for example, we could only make the register if we were at a computer. This limitation, sooner or later, would lead to a big lack of motivation to continue to register the activities. More recently, the number of mobile devices users (smartphones and tablets) has exponentially grown, associated to this, is the growth of the number of tools to this kind of practices. Another important promoter in this area is the Quantified Self group. It is an international collaboration of users and makers of self-tracking tools. They have annual conferences, to discuss and present new tools and systems for lifelogging.

2. Related Work

The personal informatics systems allow the collection of personal relevant information so that its user can obtain some self-knowledge, for example, food ingested, how many steps, etc. And then what to do with this data? The vast majority of these systems just merely create a collection of all the data collected for future analysis, not allowing the collector to use it for introspection. In the paper [6] by Rivera-Pelayo et al., it is presented a framework that combines reflective learning techniques with quantified self-tools, where were identified three main dimensions of support, including: tracking cues, triggering and, recalling and revisiting experiences:
a) Tracking cues: capturing and keeping track of certain data as basis for the whole reflective learning process.

b) Triggering: fostering the initiation of reflective processes in the learner, based on the gathered data and the analysis performed on it.

c) Recalling and revisiting experiences: supporting learners in recalling and revisiting through the enrichment and presentation of data in order to make sense of past experiences.

Another relevant work is presented by Li et al. [4], where was conducted a study that attempted to understand the main difficulties in the use of tools for self-tracking. From that study, a model based on five states has born, respectively: preparation, collection, integration, reflection, and action. It is named Stage-Based Model of Personal Informatics Systems.

The Preparation stage occurs before people start collecting personal information. This stage concerns itself with people’s motivation to collect personal information, how they determine what information they will record, and how they will record it.

The Collection stage is the time when people collect information about themselves. During this stage, people observe different personal information, such as their inner thoughts, their behavior, their interactions with people, and their immediate environment.

Integration is the stage that lies between the Collection and Reflection stages, where the information collected are prepared, combined, and transformed for the user to reflect on.

The Reflection stage is when the user reflects on their personal information. This stage may involve looking at lists of collected personal information or exploring or interacting with information visualizations.

The Action stage is the stage when people choose what they are going to do with their newfound understanding of themselves. Some people reflect on the information to inform them on what actions to take.

While some logging can easily be automatic (e.g. location, weather, calendar data, etc.), some by definition require self-logging. Food intake, pain levels, and mood are examples that are quite difficult to automatically determine, but can be quite easy for users to enter on their own. However, in practice, people tend to forget and/or lack the motivation to self-log. To address this issue, Bentley and Tollmar in [2] show how a simple reminder on a mobile phone can increase self-logging frequency.

Your.FlwingData1 (YFD) is a self-tracking system that uses twitter as storage for records. Users can record what they eat, when they go to sleep, how much television they watch, how many cigarettes, their weight or anything else they want. To use YFD users just have to follow @yfd on twitter, and then, by sending private messages to @yfd, they’re recording their actions. After that, when accessing a personal space in YFD webpage, users can analyze their records through some nice graphs. Like any other system that relies on text provided by the user, YFD will suffer from data quality problems. These problems occur, typically, when the user refers to something by more than one name.

Burns et al.[3] talks about Interface Complexity and Engagement, respectively devices or applications with high complexity and high engagement interfaces. High complexity in that a large amount of information is presented to the user, usually in the form of numbers and graphs. And high engagement in that users must commit time to regularly monitor and understand the information presented. Active users who employ technology in a supporting role will value this "rich" presentation of information. Their high level of intrinsic motivation to exercise means they will be willing to commit the time and effort to engage with such interfaces. Less active users, who employ technology in a motivational role, will benefit from devices that use low complexity and low engagement interfaces. These interfaces should be informative, yet simple to engage with. And if users choose to disengage with the interface, they should be able to re-engage at a later date with minimal effort.

Through the analysis of these works and identified problems, we have reached the five main requirements to our application:

• R1: Reflective Learning supportive
• R2: Data Quality
• R3: Balanced Interface Complexity
• R4: Easy to use and Non time consuming
• R5: Keep motivation

3. Solution
In order to build a system which properly supports reflective learning (requirement R1), we followed the approaches from Rivera-Pelayo et al.[6] and Li et al.[4] discussed earlier, by implementing a system with support to the three main dimensions of support (tracking, triggering and recalling), and proper means to address the main barriers identified

1http://your.flowingdata.com
for each one of the states in the Five-Stage-Based model.

