Responsiveness as a measure for assessing the health of OSS ecosystems

Jonas Gamalielsson, Björn Lundell and Brian Lings
University of Skövde, Sweden
{jonas.gamalielsson, bjorn.lundell, brian.lings}@his.se,
WWW home page: http://www.his.se

Abstract. The health of an Open Source ecosystem is an important decision factor when considering the adoption of Open Source software or when monitoring a seeded Open Source project. In this paper we introduce responsiveness as a qualitative measure of the quality of replies within mailing lists, which can be used for assessing ecosystem health. We consider one specific metric of responsiveness in this paper, and that is the response time of follow-up messages in mailing lists. We also describe a way for characterising the nature of communication in messages with short and long response times. The approach is tested in the context of the Nagios project, and we particularly focus on the responsiveness for contributors acting in their professional roles as core developers. Our contribution is a step towards a deeper understanding of voluntary support provided in mailing lists of OSS projects.

1 Introduction

Before an organisation adopts an Open Source project it is important to evaluate its community in order to make sure that it is healthy and that the project is likely to be maintained for a long time (van der Linden et al. 2009). It may be especially important to monitor the health of seeded Open Source projects. One important means in such an evaluation is to quantitatively assess the health of an Open Source community (Crowston and Howison 2006).

A number of studies have investigated large, well known Open Source projects through quantitative analysis, including the Linux kernel (Moon and Sproull 2000), Apache (Mockus et al. 2002), Mozilla (Mockus et al. 2002), Gnome (German 2004) and KDE (Lopez-Fernandez 2006). Several of these studies focus on social network analysis from different kinds of data sources such as CVS/SVN (Martinez-Romo et al. 2008), bug reports (Crowston and Howison 2005) and mailing lists (Kamei et al. 2008).

In earlier work (Gamalielsson et al. 2009, Gamalielsson et al. 2010) we have shown examples of how OSS communities can be assessed using quantitative means such as social network analysis and domain analysis. The results from the earlier work also show possible characteristics of healthy OSS communities represented by the mailing list communities for Nagios, a tool for monitoring IT infrastructure that has been used in many professional organisations and mission critical systems.

It has previously been shown that professional developers often perceive communication with Open Source communities to deliver quick responses (Lundell et al. 2010). In this paper we elaborate on a means for exploring such perceptions, and report on findings for the Nagios project. In so doing, we specifically elaborate on the concept of responsiveness in Open Source communities. We introduce responsiveness as a qualitative measure of the quality of replies within mailing lists. One specific metric of responsiveness is considered in this paper, and that is the response time which indicates how quickly messages are replied to in mailing lists for OSS projects. We also propose an approach to characterising the nature of communications in messages with different response times. The analysis in particular focuses on core developers, since it is one important role in Open Source projects together with users, developers and project leaders (Crowston and Howison 2006). This analysis is possible since the core developers of Nagios are explicitly listed on the Nagios website (www.nagios.org/development/ teams/core, accessed on 29 March, 2010). The importance of core developers applies to any Open Source project as it is well established in the literature that core developers "contribute most of the code and oversee the design and evolution of the project." (Crowston et al. 2006). To the best of our knowledge responsiveness has not been reported on earlier in the context of mailing lists for OSS projects.

2 Research Approach

Data was collected from the GMANE (gmane.org) archives of the SourceForge "Nagios-devel" mailing list for the period from January 2004 to October 2009. This list is intended for development related issues in Nagios. The GMANE export interface was used to retrieve the raw mbox files for the project. For a message, the "message-id" field was used to get a unique message identifier. The "in-reply-to" field of a message was used, which contains the identifier for the message it was a reply to. Each message also has a "from" field from which the name and email address of the sender were derived. Finally, the "date" field was used to obtain the time stamp of the message. Data cleaning was performed to make sure that the same person does not appear several times using different identifiers.

In order to establish how quickly individual messages sent to the developer mailing list are replied to, we calculated the response time for all replies to messages sent to the developer mailing list. The response time is defined as the difference in time between the reply to a message and the posting of the original message. Corrections were made to account for different time zones.

In order to characterise the content of individual messages, we manually inspected messages sent to the developer mailing list and responded to by core developers for the time period January 2004 to October 2009. We concentrated on two subsets of these messages, namely those with short and long response times. For each of these sets we analysed the content of all messages in order to gain a more comprehensive understanding of the nature of communication. Our initial analysis aimed to gain an overarching impression of individual messages and identify different kinds of communication. We also aimed to identify possible differences in type of content for messages with different response times.

