The Journal of Systems and Software 104 (2015): 82–89

DOI: https://doi.org/10.1016/j.jss.2015.02.055

Sentiment Analysis in monitoring software development processes: an exploratory case study on GitHub's project issues

Francisco Jurado*, Pilar Rodriguez
Universidad Autónoma de Madrid, Francisco Tomás y Valiente 11, Madrid, Spain
Email addresses: Francisco.Jurado@uam.es, Pilar.Rodriguez@uam.es
* Corresponding author Tel.: +34 91 497 7527; Fax: +34 91 497 2235

Abstract

Software process models, which allow us to develop software products, can be improved by using the corresponding quality model. However, current tendencies in the application of Global Software Engineering and Global Software Development, which forces geographically dispersed teams to collaborate, make the usual monitoring techniques obsolete. This situation has lead to looking for new methods that can help in the decision making process, such as the case of the Social Network Analysis field.

In this article we propose the introduction of Sentiment Analysis techniques in order to identify and monitor the underlying sentiments in the text written by developers in issues and tickets. Therefore, in order to check its viability we conducted an exploratory case study analysing polarity and emotional clues in development issues from nine well-known projects that are freely available. Results show that although both polarity and emotional analysis are applicable, the emotional analysis looks to be more suitable to this kind of corpus. The developers leave underlying sentiments in the text, and that information could be monitored as any other feature in the development process.

Keywords: Software Development Process; Github; Sentiment Analysis.

1. Introduction and motivation

Developing software products involves the application of Software Development Process (SDP) models (Wohlin et al., 2000). These models describe the activities to perform as well as the involved roles and their responsibilities in the life cycle of software development. The purpose of classical process improvement models is to ensure the quality of the SDP and the resulting products, as well as continuously improve the quality of development processes (Basili, 1985; (Deming, 1986).

In this context, it is possible to apply quality models, like the ISO-IEC2011 (ISO/IEC, 2011) in order to guarantee a Quality Model in Use and a Product Quality Model, and process evaluation and improvement models, like the Maturity Capability Model Integration (CMMI) (Chrissis et al., 2011; Forrester et al., 2011) or the ISO/IEC15504 (ISO/IEC, 2004). Usually, their application implies the involvement of the different people that make up the different development teams; their involvement in quality for each process is essential and must also ensure the continuous improvement of processes (Chrissis et al., 2011; Forrester et al., 2011; ISO/IEC, 2004).

Somehow, these models include ways to measure the end-users' degree of satisfaction and usability studies for a developed system or product. However, there is no way to measure how the

people involved in the development processes feel, or how to identify whether it has any direct relationship with the development process or the final product.

In addition, the role of the team members has grown with the current business trends that lead to Global Software Engineering (GSE) and Global Software Development (GSD). In these trends, projects are evolved globally by geographically dispersed development teams, some even from different time zones (Tamburri et al., 2012). In those kinds of projects the usual monitoring techniques are obsolete.

This situation has led to the introduction of the use of techniques from the Social Network Analysis (SNA) field, which helps in the decision making process (Manteli et al., 2012). However, *"more research is needed in the field of global software development by taking a social network perspective"* (Manteli et al., 2012). In addition, (Manteli et al., 2012) also states that SNA can help to *"examine the coordination, cooperation and communication structures of the remote and co-located teams, identify potential weak points and support the organization and the decision-making processes of the distributed activities"*.

So far, the basis of the above-mentioned models is monitoring measurable features on products, processes, and social network among developers. However, there are other factors that can have influence but are difficult to estimate. That is the case of the role of emotions in professional work and how they can affect productivity, task quality, job satisfaction, etc. (Fisher and Ashkanasy, 2000; Seo et al., 2010; De Choudhury and Counts, 2013). However, in spite of its importance, there are few studies related to the influence of emotions in software engineering (Wrobel, 2013). Moreover, as far as we know, there are no metrics nor indicators that allow estimating the satisfaction degree of the development team on a specific feature, the use of a particular technology, the direction of a step in the development process, etc., and all of these factors can affect the process and product quality.

Thus, in this article we purpose the use of the non-intrusive technique known as Sentiment Analysis or Opinion Mining (Pang and Lee, 2008; Liu, 2010) for monitoring emotional features in the development process. Thus, in order to check the viability of our proposal, we have analysed the sentiment embodied by developers on issues and tickets of several software development projects. To do this, we gathered the issues of some well-known projects, taking into account different environments and programming languages. Then we computed the underlying sentiments in the text the developers had written, and then analysed the obtained data.