Since the main goal of this project is to perform an easy and quick way to collect data from different lifelogging data sources, this will accomplish the tracking dimension needs. To support the triggering dimension, we implemented a system based on active notifications to the user, in order to initiate the reflection process and help to keep the user motivated (requirement R5). These notifications are mostly to remind the user to insert values that are missing.

Most of the lifelogging systems that rely on text data provided by users have to deal with data quality issues. To surpass this problem there are several possible approaches: auto completion/suggestions and/or predefined values. We used all of those approaches in our application. Data quality is directly related to normalization. To effectively process data and make it suitable for natural language processing is a challenging issue due to the fact that informally inputted text data is usually very noisy, that is why normalization is an essential step to go through if we want high quality data. These cover the requirement R2.

In terms of Interface Complexity and Engagement, our system tries to please both sides. We wanted to give users the power to select the level of interface complexity, in other words, the amount of visible information. Relatively to the level of engagement, we tried to keep it as low as possible, although, it is directly related to the level of complexity, in a way that, when one is high, the other is also high. Users which prefer high level of information, would have more information to analyze, so, would have to commit more time to it, but again, our objective was to always have interfaces that are easy to engage, no matter what is the amount of the information presented (requirement R3).

By providing a clean and easy to use interface, we also covered requirement R4 and R5. If an interface it’s easy to use, most likely, it will be non time consuming, and by not being time consuming, it will keep the users motivated to continue using it. Requirement R5 will also be covered by the quality of the conclusions presented.

Our system consists of three main blocks: Setup, Collection and Visualization (See Fig.1).

In the setup block is where the user feeds the system with existing data, whether by providing its credentials to an existing and supported system, like fitbit, or by uploading an excel spreadsheet with personal records. It’s also in this block where the user assists the system by locating and identifying redundant data between the different data sources. All the information is stored in a database for persistence.

The collection block is where this system supports the collection of information. In order to record an action, for example “eat”, user has to set it beforehand. The subject of an action also has to be preconfigured in order to avoid data quality issues.

The visualization block is where present user with the views of the data provided to the system, being able to choose a variable, period of collection and if it is going to be shown in a form of a list or exported to CSV.
needs.

We mostly relied on the 10 Usability Heuristics for User Interface Design [5]. These are 10 general principles for interaction design and they’re called “heuristics” because they are broad rules and not specific usability guidelines.

1. Visibility of system status
2. Match between system and the real world
3. User control and freedom
4. Consistency and standards
5. Error prevention
6. Recognition rather than recall
7. Flexibility and efficiency of use
8. Aesthetic and minimalist design
9. Help users recognize, diagnose, and recover from errors
10. Help and documentation

In our application, data collection can be achieved in two different ways, manually and automatically. Next we will detail each one of these options.

**Manual Data Collection:** Users can insert a value directly in the application homepage, by selecting an already created variable. If the variable was created with a set of allowed values, those will appear in the suggestions’ list, used to minimize the errors (See Fig.2). Users can also access the ”Configure Vars” option on the menu, to create a new variable.

![Figure 3: Manual data collection: creating a new variable](image)

**Automatic Data Collection** The automatic data collection requires a few more steps. The user has to choose which variable will be collected from the external source and that is presented in a list of possible variables. For example, in fig. 4, we created a new variable named Lunch Location, which is going to collect its values from the column ”Almoço (localização)” in the Google Spreadsheet.

![Figure 4: Automatic data collection: new variable creation from Google](image)

**Merge Data from Multiple Sources** Combining data residing in different sources and providing users with a unified view of these data, we think that has been one of the gaps in lifelogging. Each data source can be seen as a puzzle piece, where you can see a general shape of a puzzle’s picture, but you can’t see the entire picture. Without every piece, there are gaps that you can only fill in by guessing what the full picture looks like, which can make you make incorrect assumptions about what may best solve a problem and, at the most extreme scenario, can harm you. When put together, multiple data sources can give us powerful insights to make important decisions, allowing us to better identify the context of our daily habits.