3 Results

In this section we show how responsiveness can serve as a measure to establish the health of an OSS project, here exemplified in the context of the Nagios project. In our analysis we separate core developers from other contributors in order to illustrate the involvement of a professional OSS stakeholder.

Figure 1 shows the number of replies to earlier messages in the developer mailing list for the Nagios project, where the core developers are separated from other contributors. It can be observed that the number of replies from these two groups is in the same range, and that contributors not being core developers have posted more replies since October 2007. Another observation is that core developers were more active than other contributors during three time periods, where the longest lasted from July 2004 to July 2005. It is interesting to note that the peak for core developers around July and October 2007 co-occurs with the first beta-release and first release candidate of Nagios v3.0.

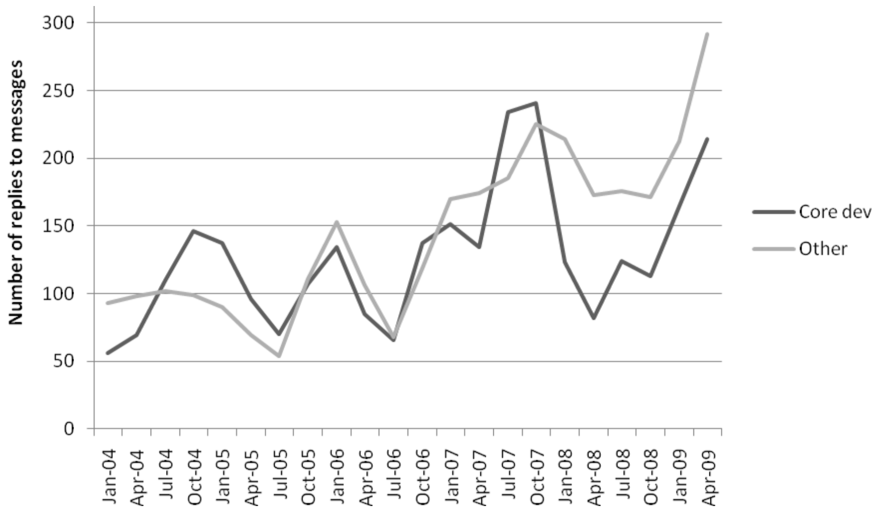

Figure 1. Number of replies to messages in the developer mailing list for core developers and other contributors during the 22 six-month time windows from January 2004 to April 2009.

In order to understand how messages are distributed with respect to response time, different percentiles were calculated for message replies during the 22 time windows for core developers (figure 2) and those that are not core developers (figure 3). The bottom trace in each figure represents the first percentile, followed by percentiles 17, 33, 49, 65, 81 and 97. The y-axis shows the response time in hours on a logarithmic scale to base 10, meaning for example that -1 represents $10^{-1} = 0.1$ hours = 6 minutes and that a y-value of 3 represents 1000 hours = 41 days. As an example, the 97th percentile in figure 2 has a response time ranging between 13 and 41 days from January 2004 to June 2006, after which the response time occasionally exceeds 41 days. An important observation is that core developers generally are slower in their response compared with other contributors. It can also be noted that there is a larger absolute difference in response time between percentiles farther from the 49th percentile, which implies that the response time distribution is non-uniform.

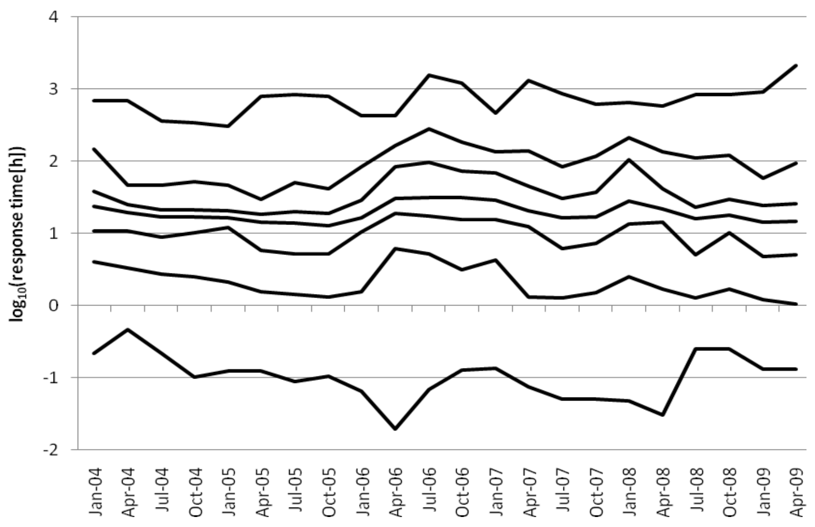

Figure 2. Response times for different percentiles (from bottom to top: 1, 17, 33, 49, 65, 81 and 97) for core developers replying to messages in the developer mailing list.