The rest of the paper is structured as follows. First, we start with a brief introduction to what Sentiment Analysis is, its techniques and key points. Second, the exploratory case study we performed and the data we obtained is presented. Third, taking into account the analysis results, improvements are discussed. Finally, concluding remarks are exposed.

2. Related work

2.1. Emotions in the software development process

As recent studies reveal, programmers' emotions have an impact on their performance in the development process. Thus, in (Wrobel, 2013), the data collected by an online survey of developers shows that emotions like *anger* have a positive effect on the developers' productivity, and that other negative emotions, like *frustration* and *disgust,* must be taken into account from the point of view of the risk.

In order to avoid subjective bias in answering the surveys when dealing with sentiments, many other sources can be taken into account, like the analysis of biometric inputs (Chandler and Cornes, 2012), keyboard interactions (Vizer et al., 2009), audio and video processing (Zeng et al., 2009), written text analysis (Strapparava and Mihalcea, 2008; Binali et al., 2010), etc.

In the field of Software Engineering, one of the few works that we can find are performed by Guzman (2013) and Guzman and Bruegge (2013). They propose to improve emotional awareness in software development teams by processing textual collaboration artefacts, such as emails, wikis, commit messages in software repositories, bug reports, etc. To do so, they extract the topics from the text and assign them a polarity score (positive or negative emotion).

In this same regard, (Garcia et al., 2013) analysed the polarity in contributions written by the developers in the Gentoo project during a specific time period. Their analysis revealed a correlation between negative emotions and the changes in the software community organization, as well as the subsequent activity performance.

For their part, (Kolakowska et al., 2013) consider the use of multimodal inputs for emotions recognition in usability testing and development process improvement. However, the preliminary research they developed reveals that some issues must be solved when working with biometric sensors and processing emotions from video. In addition, in spite that they do not limit their approach to biometric and video sources but instead also propose the use of non-invasive sources such as text-based, these authors do not mention the relevance they obtained in their preliminary research from these last kind of sources.

Nevertheless, further studies on what kinds of techniques are more suitable for the context of software development must be performed. In particular, we will centre this article in the study of the suitability of Sentiment Analysis techniques in written text.

2.2. Brief introduction to Sentiment Analysis

Also known as Opinion Mining, Sentiment Analysis can be defined as the *"computational treatment of opinion, sentiment, and subjectivity in text"* (Pang and Lee, 2008; Liu, 2010). It has been applied to many contexts, like reviewing customers' products and services, monitoring reputations in social networks, tracking people's feelings about politicians, promoting marketing campaigns, etc. (Feldman, 2013). In this regard, we mention the study performed at HP Labs by (Asur and Huberman, 2010) about automatic emotion extraction from text in tweets in order to estimate the social impact of movies.

Moreover, it is not only suitable for analysing massive messages, but also for private use. Thus, in some examples we can find the Emotional Marketing Value Headline Analyser (AMI, 2009), which analyses the proposed headline typed by users in order to assign it an Emotional Marketing Value (EMV) score; the work performed by (Bollen et al., 2011) to analyse if social mood states can affect collective decision-making; AngryEmail (Carro et al., 2012), which is used to avoid directly sending an e-mail if a high level of *"anger"* is detected by keeping the message unsent and notifying the user for avoiding undesirable situations; and SentiBuk (Ortigosa et al., 2014), which analyses the polarity in Facebook users' personal messages so they can see the evolution of their sentiments over time.

Sentiment Analysis has even been successfully applied in eLearning contexts, such as the works performed by (Bueno et al., 2011), where chat conversations are analysed using an emotion dictionary in order to provide the emotion evolution along a period of time, and (Cobos et al., 2014), where correlations between student marks and emotion traces have been found in student writings assignments.

There are several techniques and approaches to compute Sentiment Analysis from text (Pang et al., 2002; Pang and Lee, 2008; Liu, 2010; Liu, 2012), among which are naive Bayes, maximum entropy, Support Vector Machines (SVM), lexicon-based, etc. Nevertheless, the common point in several is a central piece known as lexicon (Feldman, 2013). Fortunately, there are several lexicons available (Esuli and Sebastiani, 2006; Baccianella et al., 2010; Strapparava and Valitutti, 2004; Strapparava and Mihalcea, 2007; Strapparava and Mihalcea, 2008; Bradley and Lang, 1999), as

well as several tools to perform the language processing, among which and highlighted are NLTL (Bird et al., 2009), LingPipe (Carpenter and Baldwin, 2011), and FreeLing (Padró and Stanilovsky, 2012).

