A Longitudinal Study of Community-Oriented Open Source Software Development

Kateryna Neulinger[1(✉)], Anna Hannemann[1], Ralf Klamma[1], and Matthias Jarke[1,2]

[1] Advanced Community Information Systems (ACIS) Group, RWTH Aachen University, Ahornstr. 55, 52056 Aachen, Germany
{neulinger,hannemann,klamma}@dbis.rwth-aachen.de
[2] Fraunhofer FIT, Birlinghoven Castle, Sankt Augustin, Germany
jarke@dbis.rwth-aachen.de
http://dbis.rwth-aachen.de

Abstract. End-users are often argued to be the source of innovation in Open Source Software (OSS). However, most of the existing empirical studies about OSS projects have been restricted to developer subcommunities only. In this paper, we address the question, if and under which conditions the requirements and ideas from end-users indeed influence the development processes in OSS. We present an approach for automated requirements elicitation process discovery in OSS communities. The empirical basis are three large-scale interdisciplinary OSS projects in bioinformatics, focusing on communication in the mailing lists and source code histories over ten years. Our study results in preliminary guidelines for the organization of community-oriented software development.

Keywords: Requirements engineering · End-user development · Open source software

1 Introduction

Recent communication infrastructures like Web 2.0 open up new opportunities for requirements engineering. Experts as well amateurs can easily contribute their knowledge to requirements engineering processes. By collecting external ideas, companies get access to a worldwide spread knowledge. Therefore, many companies already provide environments for community building of end-users (e.g. XEROX[1], SAP[2]). We call such software development concepts community-oriented. OSS projects represent a successful example of community-oriented development. OSS communities usually exhibit hierarchical structures with a core layer (core communities) supported by peripheral layers (peripheral communities) [34,37]. Histories of OSS projects can be used as a source to study

[1] http://open.xerox.com/.
[2] http://scn.sap.com/community/coil.

S. Nurcan et al. (Eds.): CAiSE 2016, LNCS 9694, pp. 509–523, 2016.
DOI: 10.1007/978-3-319-39696-5_31

a possible organization of end-user community integration in the requirements elicitation process. Community needs are posed through informal means and negotiated within community discourses. This kind of requirements elicitation (RE) can then be the starting point for a requirements engineering process in which these needs are formulated as requirements, and then specified and implemented in a current or later release of the OSS system. Despite a great number of OSS studies [12] and a novel research in "just-in-time" requirements engineering [2], an approach for automated requirements elicitation within OSS is seldom if ever addressed [8,35].

In this paper, we present a methodology for RE process discovery that incorporates a combination of knowledge mining techniques for community-generated content. In order to address the end user roles, we follow the observation of many papers [34,37] that OSS communities often expose a separation into core developers and a periphery. Our mining techniques are specifically geared to understand this periphery with their structure and contributions as well as the impact and evolution of these aspects over the history of an OSS community. Specifically, we pursue three research questions:

RQ1 Is there a difference of requirements contribution to the OSS community from core and peripheral members?

RQ2 To what extent do requirements correlate with general development activity?

RQ3 Does the level of participation in RE influence the level of satisfaction of peripheral project members?

Given the great variance of communities, and the early stage of this kind of research, we decided to choose for our empirical context of our technology initially three OSS communities that have at least some common properties, so that any differences we could identify can be expected not to be totally based on differences e.g. in domain, user competencies, and the like. Moreover, we wanted to conduct a long-term study over many years, and this obviously implied a certain bias towards relatively successful communities that even had such longevity. As a consequence, we focus this paper on three long-lived communities in the bioinformatics sector, for which we could mine rich, if heterogeneous data for a longitudinal study over eleven years. The rest of the paper is organized as follows. Section 2 gives an overview of existing concepts for end-user integration in the RE process and the organization of RE process in OSS. Section 3 presents a methodology for the RE process discovery in OSS. The methodology is later applied to the data shown in Sect. 4. The results of our study are described in Sect. 5. Section 6 concludes the paper with advices for the organization of community-oriented software development.

2 Related Work

Requirements elicitation (RE) process has evolved from a generally technical to an end-user oriented process [1]. In order to facilitate the requirements negotiation with end-users, different approaches are proposed: mobile technologies [30],

scenario-based RE process [33], group support systems [3]. Additionally, conceptual paradigms for end-user integration vary. Whilst von Hippel [13] suggests to identify the lead users (experts among product users) and collect their needs, Chesbrough [4] proposes to collect innovative ideas from the masses. Maiden et al. [21] advise to apply creative problem solving for innovative idea finding. Yet, the question remains - which tools and paradigms are robust to overcome challenges of community-oriented RE process?