To establish a deeper understanding of the nature of communication taking place in the developer mailing list we analysed the content of individual messages. We focused on replies posted by core developers in the developer mailing list. Furthermore, we extracted messages with short response times (between percentiles 0 and 5) and long response times (between percentiles 95 and 100). For each of the two chosen intervals there are 74 messages. From our initial analysis we identified three main kinds of messages, which we refer to as *statement*, *question* and *proposal*. We used these three as categories in our further analysis. The *statement* category comprises messages that contain simple statements of different kinds or explanations typically related to usage or installation. The *question* category includes questions or reports on errors, bugs or other deficiencies found in the code or during execution of the system. The *proposal* category contains concrete proposals for software solutions (patches) or other algorithmic improvements closely related to the source code.

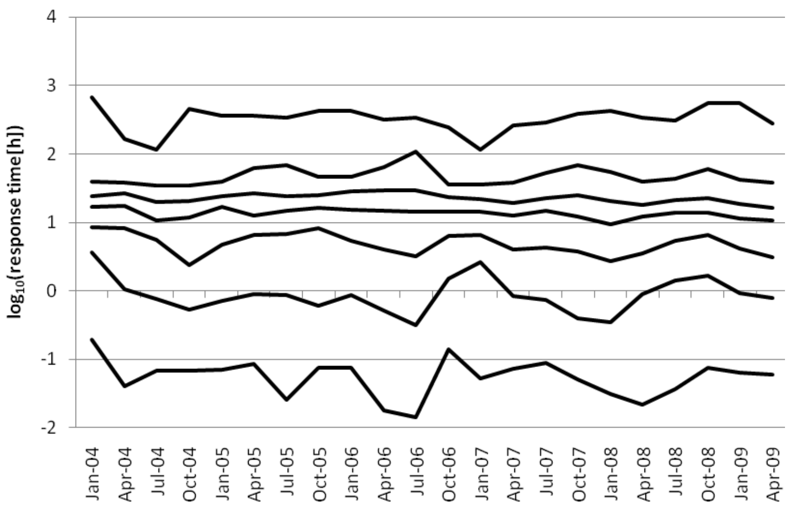

Figure 3. Response times for different percentiles (from bottom to top: 1, 17, 33, 49, 65, 81 and 97) for other contributors replying to messages in the developer mailing list.

We subsequently chose to create pairs of message categories for the observed combinations of original message and reply to that message. The observed frequencies of category pairs for the two chosen response time intervals are shown in table 1.

Table 1. Percentage of replies belonging to a specific category pair for the two different response time intervals. Replies are posted by core developers in the developer mailing list.

Category pair	**Quick response (0-5%)**	**Slow response (95-100%)**
Statement-Statement	20%	0%
Statement-Question	1%	0%
Statement-Proposal	0%	0%
Question-Statement	54%	16%
Question-Question	3%	0%
Question-Proposal	1%	7%
Proposal-Statement	15%	53%
Proposal-Question	1%	3%
Proposal-Proposal	4%	22%

It is evident that the “Question-Statement” pair is a dominant theme in the 0-5% interval, whereas it only appears to a limited extent in the 95-100% interval. A communication of this type can be that there is a question or bug report followed by a statement that says that the patch will be put in the CVS. For this kind of communication an example original message is "... immediately in the log file those lines show up: Warning: Attempting to execute the command "/check_ping ..." resulted in a return code of 127 ...", with the reply "... This most likely has to do with the fact that the forked processes receive a SIGHUP as well as the original one ...". A possible reason for the short response times for "Question-Statement" communications is that core developers prioritise technical questions and bug reports that can be answered with relatively limited effort.

Another major observation is that the "Proposal-Statement" and "Proposal-Proposal" pairs are dominating themes in the 95-100% interval, but appear to a considerably smaller extent in the 0-5% interval. Communications complying with "Proposal-Statement" in the 95-100% interval are most often about a submitted patch that will be put in CVS after approval by a core developer. A typical reply in such a communication is "... Thanks for all the patches [*name removed*]! They will be committed to CVS shortly ...". An interpretation is that patches usually take time to respond to since core developers need to review and test the patch first. A "Proposal-Proposal" pair in the same interval is often identified where a patch/solution is suggested in the original message and a modification (or a different patch/solution) is proposed in the reply, as evidenced in the reply "... I think I'll reimplement the patch as a recursive function that allows multiple levels of nesting ...".