The next sections will expose the technique, tools, and other elements we have used to conduct an exploratory case study in the context of software development, as well as the results obtained.

3. Exploratory case study

In order to perform a first study to check the viability of measuring developers' sentiment through their written text, we analysed the sentiment embodied by developers on issues and tickets of several software development projects. To do so, we collected all of the issues of some large well-known projects from GitHub (https://github.com/), taking into account different environments and programming languages. Then, we performed Sentiment Analysis on each issue and analysed the obtained results. In this section we expose the whole process in depth.

3.1. Sample collection

Based on the Git control version system, GitHub is a popular code repository site (Dabbish et al., 2012) for a lot of well-known and active projects. Each project has its own repository and gives access to the history of the source code, commits, issues, and other related data. In addition, for our purpose one of GitHub's best features is that all of the projects it hosts are publicly accessible. Moreover, in order to get access to all the available information and to gather the projects' issues, there a REST (Representational state transfer) API (GitHub, 2015) is available, which returns the desired information in JSON (JavaScript Object Notation) format for each parameterized HTTP request.

Thus, to get all of the issues for a specific repository, it is necessary to perform HTTP GET requests to the api.github.com server following the syntax */repos/:owner/:repo/issues?page=:page&state=:state*, where *:owner* is the owner of the repository, *:repo* is the repository from which we want to get the issues, *:page* is the issues' page number (issues can be paginated up to one hundred issues per page), and *:state* represents the status of the issues we want to list, which can be opened, closed, or all. For instance, the HTTP GET request */repos/scala/scala/issues?page=1&state="open"* will obtain the *first page* containing the *open* issues from the *scala* repository owning to *scala*. The information this request sends back is a list of JSON objects. Each JSON object is a list of keys with their corresponding value, which can be, in turn, a string-formatted value (independent of its data type), a JSON object, or a list of these.

These JSON objects contain all of the information related to each issue, such as the title, the text body, the author, the creation and modification dates, the number of comments received, etc. Thus, our corpus is composed by a set of JSON objects that can easily be manipulated in most of programming languages as key-value dictionaries.

The full list of examined projects, as well as the number of gathered issues, is listed in table 1. As we can see, the whole corpus contains 10,829 entries from nine different projects.

Project name	URL	Gathered issues
Homebrew	http://brew.sh	1345
Bottle	http://bottlepy.org	568
Facebook tornado	http://www.tornadoweb.org	978
IPython	http://ipython.org	1651
Joomla-cms	http://www.joomla.org	1438
Pydata Pandas	http://pandas.pydata.org	1761
Ruby on Rails	http://rubyonrails.org	1649
Raspberrypi Linux	http://www.raspberrypi.org	511
Scala	http://www.scala-lang.org	928
Total		**10829**

Table 1: Gathered issues per project

3.2. Lexicon selection

To perform Sentiment Analysis in text - that is, the title and body of the issues - we have chosen a lexicon-based approach (Taboada et al., 2011) due to its simplicity and accuracy in unbounded domains (Ortigosa et al., 2014). This kind of approach tries to identify sentiments by looking for words related to emotions contained in a lexicon. That is, they provide matches between a specific word and its related sentiment.

There is not a commonly agreed-upon classification regarding emotions. Most of the works in Sentiment Analysis are focused on following one of the next two kinds of approaches (Pang and Lee, 2008; Liu, 2012): a) identifying the sentiment polarity in text as *positive*, *negative,* or *neutral*; and b) identifying the basic emotions proposed by Ekman (Ekman and Davidson, 1995), or a subset of them, namely: *anger*, *disgust*, *fear*, *joy*, *sadness,* and *surprise*.

We can find several lists of words with sentiment information in order to follow a lexicon-based approach. One of the most widely used is that built by (Bradley and Lang, 1999) and known as Affective Norms for English Words (ANEW). It rates the words on the dimensions of valence (from pleasant to unpleasant), arousal (from calm to excited), and dominance (from control to out of control). Therefore, we can take the valence dimension and classify the pleasant terms as positive and the unpleasant terms as negative to perform a polarity analysis. However, the main drawback of this list is that it only includes the 1030 most frequent English words gathered in 1999, but not an up-to-date list with all of the words with affective valances for a specific domain.