In the following, different analysis methods relevant for the RE process exploration in OSS projects are presented.

2.1 Community Clustering

Over the past decade OSS projects have been investigated in numerous research studies [12,29]. OSS communities represent multi-layer hierarchical structures [6,7,38]. Despite various layer definitions proposed by OSS researchers, we can generalize that the core of the OSS community is a small group of developers, surrounded by a much bigger periphery. The requirements in OSS projects are traditionally based on the personal experience and the knowledge of the core developers. However, questions and suggestions from peripheral users trigger significant modifications in the OSS product [14,25].

In order to investigate the role of requirements coming from core and peripheral communities, they have to be separated. In [7] three different approaches to the identification of the core in OSS communities are compared, based on (1) information on the project Website or other resources; (2) the contribution frequencies; (3) the hierarchical clustering of the social characteristics. In case of the first method, the data on the project Website might be not up-to-date. As for the second method, the contribution log of a project does not always reflect a complete picture of personal efforts. In many projects, only a very small group of developers have the right to commit changes. Therefore, a committer is not necessary the author of the code she/he commits. Finally, the hierarchical clustering makes use of the expected properties of the core to be more dense and cohesive than the periphery. Especially in the context of non-predefined social structures, hierarchical clustering of OSS communities is the most suitable method.

2.2 Requirements Detection

The whole OSS project management takes place in publicly available open access infrastructures. Although the composition of the OSS project infrastructure varies significantly from project to project, most projects include at least mailing lists, a project page and/or wikipage, and a code repository. Requirements in OSS are continuously emerging within communication and development processes 'just-in-time' [8]. Requirements in the form of ideas, complaints, post-hoc description and others are spread among the artifacts created by the OSS community members within the project infrastructures [28].

OSS communities and projects are evolving structures [38]. Thus, OSS processes undergo continuous change and need to be approached as dynamic systems. To perform dynamic analyses of OSS repositories, some researchers divide data in periods of fixed size [26], while others use time points of releases as a cutting criterion [36]. Considering the mechanics of OSS projects, their rhythms and iterations, a continuous cycle of design-analysis-development can be identified. At some point in time during the development process, a current branch is frozen for the next release. From that time on, only bug fixes are allowed. After the code is released, only hotfixes - small code updates which address specific problems in the last release - are possible. Hence, a period (t_j, t_{j+1}) between two releases j and $j+1$ can be considered as a logical step for the dynamic analysis. This approach is consistent with the metrics and laws of software evolution [19].

2.3 Sentiment Analysis

The integration of end-users in the development process aims not only to get the access to the domain knowledge, but also to better the end-users attitude towards the end product. Thus, by measuring the mood within the community, we estimate the level of satisfaction among users. The mood of a user can be implicitly estimated based on opinionated documents generated by the user. Methods of sentiment analysis (SA) assign each user document (e.g. mailing list posting) either to a positive or to a negative class. Methods of SA are often applied to measure the mood of community-generated artifacts, for example blogposts [22]. Within the OSS knowledge mining domain, Jensen et al. [16] analyze the sentiment of OSS mail postings manually.

3 Methods: Requirements Engineering Framework for Process Discovery in OSS

In this section, we describe a method framework for RE process discovery from OSS process history data. Our approach combines several mining techniques which typically should be applied in the following sequence (with possible backtracks as usual):

- A community structure analysis separates core from one or more periphery layers, as we expect these layers to influence the OSS development process in different ways, and our special interest is more the periphery than the already well-understood core of such communities. In this structural analysis for long-lived communities, a special demand is the identification of change and evolution both in the product (OSS) and in the community structure. Here, we propose a release-based rather than a real-time based temporal structuring, and present techniques for its implementation.
- Adapted text mining techniques are employed to detect user requirements among the many messages, in our case studies mostly mail postings, found in the community logs.

– Last not least, we adapt sentiment analysis technologies in order to measure the degree of satisfaction within the different community layers in the different time periods of the community life, as one of the possible outcome measures.

In all phases, noise in the history data (ranging from system-generated messages to external spam, to discussions on topics outside the actual OSS tasks etc.) has proven a major impediment, so data cleaning is a challenge in all three major steps above; space is not sufficient to describe all techniques in detail here, but we shall at least mention the most important ones.

