Supporting Newcomers in Software Development Projects

Sebastiano Panichella
University of Zurich, Switzerland
panichella@ifi.uzh.ch

***Abstract*—The recent and fast expansion of OSS (Open-source software) communities has fostered research on how open source projects evolve and how their communities interact. Several research studies show that the inflow of new developers plays an important role in the longevity and the success of OSS projects. Beside that they also discovered that an high percentage of newcomers tend to leave the project because of the *socio-technical barriers* they meet when they join the project. However, such research effort did not generate yet concrete results in support retention and training of project newcomers. In this thesis dissertation we investigated problems arising when newcomers join software projects, and possible solutions to support them. Specifically, we studied (i) how newcomers behave during development activities and how they interact with others developers with the aim at (ii) developing tools and/or techniques for supporting them during the integration in the development team. Thus, among the various recommenders, we defined (i) a tool able to suggest appropriate mentors to newcomers during the training stage; then, with the aim at supporting newcomers during program comprehension we defined other two recommenders: A tool that (ii) generates high quality source code summaries and another tool able to (iii) provide descriptions of specific source code elements. For future work, we plan to improve the proposed recommenders and to integrate other kind of recommenders to better support newcomers in OSS projects.**

I. Introduction

Open-source software (OSS) projects consist of very complex communities, where the participants, who are mostly volunteers, are distributed across different geographic locations [1], [2]. Developers of such OSS communities develop software in a public and collaborative manner relying on versioning systems and different kinds of communication channels such as, mailing lists, issue trackers and IRC chat [3], [4]. Thus, the software is developed, tested, or improved through public collaboration and distributed with the idea that the output of this process must be shared with others [5], [6]. In such context the inflow of new developers plays an important role in the longevity and the success of OSS projects [7], [8]. Indeed, without replacing members who leave, a community will eventually wither away [9]. Kraut *et al.* [9] investigate the challenges of dealing with newcomers and point out several basic problems that online communities must solve:

- ***Recruitment***: Ensure a continuous influx of new developers;
- ***Selection***: OSS communities try to select the newcomers that are more motivated and who fit well the project needed [10]–[12]. For example, the *Apache Software Foundation* (ASF) has a *newcomers support page*[1] that points out that *"The more you give the more you get out"*);
- ***Retention***: An OSS project needs to reduce the percentage of newcomers that leave the project because of the socio-technical barriers that they meet when they join the project [7], [10], [13];
- ***Socialization***: "Teaching" to newcomers how to behave with developers teams.

Others researchers investigated the reasons behind the newcomers decision to abandon the project studying the difficulties and obstacles that they encounter starting their contributions [7], [10], [13]–[15]. However, such research effort did not generate yet concrete results to support retention and training of project newcomers.

II. The challenges of dealing with newcomers

This dissertation has the aim at conceiving possible approaches (and implementing tools) with the main goal to support OSS project newcomers during the training process [10]. As shown in Figure 1 various are the phases that characterizes a proper newcomer training process [8], [10]: (i) ***Mentoring activity***, (ii) ***maintenance/development tasks***, (iii) ***team collaboration***. All of these phases are very interlinked and are important stages for a complete and satisfactory newcomer training.

Specifically, *mentoring activity* by experienced developers represents the initial part of the training process in which a newcomer is trained by mentors in OSS project to acquire the technical and organizational information she need to properly work in the project [14]–[16]. This phase is relevant also in an industrial setting. Indeed, Dagenais *et al.* [13] performed a survey involving 18 IBM developers and discovered that *"mentoring of projects newcomers is highly beneficial"* and help to avoid problems/barriers that affect the newcomers participation [7], [10], [13]. However, nothing concrete was proposed to help newcomers in this first stage of the newcomer training. For these reasons, this dissertation investigates the possibility to build a recommender able to **suggest appropriate mentors** having adequate skills to properly train OSS newcomers.

Once a newcomer is trained by mentors of the projects she is interested to gain familiarity with source code and the related documentation to perform the first *maintenance/development tasks*. In such context *program comprehension* is preliminary

[1] https://community.apache.org/newcomers/

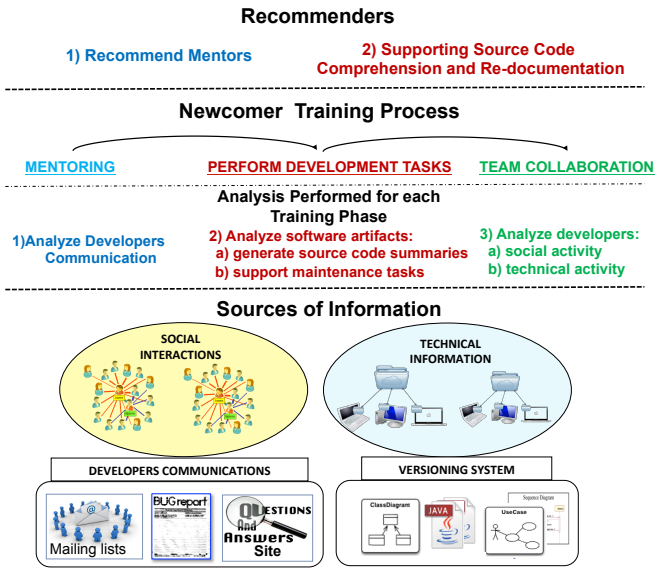

Fig. 1. Newcomer Training Process: Three high level phases.

to each development activity. However, especially for new developers, understand software written by other developers is a very effort consuming task. This is particularly true when source code lacks of comments that adequately describe its behaviour. Thus, this thesis, investigates the possibilities to (i) **build high quality summaries of source code elements** and (ii) **mining source code descriptions from developer's communication** to improve newcomers program comprehension.