Whilst dealing with polarity in text, first we must identify if the text is objective or subjective. In the second case we can analyse if the polarity is positive or negative. Under this approach, there exist domain independent subjectivity lexicons specifically designed to perform polarity analysis. Thus, Sentiwordnet (Baccianella et al., 2010) provides numerical scores for positivity, negativity, and objectivity to synsets (synonym sets) from the WordNet database. This has been used in works like (Denecke, 2008; Ohana and Tierney, 2009). Another is the lexicon developed by (Wilson et al., 2005b), which provides positive and negative tags for words, as well as their subjectivity degree. This is the base for tools like OpinionFinder (Wilson et al., 2005a), software like the Sentiment R Package (Jurka, 2012), and research works like (Bollen et al., 2011).

Opposite to the mentioned domain independent approaches, to perform polarity analysis specifically for microblogging (tweets in particular), we find the lexicon included as part of SentiStrength (Thelwall et al., 2010), which has been applied in studies like the above-mentioned developed by (Garcia et al., 2013) to analyse the emotions in the Gentoo project.

So far, the mentioned lexicons allow annotation of the polarity of words, but not emotions. Polarity analysis allows us to identify if a subjective text has positive or negative clues (if the writer is pleasant or unpleasant, for or against a topic, etc.), but not the emotion to which they are related. For instance, positive clues are associated with feelings of pleasure and happiness, that is, the joy

emotion. However, if we identify negative clues in a text, these can be related to emotions such as sadness, disgust, fear, or anger. In addition, we can find both positive and negative clues related to surprise emotions (depending on the kind of surprise).

To classify words within the six basic emotions identified by (Ekman and Davidson, 1995), we use the Wordnet-affect lexicon elaborated by (Strapparava and Valitutti, 2004). Freely distributed under Creative Commons License, this lexicon has been included in other widely used tools like the Sentiment R Package, developed by (Jurka, 2012).

In order to provide a quantitative view of the differences among lexicons, table 2 shows the intersection among some of the above-mentioned. We have excluded SentiWordNet because it works with synsets. The diagonal of the table represents the number of different words per lexicon, denoting the differences among them. However, the accuracy of each lexicon not necessarily depends on the number of words. In addition, the number of words that intersect between pairs of lexicons is quite low.

Under this perspective, to select one lexicon or another depends on the domain it will be applied to, the information it provides (scores, labels, etc.), as well as the algorithm used in order to perform the sentiment analysis. Therefore, taking into account all of these issues, we used all of the above-mentioned lexicons in order to analyse which would be the most suitable to our exploratory case of study.

	ANEW	OF	SS	WA	WC
ANEW	1030				
OpenFinder (OF)	451	6457			
SentiStrenght (SS)	93	598	941		
WN-Affect (WA)	114	654	100	1512	
Words in corpus (WC)	315	670	177	180	35202

Table 2: Words' intersection between lexicons

3.3. Preparing the lexicon

Once we identified the lexicons to work with, we noticed that they were too general. Therefore, domain specific filters were introduced. One of the main issues faced was removing the usual compiler/interpreter messages indicating warnings, errors, and exceptions.

Developers usually paste these messages in the text. They include sentences like: *"HTTP Error 400: Bad Request"*, which contains words like "bad" (labelled as sadness) that can affect the corresponding sentiment ratios. Thus, it is desirable not to compute the words of these message in the sentiment ratio computation, since they were not written by the developer. Therefore, we analysed the software developers' expressions and their usual error, warning, and exception messages, and removed all of these words from the lexicon. In addition to this we also removed usual words related to message expressions like *"regards"*, *"kind regards"*, etc.

Examples of all these words include: aggressive, attach, attachment, bad, bash, catch, cheer, close, compact, console, content, expect, flush, get, identify, impress, kind, like, low, move, partial, protect, please, regard, shadow, together, top, and weight.