3.1 Structural Analysis of OSS Communities' Evolution

Prior to the mailing lists analysis, multiple aliases of the same individuals are detected and consolidated. Communications created by automated notification services like Bugzilla, Redmine and Nightly Build are excluded from the analysis. Next, for each period between two releases j and $j+1$, an OSS community under study is mapped to a social network graph structure: community members are represented by nodes, their interaction by edges. Thus, for the OSS project with k releases, we generate a sequence of k project graphs:

$$\{graph_{(0,1)}; graph_{(1;2)}; \ldots graph_{(j;j+1)}; \ldots graph_{(k+1;k)}\} \quad (1)$$

Edges are defined as follows. If at least one thread exists, in which two project participants have submitted at least one mail posting in a mailing list, a link between them is added to the project graph. To simplify the further analysis, the edges are unweighted. As previously stated, we assume that core and periphery of the OSS communities participate differently in the RE process. Hierarchical clustering is used in order to separate these two community layers. The method is based on social network properties of community members. In this study, we cluster communities based on degree centrality (the number of edges incident upon a node). The primary approach behind this method is shown in Algorithm 1. In order to track the evolution of periphery and core over time, periphery is separated from core in period between two releases j and $j+1$ in the corresponding social network $graph_{(j;j+1)}$.

3.2 Requirements Detection Within the OSS Mailing Lists

In order to automatically extract postings from the OSS mailing lists which contain requirements, adapted text mining algorithms are applied. First, irrelevant (or even distorting) posts such as quotations, Spam, auto-generated bug reports and announcements are deleted [9]. After this data cleaning, the OSS mailing list postings are considered as bags of words: one posting - one document. Each document is modeled as a vector of features which correspond to terms in the corpus vocabulary.

For the classification tasks, the Naïve Bayes algorithm is applied. The Naïve Bayes technique is one of the most efficient classification learning algorithms [39].

Algorithm 1. Divide social network in two hierarchical layers $C_{periphery}$ and C_{core}

```
Require: Graph G = (V, E) with |V| = N nodes
  Calculate out-degree k_i for node i ∀i, 0 ≤ i ≤ N − 1
  Sort all nodes based on out-degree k_i in ascending order
Ensure: k_min < k_i ∀i, 0 ≤ i ≤ n − 1
  k = k_min; j = 0;
  while j ≤ N do
     Calculate out-degree of the v_j ∈ V node k_j
     if k_j < k then
        k_j = k
     else
        k = k_j
        Remove v_j : V = V/v_j and its edges
     end if
  end while
  Determine (v_h; v_l) with the largest |k_h − k_l|
  j=0;
  while j ≤ N do
     if k_h ≥ k_j then
        Add v_j to C_core
     else
        Add v_j to C_periphery
     end if
  end while
```

It is based on a probabilistic generative model. In order to assign documents (=postings) either to `requirement` or to `non-requirement` groups based on their content, a domain specific lexicon optimized to the jargon of bioinformatics OSS projects is created and presented in the Table 1. In this context, the Bayes rule is defined as follows:

$$Pr(req|words) = \frac{Pr(words|req)Pr(req)}{Pr(words)} \tag{2}$$

where $Pr(req|words)$ is the probability that a document classified to the class `requirements` contains certain `words` which identify this class. The number of requirements is calculated and normalized for each period between two releases $REQ(t_j, t_{j+1})$. To perform classification tasks, the open-source data mining framework RapidMiner is used [18].

3.3 Sentiment Within the OSS Communities

To measure the "mood" within the OSS layers, a proportion of postings with positive sentiment to the overall number of postings is monitored. We expect positive mood among peripheral OSS project participants to reflect the satisfaction with the system and in turn the success of the RE process. In order to

Table 1. Segment of domain specific lexicon for detecting requirements

leak	crack	enhancement	bug	defect
shortcoming	change	adjustments	alter	modify
shift	transform	complain	protest	disagree
mistake	slip	exception	anomaly	deviation
unsuccessful	breakdown	break	crash	fault
insufficiency	misconception	feature	characteristic	highlight
restore	settle	flag	signal	idea
virus	replace	inaccuracy	fail	incompleteness
hypothesis	inspiration	intention	opinion	incorrect
improvement	adjustment	contribution	correction	flow
enrichment	recovery	insufficient	lacking	missing
absent	non-existing	wanting	miss	necessary
mandatory	need	require	vital	want

estimate the polarity of mailing lists' postings, we created a classification model for our data set. We selected a Support Vector Machine (SVM) algorithm as a classification approach, because it showed the most convincing results in the sentiment analysis [24]. A basic SVM classifier applied upon a set of input data classifies each given input into one of two possible classes: `POS` and `NEG`. Initially, the training set has to be provided, in order to infer some general correspondence between the input data and classification groups. Following a particular training set of labeled examples, the learning algorithm constructs a decision rule which can then be used to predict the labels of new unlabeled examples. The decision rule is based on the linear distance function [20].