Finally, as last step of the newcomer training, developers teams need to "teaching" newcomers how to behave with developers teams in a way that newcomers can perform development tasks benefited by the *team collaboration*. However, recent studies suggest that newcomers ties in the OSS communities are pretty fragile [7]–[10]. For this reason, in this dissertation we also analyze development discussions (in form of issue trackers, mailing lists and IRC chat), to extract relevant facts from the "social environment" that can be exploited to support newcomers during team work.

III. Dissertation Overview

This dissertation is composed by nine chapters. Each chapter is dedicated towards a specific research problem and is conceived as a self-contained unit, such that readers can read each chapter independently. We organized the dissertation into three parts (as shown in Figure 1):

- **Part I - Analysis of Developers' Communication**: This part of the dissertation *analyzes* developers' communication over different channels (mailing lists, issue trackers, IRC chat) and their co-change activities captured from versioning systems with the main *goal* to derive important facts from the *social environment*. The findings of this part of the dissertation are used to build possible recommenders to better support newcomers in the integration of the development team.
- **Part II - How Developers Browse and Understand Software Artifacts**: This part of the dissertation *investigates* how newcomers navigate and browse documentation artifacts during maintenance tasks. The findings of this part of the dissertation are used to build possible recommenders able to improve newcomers program comprehension of code written by others developers.
- **Part III - Recommenders**: In this last part of the dissertation, we use the findings of the previous parts to *build recommenders* able to support newcomer during the training process. Clearly, other researchers can use the findings of Part I and Part II of the dissertation to perform further studies and design other kind of recommenders.

IV. Dissertation Summary and Contributions

In the following we provide short summaries of each part of the doctoral dissertation, together with a summary of the *technical* (tools and techniques) and *conceptual* (or empirical) contributions to the research field.

A. Part I - Analysis of Developers' Communication

Analysis Performed: This part of the dissertation analyzes development discussion (e.g. mailing lists, issue trackers, IRC chat) and tries to determine the most reliable communication channel(s) to getting touch with more experienced developers that, from the newcomer point of view, cover important project roles (e.g., project *mentors* and *coordinators*). Moreover, we also investigates how dependencies between sub-projects are discussed by experienced developers of OSS projects.

Summary of the Main Findings:

- *Analyzing developers collaboration/communication through specific channels would only provide a partial view of the reality*; indeed, people interacting through a given channel may not necessarily also communicate through other channels [3], [4].
- *Newcomers should rely on more than communication channel*, to have a real indication of the social/technical roles played by a given developer [4].
- *Issue trackers and mailing lists* are more appropriate sources for the identification of key project roles for the newcomers—such as developers with a high communication degree or potential good *mentors*— perspective [4].

Such preliminary analysis has made it possible to define in this dissertation an approach able to detect *"emerging" teams* of OSS project by analyzing developer' socio-technical activity [3], [17]. This approach rely on issue trackers and mailing lists (the most reliable channels) to detect developers teams. We designed this approach to help newcomers to getting in touch with the main emerging teams of the project and more important to have a better view of existing developers' collaborations.

However, by analyzing how developers teams evolve over time, and to what extent such evolution relates with the developers' activity on source code [3], [5] we found relevant dynamics of developers collaborations in open source projects:

- *Emerging teams tend to recombine over time*: We observed groups having a stable collaboration over time, although involving different other people moving between different emerging teams [3].

- *Software ecosystems size exponentially grows over time, and consequentially the dependencies between projects grow too* [5], [6].
- *developers discussions are massively characterized by dependencies discussion of dependent sub-projects* [5], [6].

Possible Future Directions: We plan to develop a new recommender able to extract from development discussions paragraphs describing dependencies with third-party libraries and help newcomers to make informed decision and avoid changes that could break the dependency with used libraries.

B. Part II - How Developers Browse and Understand Software Artifacts

Analysis Performed: This part of the dissertation investigates (i) to what extent newcomers use different kinds of documentation when identifying artifacts to be changed, and (ii) whether they follow specific navigation patterns. Such analysis are used to defined an optimal (ad-hoc) approach to generate high-quality summaries of source code to facilitate newcomers program comprehension.