3.4. Sentiment Analysis algorithm

The algorithm we implemented is shown in figure 1. The algorithm shows how, for each issue in our sample collection, we extracted the words by dividing the text into sentences and tokenizing each sentence into words after deleting the punctuation symbols and numbers. Then, we lower-cased each word and removed all of the stopwords (words without meaning, such as articles, prepositions, etc.) Next, because each word can be derived, we performed a stemming process to

remove the lexemes (prefixes and suffixes) and take just the "stems", which are the minimum part of the word with meaning. After this last step, we looked up what sentiments were related to those stems, created a list of sentiments per stem, and added them to the list of sentiments for the whole issue. Finally, we computed the sentiment of the issue by counting how many times each sentiment appeared in the issue.

```
for-each issue in {issues} do
    {sentences} = extract_sentences(issue)
    for-each sentence in {sentences} do
        remove_punctuation(sentence)
        {words} = tokenize(sentence)
        {words} = lower-case({words})
        {words}={words}−{stopwords}
        {sentiments}= ø
        for-each word in {words}
            stem = stemmize(word)
            if stem in lexicon then
                {sentiments} = {sentiments} ∪ lexicon_stem
    {sentiment_scores_issue} = count({sentiments})
return {sentiment_scores}
```

Figure 1: Algorithm to compute the issues' sentiment per project

As technical notes, for text processing we used NLTK (Bird et al. 2009). For the stemming process we used the SnowBall stemmer (Porter, 2001), which implements the well-known and widely used Porter's algorithm. This is easily applied in several languages just by selecting the corresponding stemming rules specification for the target language. This algorithm is also included in the NLTK software.

3.5. Data analysis

The above algorithm was applied to the whole corpus of gathered issues by using all of the mentioned lexicons in section 3.2. Thus, for each *issue_id* we obtained the number of words identified under the corresponding emotion identified by the WordNet-Affect lexicon, as well as the corresponding polarity analysis obtained by using ANEW and the lexicons included in OpenFinder and SentiStrength tools.

Table 3 shows the Pearson's correlation among the number of words computed for each emotion and the polarity computed by applying each different lexicon. It is interesting to notice that there is a positive correlation between the positive and negative analysis for the polarity obtained with the corresponding lexicons. This fact introduces undesirable ambiguity and affects the possible interpretations that can be done in the polarity.

Therefore, in order to check the correlation significance, we applied a one-tailed *t-test*. Table 4 shows the obtained *p-values*. It is remarkable that the results corroborate that there are high correlation significances for most of the polarity analysis, but correlations between the computed emotions look to have very low significances. According to these results, performing a polarity analysis with the selected lexicons is inappropriate for our corpus, but are suitable for the emotional analysis.

	anger	disgust	joy	surprise	fear	sadness	pos_OF	neg_OF	pos_SS	neg_SS	pos_ANEW	neg_ANEW
anger	1.00	0.01	0.07	0.05	0.02	0.03	0.10	0.11	0.03	0.09	0.07	0.03
disgust	0.01	1.00	0.00	0.00	0.33	0.00	0.00	0.04	0.12	0.00	0.00	0.02
joy	0.07	0.00	1.00	0.06	0.02	0.03	0.29	0.24	0.20	0.15	0.22	0.05
surprise	0.05	0.00	0.06	1.00	0.17	0.01	0.08	0.08	0.03	0.03	0.09	0.05
fear	0.02	0.33	0.02	0.17	1.00	0.01	0.05	0.08	0.07	0.03	0.03	0.04
sadness	0.03	0.00	0.03	0.01	0.01	1.00	0.04	0.08	0.03	0.06	0.12	0.04
pos_OpinionFinder	0.10	0.00	0.29	0.08	0.05	0.04	1.00	0.45	0.20	0.31	0.36	0.08
neg_OpinionFinder	0.11	0.04	0.24	0.08	0.08	0.08	0.45	1.00	0.22	0.77	0.48	0.17
pos_SentiStrenght	0.03	0.12	0.20	0.03	0.07	0.03	0.20	0.22	1.00	0.13	0.13	0.03
neg_SentiStrenght	0.09	0.00	0.15	0.03	0.03	0.06	0.31	0.77	0.13	1.00	0.40	0.17
pos_ANEW	0.07	0.00	0.22	0.09	0.03	0.12	0.36	0.48	0.13	0.40	1.00	0.43
neg_ANEW	0.03	0.02	0.05	0.05	0.04	0.04	0.08	0.17	0.03	0.17	0.43	1.00

Table 3: Correlation analysis.

	anger	disgust	joy	surprise	fear	sadness	pos_OF	neg_OF	pos_SS	neg_SS	pos_ANEW	neg_ANEW
anger	0.00	0.24	0.00	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
disgust	0.24	0.00	0.37	0.44	0.00	0.45	0.32	0.00	0.00	0.43	0.39	0.03
joy	0.00	0.37	0.00	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00
surprise	0.00	0.44	0.00	0.00	0.00	0.21	0.00	0.00	0.00	0.00	0.00	0.00
fear	0.03	0.00	0.02	0.00	0.00	0.22	0.00	0.00	0.00	0.00	0.00	0.00
sadness	0.00	0.45	0.00	0.21	0.22	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pos_OpinionFinder	0.00	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neg_OpinionFinder	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pos_SentiStrenght	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neg_SentiStrenght	0.00	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
pos_ANEW	0.00	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
neg_ANEW	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4: *p-value* for the correlation analysis.