A sentiment classifier was trained on the polarity data set used by [23] to assign `POS` and `NEG` polarities to the mailing list posts. A training data set adapted to the OSS domain was used to improve model performance. The proportion of positive sentiment to the total amount of postings is calculated for each period between two releases j and $j+1$:

$$\frac{|POS(Postings_{j;j+1})|}{|(Postings_{j;j+1})|} \tag{3}$$

Despite the fact that the training data set, adapted for the OSS domain, consists of 100 entries, the results were improved compared to the initial classification model.

4 Data

Our framework for the RE process discovery is applied to the three large-scale bioinformatics OSS projects: BioJava [15], Biopython [5] and BioPerl [32].

Hereafter, Bio* refers to the three OSS projects. Bioinformatics represents an interdisciplinary research field: innovative computer science technologies and algorithms are developed in order to answer current research questions of computational biology. Interdisciplinary development is indispensable. In the Bioinformatics OSS peripheral project participants are expected to be mainly biologists, who make their first steps towards software development. Thus, generally speaking such OSS also represent a rich approximation for end-user integration in general. The Bio* projects provide open-source bioinformatics frameworks for the manipulation of biological sequences and structures. The frameworks of Bio* projects are based on Java, Python or Perl respectively.

BioJava and Biopython started in 1999, while BioPerl has been already developed since 1996. The infrastructures of the projects include wikipages, developer and general discussion mailing lists, bug management systems and GIT repositories for code management. Table 2 summarizes the status of each project.

Table 2. Bio* OSS overview (on January 1, 2011)

Project	Messages	Users in mailing lists	Commits	Developers
BioJava	11951	2208	8267	94
Biopython	16108	1138	16868	29
BioPerl	31755	2824	12848	139

Beside conceptual similarity, the projects have a long history of over thirteen years, which provides an ideal basis not only for comparative but also comprehensive longitudinal analysis. The entire data set amounts to ca. 60 000 postings from the Bio* mailing lists in the period of eleven years (January 2000–January 2011). In the following section the co-evolution of community, requirements, development and sentiment within the bioinformatics OSS projects is presented. By taking a look at all four dimensions of results, our goal is to relate the changes in RE to the correct historical events in the OSS projects lifetime.

5 Results

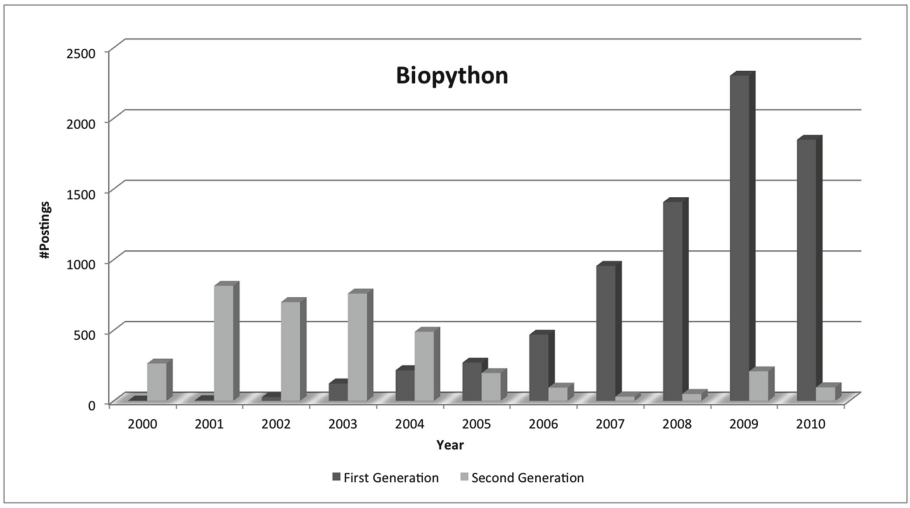
To perform a release-based dynamic analysis, all releases in every project under study are identified. Our investigations show, that in eleven years there were 8 releases in BioJava, 26 in Biopython and 18 in BioPerl.

5.1 Structural Analysis of the OSS Projects

Our study shows that the core members in the Bio* projects are responsible for creating the majority of messages in the mailing lists and of the contributions to the code repositories. Furthermore, the core communities in Bio* projects consist of two to three permanent leaders who play a significant role in the project.