It can also be noticed that the "Statement-Statement" category pair is well represented in the 0-5% interval, but not at all in the 95-100% interval. This kind of communication can be about a comment to a comment, for example the original message " ...Yes, but I prefer modify some C lines than PHP :)" with the answer "The web-interface is written in C, so it'd be the same there.". It may also be a communication that should be ignored, as evidenced in the original message: "... I am running a relative huge installation with up to 5 instances (for load balancing) on one hardware server (yes - that works) ...", followed by the reply "... Sent too early...Please ignore this ...". Another example is "... I suggest you get a nice .indent.pro (mine is attached) and run it ..." followed by the reply "Forgot the file.".

From the analysis of message content it was observed that some replies with short response times have original messages that should have been posted on a different mailing list, as evidenced in the reply "... Please send questions like these to the nagiosplug-help list instead ...". It may also be suggested in some messages that complementary information should be supplied, for example "... Please cut'n paste your service-notification command in a mail and send it here ...". It may also be that the same contributor wants to add some additional information to the original message, as evidenced in the reply "... And for the 1 millionth time I've forgotten to attach. This calls for celebration! ;) ...".

5 Conclusion and discussion

In this paper we have proposed approaches for analysing the responsiveness of OSS communities, and for establishing the characteristics of communication in a healthy OSS community. Further, we have reported on a novel application of the proposed approaches for the Nagios project. Our findings regarding the nature of communication were largely unsurprising, which may be expected for a healthy community.

As future work we plan to look for "unhealthy" OSS communities and analyse these using the same approaches in order to see whether the nature of communication differs significantly from the case with healthy OSS communities. In our current work we have analysed message content manually, but we would also like to develop automated approaches, for example using GATE[1], tuning the process using the results from this analysis.

In healthy Open Source communities people are active and responsive to questions during the life cycle of a software system. It is important to consider such indicators of health prior to any organisational adoption or during the seeding of a community. The kinds of analysis elaborated in this paper serve as important means for establishing the health of an Open Source community.

6 Acknowledgement

The authors would like to thank their colleagues in the OSA-project for their encouragement and support. The OSA-project is financially supported by the Knowledge Foundation ("KK-stiftelsen").

References

Crowston, K. and Howison, J. (2005). The social structure of Free and Open Source software development. First Monday, 10(2).

Crowston, K. and Howison, J. (2006). Assessing the Health of Open Source Communities. IEEE Computer, 39(5):89-91.

Crowston, K., Wei, K., Li, Q. and Howison, J. (2006). Core and Periphery in free/Libre and Open Source software team communications. In Proceedings of the 39th Hawaii International Conference on System Sciences, page 118.1.

Gamalielsson, J., Lundell, B. and Lings, B. (2009). Social Network Analysis of the Nagios project. Open Source Workshop (OSW 2009), Skövde, Sweden, October 15-16 2009.

Gamalielsson, J., Lundell, B. and Lings, B. (2010). The Nagios community: An extended quantitative analysis. In Proceedings of the 6th International Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, 30 May – 2 June 2010 (to appear).

[1] http://gate.ac.uk/

German, D. (2004). The GNOME project: a case study of open source global software development. Journal of Software Process: Improvement and Practice, 8(4): 201–215.

Kamei, Y., Matsumoto, S., Maeshima, H., Onishi, Y., Ohira, M. and Matsumoto, K. (2008). Analysis of Coordination Between Developers and Users in the Apache Community. In Proceedings of the Fourth Conference on Open Source Systems (OSS 2008), pages 81-92.

van der Linden, F., Lundell, B. and Marttiin, P. (2009). Commodification of Industrial Software: A Case for Open Source. IEEE Software, 26 (4): 77-83.

Lopez-Fernandez, L., Robles, G., Gonzalez-Barahona, J. M. and Herraiz, I. (2006). Applying Social Network Analysis Techniques to Community-driven Libre Software Projects. International Journal of Information Technology and Web Engineering, 1:27–48.

Lundell. B., Lings, B. and Lindqvist, E. (2010). Open source in Swedish companies: where are we? Information Systems Journal, DOI: 10.1111/j.1365-2575.2010.00348.x. (Early view)

Martinez-Romo, J., Robles, G., Ortuño-Perez, M. and Gonzalez-Barahona, J. M. (2008). Using Social Network Analysis Techniques to Study Collaboration between a FLOSS Community and a Company. In Proceedings of the Fourth Conference on Open Source Systems (OSS 2008), pages 171–186.

Mockus, A., Fielding, R. T. and Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and Mozilla. ACM Transactions on Software Engineering and Methodology, 11 (3): 309–346.

Moon, Y. J. and Sproull, L. (2000). Essence of distributed work: The case of the Linux kernel. First Monday, 5 (11).