Therefore, once the polarity analysis was discarded, we continued with the emotional analysis. Thus, we calculated the ratio for each sentiment as the number of sentiment clues divided by the total number of meaningful words in the issue. Then we assigned a label for each issue with the highest and most representative sentiment ratio. We will refer to this as the *most common* sentiment for the issue.

The obtained results are summarized in table 5 and graphically in figure 2. In that figure, the highest a bar is the most number of issues for that project that appeared in the corpus. From the point of view of measuring sentiments, we can see how there are a lot of messages that do not reflect any kind of sentiment. We have labelled them as *None*.

It is interesting to observe that all the projects have a great portion of messages in which the *joy* sentiment is identified, followed by the *surprise* sentiment. Then, in lesser measure we can see messages labelled with *anger*, *fear,* and *sadness* sentiments.

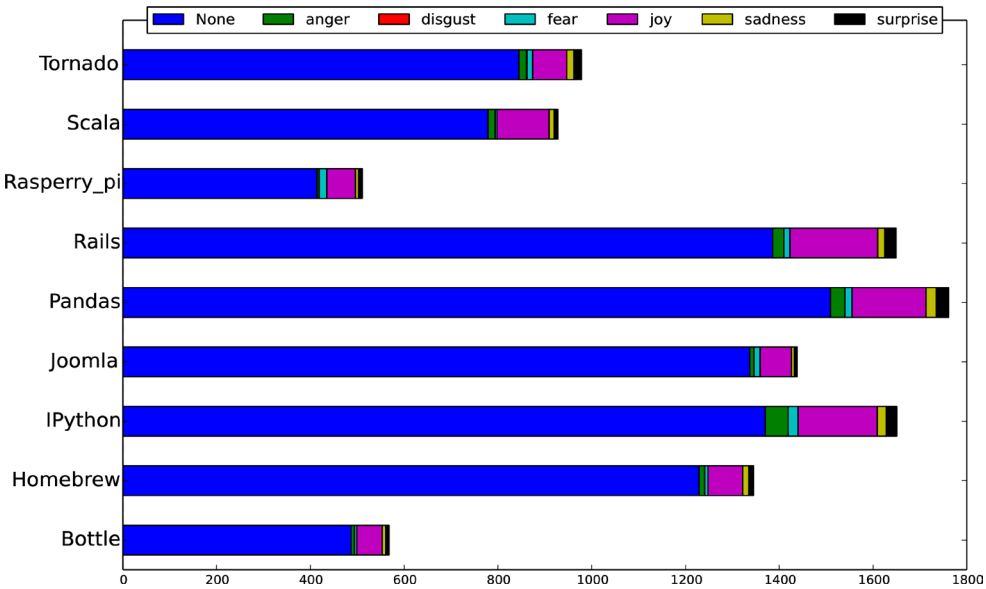

Figure 2: Sentiment in messages per project

	None	Anger	Disgust	Fear	Joy	Sadness	Surprise
Bottle	85.74%	1.23%	0%	0.88%	9.51%	1.41%	1.23%
Homebrew	91.38%	0.89%	0%	0.52%	5.50%	0.97%	0.74%
IPython	82.98%	2.97%	0%	1.27%	10.24%	1.15%	1.39%
Joomla	92.98%	0.63%	0%	0.90%	4.66%	0.42%	0.42%
Pandas	85.69%	1.76%	0%	0.85%	8.97%	1.25%	1.48%
Rails	84.05%	1.46%	0%	0.79%	11.34%	0.91%	1.45%
Rasperry pi	81.02%	0.98%	0%	3.13%	11.94%	1.37%	1.56%
Scala	83.94%	1.62%	0%	0.43%	11.96%	1.18%	0.86%
Tornado	86.40%	1.64%	0.10%	1.23%	7.46%	1.53%	1.64%

Table 5: Sentiment percentage distribution per project

As we can see in figure 2, the results show that there are a lot of issues where the *joy* emotion is highlighted. Since all of these projects are free and open-source, these results make sense because people usually enrol in these kinds of projects of their own free will and just for fun. Nevertheless, as (Garcia et al., 2012) demonstrated, this result could be in part because of the positive emotional bias in languages. However, as (Garcia et al., 2012) also affirms, those words with negative emotional clues carry more information. These are the situations we are interested in monitoring. Then, continuing with the analysis of results, we observe how the *joy* emotion is followed by *anger* and *surprise* in some projects. In contrast, the number of issues with the sentiments *fear*, *sadness,* and *disgust* are fewer in most of the cases.