An additional two to three developers from the peripheral communities join the core groups temporarily. Hence, the core groups experience continuous change. Despite similarity in the average size of the core communities (six members), the total number of the project members considerably differs. This results in different proportions of the core size to the total community. The core ratio in the total community in Biopython is 12 %. In BioJava, the ratio is about 6 %, while in BioPerl, it is about 4 %.

BioPerl managed to cultivate the biggest periphery among the three projects under study. More detailed investigation on the Bio* communities shows that the BioPerl community has evolved to a complex structure: the highly active core of creators, the long-tail of lurkers with a very low activity, and the intermediate layer of contributors. This social distribution can be related to "90-9-1" structure from crowdsourcing.[3]



(a) Biopython

Fig. 1. Generation change in the Biopython project

In our study [11] we reported that we detected two generations in each of the Bio* projects. The central members of "first generation" were active near the first five years of the project. They also linked to all other project participants active during that period of time. During the next five years, new leaders together with other user layers formed a second community (="second generation"). As an example, Fig. 1 presents the sum of postings written by the core of each generation year per year in the Biopython project. Due to their private, preferential or personal issues, people spend different amounts of time and

[3] http://www.nngroup.com/articles/participation-inequality/. For example, only 1 % of people create wikipedia-articles and 9 % modify and adjust them, the rest 90 % of wikipedia-users just use the content without any contribution.

effort for an OSS project. The displayed bar chart makes the generation switch obvious. Similar generation switch we observe in BioJava and BioPerl projects.

The generation switch introduces automatically detected changes in terms of sentiment within communities, development progress, requirements production/communication level/requirements creation in an OSS project. For instance, the substitution of the main contributors *Jeffrey C.* and *Brad C.* by *Peter C.* and *Michael H.* in Biopython led to a fivefold increase of releases and threefold increase of commits per year. The modification of the main concepts can induce people to leave an OSS community. The period of change of the core leaders is marked by the decrease of the development activity, especially in case of non-overlapping substitution. Interestingly, the generation switch happens around 2005 in all three projects.

5.2 Co-evolution of Requirements and Development

In the next stage of the RE process discovery framework, the explored social structures are connected to requirement creation and development progress. The amount of submitted software requirements from each sub-community (*core* and *periphery*) for each period (t_j, t_{j+1}) between two releases j and $j+1$ in every Bio* project is identified. An example of identified mail with requirement in the header text: *'Problems runing BLAST; blastall does not exist at blastcmd; New: Bug in PHYLIPFileBuilder with protein sequences.'* An example of identified mail with requirement in the content text: *"Looks like a good time to do the release. Yup, seems good. I guess there is only one request I have before release: Can we fix the tests that are failing? I think it would be nice if people could install biopython and not have tests failing on them. It seems like just some minor adjustments are all we need to do. Brad, how do you make the documentation? Do you have time to do that, or should I try and muddle through it?"* In order to get an approximate insight, if the detected requirements have influenced the project development, the correlation with general development activity is estimated during the release-periods.

In BioPerl, we observe the highest correlation coefficients for both core and periphery. Hence, the more requirements were submitted within the project, the more lines of code were implemented. In BioJava, again both core and periphery show the correlation. However, in this case the coefficients are much smaller indicating weaker influence of the requirements on the development progress. In Biopython, a small correlation could be identified only for the core sub-community. This may be linked to the fact that Biopython has the highest core/periphery ratio compared with other two projects. Due to the small periphery, the development of the Biopython project is perhaps driven by the core. This observation emphasizes the important role of the periphery and supports empirically the claim, that "OSS projects depend on the increase of the size of this user community" [31].

5.3 Sentiment Within the OSS Communities

End-user integration in the requirements negotiation is believed to improve the end-user attitude towards the developing system. To analyze whether the level

of end-user participation in the requirements generation process influences the mood of peripheral community, sentiment analysis is applied to the mailing list postings. Our findings indicate that the general mood within Bio* projects is positive. In all three cases, approximately 60 % of mails from each project are classified to the `POS` group. However, there are several remarkable mood shifts, relevant to further investigation. Due to the space constraint, we provide an example of one community: Biopython. In this community a significant decrease in the number of positively marked messages from the periphery can be observed during period 29 to 37 This period correlates with a significant decrease in requirements fraction from peripheral project members. At the same time, the substitution of core leaders in Biopython happened. Biopython new core leaders, namely *Peter C.* and *Michiel H.* introduced new organizational principles: much shorter release iterations and continuous contribution of a high amount of changes to the code repository. The amount of submitted requirements from the periphery decreased compared to the core and the sentiment together with a great drop in activity within the peripheral community.