In this thesis we used two different sources: Google spreadsheet and FitBit. Our choice fell to these sources as they are largely used and are available:

- **FitBit:** one of the most used and known lifelogging gadgets, tracks steps taken, distance traveled, calories burned, stairs climbed and active minutes throughout the day. At night, it tracks sleep;
- **Google Spreadsheet:** it’s easy to access, create and edit your spreadsheets anywhere (phone, tablet or computer), changes are automatically saved as you type, you can access your spreadsheets anywhere, even without detection network (just activate the offline edit to work lay in the browser or files on your mobile device);

**Data Normalization** Normalization means adjusting values measured on different scales to a notionally common scale, allowing it to be compared in a meaningful way. Text normalization is an important problem in natural language processing, it means converting ‘informally inputted’ text into the canonical form, by eliminating ‘noises’ (Zhu et al.).

If the user manually inserts a value that doesn’t appear in the suggestions’ list, the application will
present two options: add the value inserted to the allowed values or change the value inserted to one of the allowed values (See Fig. 5).

Figure 5: Dealing with conflicts: changing the value inserted

If the variable collected from an external source has been configured with a list of possible values, and if there is any conflict between the imported values and the allowable values, a list of conflicts will be presented to the user (see. fig. 6). At this point, it will be given to the user several options to solve these conflicts.

Figure 6: Automatic data collection from a Google Spreadsheet: dealing with conflicts

An option ”Insert Now” helps the user to directly go to the insertion function.

3.1.2 Reminders’ System

Motivation is the combination of desire, values and beliefs that drives someone to take action. These motivating factors and/or lack of them, are at the root of why people behave the way they do. By keep forgetting to register lunch, for example, when you notice it (maybe the end of the week?), you don’t have the drive, nor maybe the mental availability, to do it. Thus, motivation drops and the probability of achieving your goal falls tremendously. Therefore, a reminders’ system is so important and essential. If you have something that keeps you reminding what you have to register, then you start seeing results, your motivation ascends, turning into a vicious circle, and the odds of a good outcome are greatly increased.

In our application, we have a reminders’ system that goes through all the variables that have a frequency associated and will check which of these variables have missing values for the current day. Then, it presents the user a list of reminders to inform which are the missing values (see. fig. 7).

Figure 7: List of reminders that shows the missing values

3.1.3 Data Exporting

To see the values collected, users must click in the ”View” option on the menu. Users can choose the variable to present its values, with the possibility to choose the period associated and it’s possible to list the results or to export them into a CSV, see fig. 8.

Figure 8: Data Visualization: presenting the values collected for the Lunch Location variable

3.2. BackEnd

To implement this application, several technological options had to be made. The first was the choice between Standalone application VS web application. The fact that a web application is accessible from any computer with an internet connection, and also the fact of not requiring facilities (beyond the initial deployment where the application will be running) were the main reasons that led us to opt for such approach.
3.2.1 System Architecture

The language chosen for this implementation was ASP.Net (C#)\(^1\) since that it is a language that we were already familiar with and knew would perfectly suit our purposes. To implement some features, we also appealed to JavaScript/jQuery. Regarding the layout, we used CSS Bootstrap frameworks\(^2\) and UIKit\(^3\).

For data storage, we had discussed several options, such as MySQL, Microsoft SQL Server and SQLite \(^4\). Again, in order to turn the initial setup as easy as possible, we ended up choosing to use a SQLite database as this does not require any installation or configuration. In fig. 9 we present the schema used.

![Figure 9: Database Model of our application](image)

**Implementation Layers**

This application was implemented following a layered approach. We have three different layers (see fig. 10):

- Presentation
- Business
- Data

The **presentation layer** is the one responsible for the information delivery and formatting to be sent to the business layer for further processing, and also for presenting information to the user.

The **business layer** is the layer where all the information processing happens. This is where are executed all the rules and calculations needed for the proper functioning of the application.

Finally, the **data layer** provides all necessary methods for the interactions performed on the database by the business layer.

Each variable that is created corresponds to an entry in Vars table, where each of these inputs keeps various types of information such as: an unique id, name, data source, frequency and possible values. Regarding our ER model, we could have added a third table to keep the possible values of each variable. For the sake of simplicity, we ended up by storing all the allowed values together, using a special character to separate them. This decision is completely transparent to the system usage, since it is the data layer who has the responsibility to deal with this detail.