So, to monitor the sentiments' growth in perspective, we calculate the word ratio per sentiment (by dividing the words found for each specific sentiment by the total number of words in each issue) and compute the cumulated sum for each sentiment ratio. As an example, figure 3 shows the cumulated sum for each specific sentiment in the Pandas project, except *joy*. Every abrupt variation of the sentiments can be easily identified. In that figure, we can see how the sentiments cumulatively grow more or less constantly; there are specific issues where growth is more pronounced, as observed for *fear* around the issues 4000 and 5000.

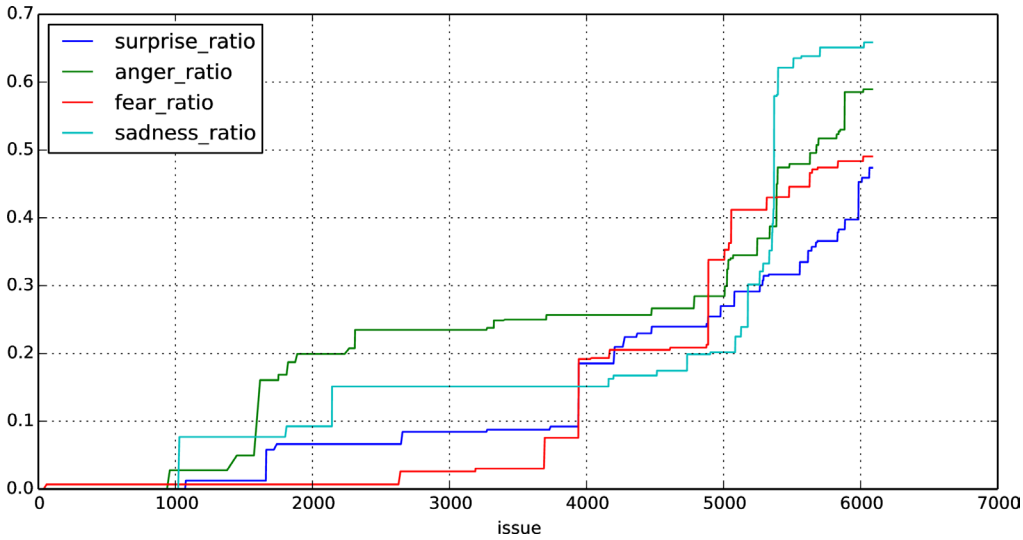

Figure 3: Cumulated sum for each specific sentiment in the Pandas project

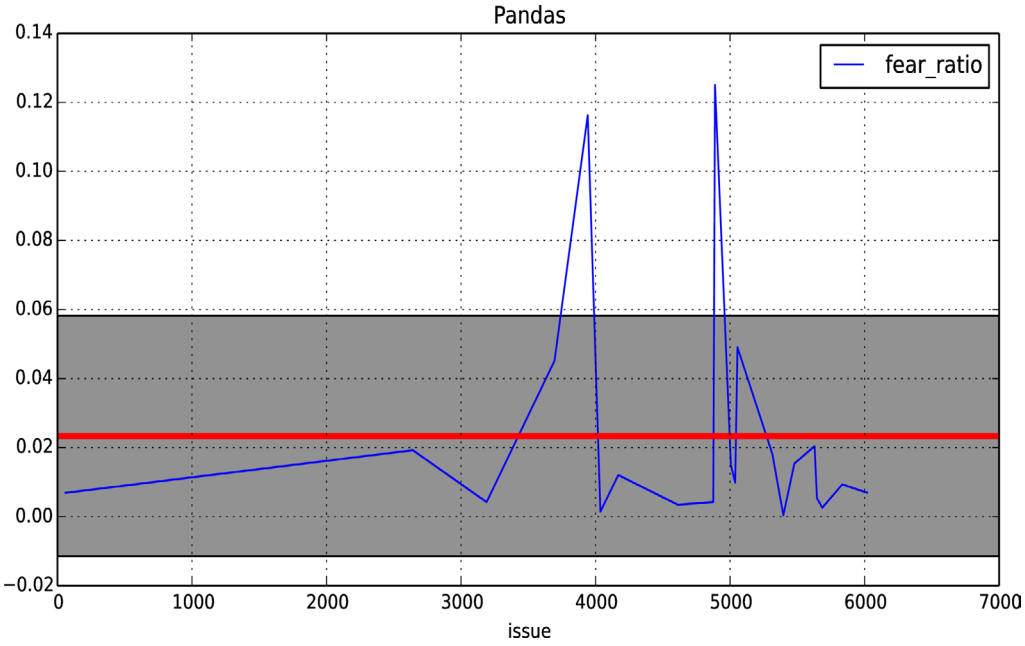

Figure 4: Control chart for fear in Pandas

If we focus on a specific project we are able to monitor those issues where some specific sentiment is highlighted against others, or if a specific issue has a sentiment out of reasonable bounds. Thus, we can visualize these measures as any other metric by using control chart. In these charts, the thick line represents the mean for that sentiment (when it appears), and the shadowed areas are bounded by the standard deviations. Hence, it is easy to find if a specific sentiment is out of reasonable bounds.

Figure 4 shows the control chart for *fear* in the Pandas project. We can see how there are two high levels of fear in issues 3942 and 4888. Thus, for instance, if we look at the first of these two issues we can read the text shown in 5. As we can read in that issue, the team member looks to be a bit displeased with the corresponding feature. Nevertheless, whoever is in charge of analysing the process development can interpret whether or not that issue is really *fear*.

Moreover, in order to monitor sentiment implication within the development process, the project manager or Software Quality Assurance team member in charge of analysing it can correlate it with other metrics, such as productivity, task quality, etc.

Therefore, with the obtained results from analysing the issues of several software development projects, we checked the viability to use Sentiment Analysis or Opinion Mining for monitoring emotional features in the development process.

```
Title: Add key to sorting functions

Body:
Many python functions  (sorting, max/min) accept a key argument,
perhaps they could in pandas too.

*The terrible motivating example was this awful hack from [this
question](http://stackoverflow.com/a/17157110/1240268)....
for which maybe one could do*