Such behavior is likely to be explained by negatively influenced sentiment. Negatively influenced sentiment in its turn can be caused by the fact that the development was mainly centered around the tasks that the community leaders found particularly useful for their own work. Despite not having positive attitude from the periphery, the development process intensively continued by a small group of active contributors from the core. This observation supports the leading role of core members in OSS development.

In BioJava, a decrease of sentiment in the peripheral community is observed during period 5. A more detailed analysis of this period shows that a very high amount of SPAM messages was submitted to the mailing lists [9]. For instance, out of 164 messages only 16 were not SPAM in the BioJava for the period of November, 2004. In the SPAM-free data set, we observe the negative mood within the project periphery. Enormous amount of SPAM annoys people subscribed to the project mailing lists. In [10], we detected the highest user outflow from the BioJava project in the period of the highest SPAM level in the project history. This further supports our assumption that high level of SPAM results in dissatisfaction within the community. During hotfix detection [10], we also discover an extremely large release within the BioJava project (33.5 times more edited lines of code than in most other releases). A manual analysis has shown, that this release was the result of complete restructuring of the project code base. The modification was executed in period 9. Although no sentiment reaction can be detected in the period-oriented view a more fine-granular overview (with a month as a step) shows that the modifications were first met with a negative reaction among peripheral project participants. After some time, the mood within the periphery became more positive again. Big reengineering and restructuring of a project usually has a long-term benefit while in the short run, peripheral members do not appropriate any changes. Restructuring of the project could mean for the peripheral members a need to rewrite their own programs, and therefore presents a short-term disadvantage.

BioPerl proved to be an example of an OSS project with the most "healthy" community and steady project development. Accordingly, the attitude towards the project by core and periphery is stable, both communities have more or less constant mood. The amount of submitted requirements from core and periphery are rather similar. The project development is triggered by the needs from the periphery and the core to the same extent.

6 Discussion and Conclusion

In this paper we proposed a framework for the requirements elicitation (RE) process discovery within OSS projects. The approach was successfully applied to the three bioinformatics OSS communities. Our study shows, that the communication in general and specifically the requirements stated within the community communication resources, give rise to the development process. However, when the core/periphery ratio exceeds 10 %, the development is mainly driven by requirements from the core leaders (**RQ2**). Hence, we do find a difference between the requirements contribution from the core and from the peripheral project participants (**RQ1**). For example, in BioPerl the periphery generates 58 % of requirements, while in BioJava and Biopython the peripheral requirements fraction is only 40 %.

The overall mood within the OSS communities is quite positive. Periods, when the periphery has almost no influence on the project, are marked by a decreasing level of satisfaction among the periphery (**RQ3**). Further, the mood of periphery gets more negative as a reaction to: (1) technical problems within project organization (e.g. high level of SPAM) and (2) major restructuring of the project. The organization of the RE process is mainly defined by the core leaders. The change of core leaders could be a stress factor for the community. In the most-established OSS projects within our study, the requirements from both core and periphery influence the development to a high extent. A stabilizing factor appears to be an intermediate layer of contributors coming from peripheral users. Based on the observed practices within the OSS communities, we hypothesize the following advices how to foster community building of end-users:

- Project managers and/or core developers, who will hold and lead the community need to be set. The management core should consists of 2–3 permanent project participants.
- Detect arising experts among the peripheral participants and motivate (provide special rewards) them to form an interlayer between the project managers and the community long-tail.
- Listen to the community needs, otherwise it will result in a negative mood and community shrinkage.
- Take care of the technology used for community management. Errors and disturbances within the community tools can chase the users away.
- In case of serious project restructuring, take time to explain the reasons for and advantages of the planned changes. Otherwise, it could cause some resistance of peripheral project participants.

Our findings are consistent with the few existing results of evolution studies achieved by other researchers. For example, [27] conducted a quantitative study of the evolution of the Debian community. The authors found out that the volunteer teams are dynamic and changeable over time, while their efforts are stable and reliable.

6.1 Threats to Validity

First, the quality of text and sentiment mining strongly depends on the quality of training data sets used for the classification models. These data sets have to be adapted for the OSS domain. Moreover, bioinformatics OSS projects are mostly driven by bioinformatics scientists and, therefore, presents an exploration-oriented [38] OSS. It is interesting further investigate Bio* communities in order to detect other sub-communities and their influence to development process. As also, further studies with domains outside bioinformatics are needed to achieve truly generalizable results.