Concerning the source of the data, this version of the application supports the following:

- User: all data is entered manually by the user;
- Google Spreadsheets: data is collected automatically from a Google Spreadsheet via the official API;
- Fitbit: data is collected automatically by the official Fitbit API.

The implementation of this application it was all done in order to achieve a modular system for the introduction of new external data sources. There is a main class, called Plugin, from which data access classes implementations extends, thus it makes available a set of methods that the system will later use without having to worry about to which external sources is interacting with.

From the other hand, records live in another table (named Records) and each entry stores information, such as the record timestamp, its value, a link (foreign key) to the corresponding variable and a field indicating whether this record is pending or not. Record pending will be addressed in Data Quality section.

3.2.2 Manual Data Collection

To manually enter a record is the most simple way of feeding data to the system:

- We choose the variable to which we want to add the record, the record timestamp and its value.
• In case the chosen variable have a list of possible values associated, they are presented in the form of a suggestion (listed in a dropdown), so that the user can make a choice without errors.

In a more technical perspective, this part of the application was simpler to implement because it only requires the user to select the variable, the date indication and its value. If there is no conflicts, the data goes straight to the database.

3.2.3 Automatic Data Collection

The automatic data collection requires specific background jobs:

Variable creation
When we create a variable in which the data source is external, a few more settings are asked for the user to insert, as has been shown in the LogMe User section and as we can see on fig. 11. It may seem that there are a lot of fields to fill, although it is done only the first time and it will ease all the future processes and save time, making it quicker. Our application provides access to two external data sources, Fitbit and Google spreadsheets.

Fitbit provides an API that allows us to collect any information that is generated/collected by the device. For the sake of this application we have implemented access to three values: Steps, Water and Sleep; respectively, number of steps in a given day, quantity of water ingested, and hours of sleep. When we perform a request to the Fitbit API, we can choose in which format we want the response, it can be Json or XML. We have chosen XML because of the great .NET support to deal with information formats and also because of our familiarity with it. Since there are no official libraries, we have implemented these methods from scratch, responsible for making the requests and parsing the XML responses.

Google spreadsheets, contrary to Fitbit, offers an official library that simplifies access to data through a set of methods and, by using it, we have access to every spreadsheet stored in our Google drive. In this case, we can access a list of all the existing spreadsheets and the sheets within each one. When in a sheet, we can navigate through their cells and retrieve data.

Once the variable is set, we proceed to the initial data collection. In this stage, system will make use of the access’ plugin to link to the outside source. Using the settings provided by the user, it will make a collection of all data in the fields that the user selected, in the time gap inserted, and it will update the variable indicating the last value consumed, so it knows where to start next time. If the variable has been configured with a list of possible values, a list of conflicts will be presented to the user (if there is any conflict), between the imported values and the allowable values. Here will be given to the user several options to solve these conflicts (see fig. 6). Concluding this step, the process of variable creation is completed.

Background job
The process of automatic data collection occurs each time users load the homepage of our application. Regarding the periodicity of the execution of this operation, and in order to obtain a better performance, we pondered whether this should be performed within a fixed time interval, instead of being executed every time you load the application. However, we opted for the second approach, since this only has impact on performance when there is new data to be imported, which does not happen most of the time.

By the end of every automatic data collection operation, the system will update a field on the variable database record, which holds information about the location of the last value consumed. In the case of a Fitbit variable this will be a date. In a Google spreadsheet it is the row number. With this, the system knows where to start collecting more data without user intervention.

Figure 11: Setting up a new variable from a Google spreadsheet

Authentication with External Data Sources
When we talk about systems that deal with personal information and allow access to the data through an API, these accesses are not public and we always have to associate an authentication mechanism. For the two implemented systems we used the authentication mechanism through the OAuth protocol.

The alternative to OAuth system would be to ask a user to introduce their access data somewhere in
our application, which is not always seen as something favorably, as there have been cases of dishonest developers who keep that data and use it for other purposes. That's why we decided on using OAuth. In OAuth system, the authentication is done in two steps:

1. The user is redirected to a login page of the entity in question;
2. After a successful login, it goes back to our application. When user returns to the application, brings a token, which is provided to the application. This token serves to sign our requests to the respective APIs. At any given time, the user is able to revoke access, causing the token to no longer be valid, invalidating future orders.