    df.sort_index(key=lambda t: literal_eval(t[1:-1]))

*This would still be an awful awful hack, but a slightly less awful
one*.
```

Figure 5: Issue 3942 of the Pandas project

4. Concluding remarks and future works

In this article we have exposed the necessity of introducing new techniques in the monitoring of the software development process to measure how those involved feel. Under this scenario, we propose the introduction of Natural Language Processing methods in order to analyse the underlying sentiments in the text written by developers.

In order to check our assumption we have detailed the exploratory case study we conducted analysing 10829 issues from nine well-known big projects. This was done by exposing how we collected the data and created the corpus, the lexicon we chose, and the algorithm we implemented in order to process the text. Results showed that the developers do leave underlying sentiments in the text, and that information could be used in order to analyse the development process. This also allowed us to check the viability of our proposal for monitoring emotional features in the development process.

For future work, in order to get the specific topic, technology, activity, etc. that brings about a particular sentiment in the developers, we think that the introduction of named-entities recognition techniques (Nadeau and Sekine, 2007) can help. With that we would be able to refine the information gathered in order to detect what has to be changed to improve the development process and product quality. In addition, we are really interested in correlating sentiments with metrics like team productivity, individual and group task quality, etc.

Nevertheless, the proposal is not limited to monitoring the software development process. Together, with the emerging field of Educational Data Mining (EDM), which uses statistical analysis, machine-learning, and data mining (Baker and Yacef, 2009; Romero and Ventura, 2010), we are interested in checking if the use of Sentiment Analysis techniques can contribute to the instructor's task of monitoring the overall learning/teaching process (Romero-Zaldivar et al., 2012) in subjects that involve developing programming labs within work groups, and thus help them act.

5. Acknowledgements

This research has been partially funded by the Ministry of Science and Innovation (*Ministerio de Ciencia e Innovación*) through the projects REF: TIN2011-29542- C02-02 and REF: TIN2013-44586-R, and by the Ministry of Education, Youth and Sports, Community of Madrid (*Consejería de Educación, Juventud y Deporte, Comunidad de Madrid*) through the project S2013/ICE-2715.