Moreover, the quality of any data mining analysis is as good as the data. It can always happen, that some important decisions or negotiations take place privately. Cross examination of our automatically achieved results with other data acquisition methods such as interviews would thus be helpful to further validate our results. Within this work Bayes and SVM algorithms were used as the base for the requirements detection and opinion mining models. Those can be effectively extended by considering new achievements in the text mining discipline. For example, one of the possible ways to improve our sentiment classification model is to use the method based on Part-of-Speech (POS) tagging. Last but not least, in OSS mailing lists the same requirement can be described within various artifacts. Differently formulated identical ideas currently are classified as distinct requirements. The clustering model based on latent semantic analysis (LSA) can be used in order to identify similar requirements even if they do not share any common words.

References

1. Alexander, I.: Migrating towards co-operative requirements engineering. Comput. Control Eng. J. **10**(1), 17–22 (1999)
2. Bhowmik, T., Reddivari, S.: Resolution trend of just-in-time requirements in open source software development. In: Just In Time RE Workshop, Canada (2015)
3. Boehm, B., Grünbacher, P., Briggs, R.: Developing groupware for requirements negotiation: lessons learned. IEEE Softw. **18**(3), 46–55 (2001)
4. Chesbrough, W.: Open Innovation: The New Imperative for Creating and Profiting from Technology. Harvard Business School Press, Boston (2003)
5. Cock, P., Antao, T., Chang, J., Chapman, B., Cox, C., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., de Hoon, M.: Biopython: freely available python tools for computational molecular biology and bioinformatics. Bioinformatics **25**(11), 1422–1423 (2009)
6. Crowston, K., Howison, J.: Hierarchy and centralization in free and open source software team communications. Knowl. Technol. Policy **18**, 65–85 (2006)

7. Crowston, K., Wei, K., Li, Q., Howison, J.: Core and periphery in free/libre and open source software team communications. In: Proceedings of the 39th Annual Hawaii International Conference on System Sciences, HICSS 2006. IEEE Computer Society, Washington, D.C. (2006)
8. Ernst, A., Murphy, C.: Case studies in just-in-time requirements analysis. In: Proceedings of the Second IEEE International Workshop on Empirical Requirements Engineering (EmpiRE), pp. 25–32 (2012)
9. Hannemann, A., Hackstein, M., Klamma, R., Jarke, M.: An adaptive filter-framework for the quality improvement of open-source software analysis. In: Kowalewski, S., Rumpe, B. (eds.) Software Engineering. LNI, vol. 213, pp. 143–156. GI (2013)
10. Hannemann, A., Klamma, R.: Community dynamics in open source software projects: aging and social reshaping. In: Petrinja, E., Succi, G., El Ioini, N., Sillitti, A. (eds.) OSS 2013. IFIP AICT, vol. 404, pp. 80–96. Springer, Heidelberg (2013)
11. Hannemann, A., Klamma, R., Jarke, M.: Soziale Interaktion in OSS. Praxis der Wirtschaftsinformatik (2012)
12. Hauge, O., Ayala, C., Conradi, R.: Adoption of open source software in software-intensive organizations - a systematic literature review. Inf. Softw. Technol. **52**(11), 1133–1154 (2010)
13. Hippel, E.: Lead users: a source of novel product concepts. Manag. Sci. **32**(7), 791–805 (1986)
14. Hippel, E., Krogh, G.: Open source software and the "private-collective" innovation model: Issues for organization science. J. Organ. Sci. **14**(2), 208–223 (2003)
15. Holland, R., Down, T., Pocock, M., Prlić, A., Huen, D., James, K., Foisy, S., Dräger, A., Yates, A., Heuer, M., Schreiber, M.J.: Biojava: an open-source framework for bioinformatics. Bioinformatics **24**(18), 2096–2097 (2008)
16. Jensen, C., King, S., Kuechler, V.: Joining free/open source software communities: an analysis of newbies' first interactions on project mailing lists. In: Proceedings of the 44th Hawaii International Conference on System Sciences (HICSS), pp. 1–10 (2011)
17. Klamma, R., Spaniol, M., Cao, Y.: MPEG-7 compliant community hosting. J. Univ. Knowl. Manag. **1**(1), 36–44 (2006)
18. Land, S., Fischer, S.: Rapid Miner in Academic Use (2012)
19. Lehman, M., Ramil, F., Wernick, D., Perry, E., Turski, M.: Metrics and laws of software evolution - the nineties view. In: Proceedings of the Fourth International Software Metrics Symposium, pp. 20–32 (1997)
20. Lovell, C., Walder, C.: Support vector machines for business applications. In: Voges, K., Pope, N. (eds.) Business Applications and Computational Intelligence, pp. 267–290. IGI Global, Hershey (2006)
21. Maiden, N., Jones, S., Karlsen, K., Neill, R., Zachos, K., Milne, A.: Requirements engineering as creative problem solving: a research agenda for idea finding. In: Proceedings of the 18th IEEE International Requirements Engineering Conference, pp. 57–66 (2010)
22. Melville, P., Gryc, W., Lawrence, D.: Sentiment analysis of blogs by combining lexical knowledge with text classification. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009, pp. 1275–1284. ACM, New York (2009)
23. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retrieval **2**(1–2), 1–135 (2008)

24. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up?: Sentiment classification using machine learning techniques. In: Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing, EMNLP 2002, vol. 10, pp. 79–86. Association for Computational Linguistics, Stroudsburg (2002)
25. Raymond, E.: The Cathedral and the Bazaar. O'Reilly Media, New York (1999)
26. Robles, G., Gonzalez-Barahona, J.M.: Contributor turnover in libre software projects. In: Damiani, E., Fitzgerald, B., Scacchi, W., Scotto, M., Succi, G. (eds.) Open Source Systems, vol. 203, pp. 273–286. Springer, Boston (2006)
27. Robles, G., Gonzalez-Barahona, J.M., Michlmayr, M.: Evolution of volunteer participation in libre software projects: evidence from debian. In: Scotto, M., Succi, G. (eds.) Proceedings of the First International Conference on Open Source Systems, pp. 100–107 (2005)
28. Scacchi, W.: Understanding requirements for open source software. In: Lyytinen, K., Loucopoulos, P., Mylopoulos, J., Robinson, B. (eds.) Design Requirements Engineering. LNBIP, vol. 14, pp. 467–494. Springer, Heidelberg (2009)
29. Scacchi, W.: The future research in free/open source software development. In: Proceedings of ACM Workshop on the Future of Software Engineering Research (FoSER), Santa Fe, NM, pp. 315–319 (2010)
30. Seyff, N., Graf, F., Maiden, N.: Using mobile re tools to give end-users their own voice. In: Proceedings of the 18th IEEE International Requirements Engineering Conference, pp. 37–46 (2010)
31. Sowe, S.K.: Emerging Free and Open Source Software Practices. IGI Publishing, Hershey (2007)
32. Stajich, E., Block, D., Boulez, K., Brenner, E., Chervitz, A., Dagdigian, C., Fuellen, G., Gilbert, J., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C., Osborne, B., Pocock, M., Schattner, P., Senger, M., Stein, L., Stupka, E., Wilkinson, M., Birney, E.: The bioperl toolkit: Perl modules for the life sciences. Genome Res. **12**(10), 1611–1618 (2002)
33. Sutcliffe, A.: Scenario-based requirements engineering. In: Proceedings of the 11th IEEE International Conference on Requirements Engineering, RE 2003, pp. 320–329. IEEE Computer Society, Washington, D.C. (2003)
34. Sutcliffe, A.: Evaluating the costs and benefits of end-user development. SIGSOFT Softw. Eng. Notes **30**(4), 1–4 (2005)
35. Vlas, R., Robinson, W.N.: A rule-based natural language technique for requirements discovery and classification in open-source software development projects. In: Proceedings of the 44th Hawaii International Conference on System Sciences (2011)
36. Wiggins, A., Howison, J., Crowston, K.: Heartbeat: measuring active user base and potential user interest in FLOSS projects. In: Boldyreff, C., Crowston, K., Lundell, B., Wasserman, A.I. (eds.) OSS 2009. IFIP AICT, vol. 299, pp. 94–104. Springer, Heidelberg (2009)
37. Wulf, V., Jarke, M.: The economics of end-user development: tools that empower users to create their own software solutions. Commun. ACM **47**(9), 41–42 (2004)
38. Ye, Y., Nakakoji, K., Yamamoto, Y., Kishida, K.: The co-evolution of systems and communities in free and open source software development. In: Koch, S. (ed.) Free/Open Source Software Development, pp. 59–82. Idea Group Publishing, Hershey (2004)
39. Zhang, H.: The optimality of naive bayes. In: Barr, V., Markov, Z. (eds.) FLAIRS Conference, pp. 562–567. AAAI Press, Miami Beach (2004)