3.2.4 Data Quality

A major goal of this thesis is to enable the user to build a single data source, where these are normalized and consistent. In order to achieve this, we have taken great attention, and implemented various mechanisms, that allow the user to achieve a very high level of quality data. Next we will talk about these mechanisms and where do they come into play.

**Manual Insertion:** when manually entering a value that is not part of the list of possible values, by submitting that registration, system detects that the value is not allowed and let the user to make a decision about what to do next, giving two options (See Fig.5):

1. Add the value to the list of possible values and proceed with registration. Thus, this new value starts to be one of the allowed values and next time it is inserted, it is already part of the suggestions made for this variable.
2. Replace the value entered by one of the existing allowed values.

**Automatic data collection:** when the system connects to an external data source and collects new data, it will check for possible conflicts between these new data collected and the allowed values. If there are any conflicts, a table will be presented with all "conflicting" values, where you can choose the action to take for each case. Like in the manual insertion, here will also be presented the chance to add the new values to the list of possible values or to change the value with one presented in the list displayed. Additionally two other options are available (see fig. 6):

1. to edit the value and add it to the set of allowed values;
2. to ignore all occurrences of that value, not importing any of those values.

Only at the conclusion of these steps, data will be loaded into the database.

This process of data normalization is one of the key aspects of this application because it is through it that we can guarantee to users that they will obtain a set of consistent and correct data.

After each data entry, and in the existence of conflicts, the user is obligated to choose one of the options already mentioned above. It is not possible to make such an operation later. This decision was considered and discussed, and we chose this path because if it were possible to leave it to later, users would eventually forget about it or simply ignore it, accumulating, that way, a large number of data to be processed later on, which may lead to stop using the application because of the increased amount of time needed to deal with it.

3.2.5 Reminders

In our application home page, after the execution of automatic data collection, system will cycle through all the variables that have a frequency and will check which of these variables have missing values for the current day. Then, it presents the user a list of reminders to inform the user which records have missing values (see. fig. 7). The fact that we only check missing records for the current day is deliberate, since this operation requires some processing. If we were to verify the completeness of the data since the "beginning of time" on the homepage, this would take more time than the acceptable (and, as we referred earlier, users are impatient). Therefore, and since we consider that there should be an option to make this more exhaustive verification, a section was created off the home page, where the user can observe all data that is missing.

Another important aspect is that if the user has a huge amount of values missing for the current day, the system will only present the first five. Again, this was implemented deliberately in order to not affect the initial page loading time.

4. Results

To evaluate the usability of our system, a set of testers were given a prototype to try and comment on its use. It consisted in a session, composed by an initial form, a set of exercises and a final questionnaire.
4.1. Prototype

Users were given a set of tasks that they had to perform, using all the different approaches. By observing users, measuring the task completion rates and time used, results were processed and analyzed. After the tests, we used questionnaires to evaluate the results. In the tests’ guide, questionnaires, data collected from them and their feedback were analyzed and studied, and the conclusions will be presented on the next section.

4.2. Evaluation

The evaluation was divided in 3 parts:

- Initial form where we traced testers’ profile;
- Exercises, where were presented a set of actions to perform on a prototype that will cover all the possible solutions to a certain problem;
- Final questionnaire, aimed at usability and qualitative assessment of the system.

4.3. Experimental Results

To open the tests’ session, we have made an introduction to lifelogging, because not all users were familiarized with that term. Briefly, we described lifelogging as the recording, storage and distribution of everyday situations, with the support of technologies.

4.3.1 Analysis of Initial Form Results

After that clarification and introduction where we referred the application purpose, we started by distributing our initial form to users.

Tests were executed by 16 people, 7 male and 9 female, whom demonstrated interest in using this kind of tools, with some experience on using technological gadgets (such as smart phones, tablets, computers). On table 1 we can see the ages summarized.

<table>
<thead>
<tr>
<th>Age</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18-30</td>
<td>44%</td>
</tr>
<tr>
<td>31-40</td>
<td>31%</td>
</tr>
<tr>
<td>41-50</td>
<td>25%</td>
</tr>
</tbody>
</table>

Table 1: Users’ ages summarized

4.3.2 Analysis of Tests’ Results

When the initial form was filled by every user, we delivered the tests’ guide to the users at the same time and the counters were set to zero for each user in each exercise. Given the users’ characteristics (including the high familiarity with technology), the error rates during the tests’ execution was considered zero. We also associate these results to the fact that this is a system with a low degree of difficulty.

The first set of exercises were easily executed by every user, although the first and third exercises have low standard deviations which, in our opinion, would be expected since the course of the task is fixed. However the second exercise is different, as the user needs to handle the conflict resolution, and that implies a choice that differs from user to user which, in turn, may have different durations.

The results of the second set of exercises, which goal is to instruct about the insertion of values, we asked users to insert values in different situations. These exercises all have a low standard deviation, such result was also expected, since all exercises, except for the fourth, had a fixed course. The exercise 4, although it has two possible courses, both hypotheses would take approximately the same amount of time.

The third set of exercises has the purpose to show users how they can export data, in the first exercise the user inserts a range of time and a variable to list all the records associated; in the second exercise it is asked to export the previous list to CSV. The results obtained from these two exercises are exactly what were expected. The small variation that is presented on exercise 3.1 relates only to the time taken by users to insert the start and end dates. The duration of the exercise 3.2 is the same for all cases, as it is boiled down to pressing a button.

4.3.3 Analysis of Final Questionnaire’s Results

After executing the proposed exercises, we asked our users to answer a final questionnaire. This questionnaire is divided in 2 parts: first part where we used the System Usability Scale (SUS) and the second part where we have specific questions related to the application.

SUS scores have a range of 0 to 100, and in order to evaluate what is a good result, we looked into several studies, as the opinions can slightly vary. In "An Empirical Evaluation of the System Usability Scale" [1], the author has done 2324 assessments with an average on 70.14 based on all the assessments. The author claims that good systems get between 70-80 points, and exceptional systems get 90 or more. In "Measuring Usability with the System Usability Scale (SUS)" [7], from a study with over 5000 users across 500 different evaluations, the average result was 68. Based on both studies, we can conclude that a score above 70 indicates that it is a good system.

Our overall SUS Score is 74.1, which means that we have a good system and we are on the right track! Even though a SUS score can range from 0 to 100, it isn’t a percentage.
4.4. Experimental Results’ Discussion
Generally, the results of our experiments were positive, to the extent that most people considered that our application is helpful and the exercises were adequate. Also, having a score of 74 on SUS is very good and exciting, although we know that there is a lot more to improve. The overall assessment was also positive, as most of the users see themselves using this application in the future.

Regarding our achievements, we can say that we have satisfied requisite R1 (Reflective Learning supportive), as our application provides adequate means to address the main barriers identified for each one of the states in the Five-Stage-Based model (Preparation, Collection, Integration, Reflection and Action) through our data collection processes, data quality and normalization, reminder’s system for users and data exporting.

As for R2 (Data Quality), we used auto completion/suggestions and predefined values, also data quality is directly related to normalization, which we also executed.

Requisite R3 (Balanced Interface Complexity), according as give users the power to select the amount of visible information.

Concerning requisite R4 (Easy to use and Non time consuming), by providing a clean and easy interface, which turns out to be non time consuming.

Finally, requisite R5 (Keep motivation), is satisfied through our active notification system to the user, in order to initiate the reflection process and help to keep the user motivated, along with the clean, easy and non time consuming interface.

We can conclude that we accomplished mostly of our goals, as we have created a system that allows both manual and automatic collection, thus facilitating the collection and integration process.

5. Conclusions
In this work we proposed a solution to perform a mash-up of different lifelogging data sources, building a system which properly supports reflective learning, data quality, possibility to users to select different data sources, and it will be an easy to use application and non time consuming, which is an important feature nowadays. Filling these requirements we created a system that solves (or greatly minimizes) the problems indicated above.

At the end of this work we are pleased with the results and believe that we developed something useful. As already mentioned, we currently have a lot of personal information collection systems, where each one of them works alone, not allowing its users to know how certain factors can influence others.

Regarding the future work, one of the most obvious features is data visualization. Since we are now able to aggregate data from several sources, and together with a high level of data quality, it will be very interesting and useful to present results based on the correlation between all the variables collected. It is also important to take into account what types of information visualization best suit each scenario.

References