Summary of the Main Findings:

- *Newcomers generally start their tasks from source code, or from design documents* (84% of cases), then browsing back and forth between source code and either class or sequence diagrams.
- *Experienced (or senior) developers—-follow an "integrated" approach*, by using different kinds of artifacts before accessing the source code.
- Participants (newcomers/senior developers) of the survey considered very important *source code identifiers for understanding the source code* and perform any maintenance/development activity.

This result motivated us to conduct a further empirical study where we investigating to what extent a source code labeling based on IR techniques can be used to generate *high quality summaries* of source code. Thus, we defined a set of *ad hoc* heuristics picking terms from specific parts of the source code and comments and compared them with traditional techniques such as VSM, LSI, LDA [18], [19]. We discovered that *Developers mainly used words from class names, method names and signatures*, and (partially) from class and method comments to label artifacts. *The highest overlap with human generated summaries is obtained by the experimented ad-hoc heuristics*, while the most sophisticated techniques, i.e., LSI and LDA, provide generally the worst accuracy. Thus, the experimented *ad-hoc* heuristics *generate accurate summaries* of source code useful to support newcomers program comprehension.

Possible Future Directions: We plan to (i) develop new recommenders able to suggest to newcomers how to proper navigate software artefacts during maintenance tasks and (ii) define a wider set of summarization techniques to support them during program comprehension.

C. Part III - Recommenders

This section describes the recommenders (or tools) we designed to support newcomer during the training process.

Proposed Recommenders and Main Findings: In this dissertation we presented an approach, named YODA (Young and newcOmer Developer Assistant) [20], [21] to identify mentors by relying on historical data of a software project, and then recommend them when a newcomer joins the project (the tool is available for download[2]). YODA has been evaluated on data of several open source projects using as as historical data, mailing list and issue tracker data (as suggested in Part I of the dissertation) and data from versioning systems. Results of the empirical evaluation highlight that:

- *YODA identifies candidate pairs of mentor-newcomer with a precision in most cases higher than 80%, and recommend them with a precision greater than 70%.*
- *The presence of project mentors in software project impacts positively the trajectory (career) of newcomers that joined the project*: Newcomers trained by mentors have an higher permanence (in year) in the project (almost twice) with respect to developers that did not receive any initial support.

As discussed in Section II, suggesting mentors is only a part of the support that newcomer needs when she join a software project. We are also interested to support newcomers during program comprehension and code re-documentation tasks. For this reason we proposed an approach named CODES (mining sourCe cOde Descriptions from developErs diScussions) based on ad-hoc heuristics which mine method descriptions from developers communication (i.e. mailing lists and issue trackes) and automatically generate Javadoc comments [22], [23] (the tool is available for download[3]). The proposed approach is able to identify descriptions with a precision up to 79% with a method coverage ranging between 7% and 36%.

V. Advice to New and Young Researchers

As part of this post-doctoral dissertation presentation, we would like to present the following pieces of advice that we would give a younger us:

- **Trust in your ideas**: Especially in the beginning of the PhD we do not believe a lot in our ideas and their potential usefulness, we trust more in ideas proposed by more senior researchers. In the majority of the cases this is not wrong, especially when we start of our academic career where we are academically *younger*. In this way something goes wrong and something goes better but it helps us to grow up as researchers and as people.
- **Be open to collaborate**: In any research environment, as in private life, *No man is an island, entire of itself*. It is really important to (i) follow conferences and *hear* people presenting interesting new ideas related to our research interests; (ii) reading the literature; and be (iii) open to share with other researchers our ideas. The best ideas born in a collaborative environment.

[2]http://www.ifi.uzh.ch/seal/people/panichella/tools/YODA-tool.html

[3]http://www.ifi.uzh.ch/seal/people/panichella/tools/CODES-tool.html

- **Validate your ideas**: An important part of our career is focused in developing ideas and evaluating them. Even if evaluating research ideas is a very effort consuming task, it is very important try to involve developers (doing a survey or controlled experiments) in this loop.

VI. Conclusion and Future Works

This thesis explores possible ways to support newcomers during the training process. Among the many contributions of this thesis, we propose to support newcomers by (i) *suggesting them appropriate mentors* and improving their program comprehension during maintenance activities with the (ii) *generation of high quality source code summaries* and (iii) the *identification of descriptions which describe source code elements*. As first future direction we plan to enhance CODES adding new features, e.g., for re-documenting classes or packages. We are also currently working on a *development content analyzer* designed to enable a newcomer to *retrieve all feature requests* contained in different communication means in order to plan a set of change activities [24]. Moreover, since such change activities are performed to answer requests of specific users we are also working on a approach which enables newcomer to *retrieve all feature requests* and all the *requests to fix a bug* by classifying user reviews of mobile app [25]. Recently, we conducted a study on code review practises performed by developers of several OSS projects [26] and as (new) future direction we plan to perform further studies to find possible way to support newcomers during code review task. Finally, we also plan to conduct other controlled experiments to explicitly investigate (i) possible relationship between use of documentation and task correctness.

References

[1] S. Panichella, "Supporting Newcomers in Software Development Projects," Ph.D. dissertation, University of Sannio, 2014, http://www.ifi.uzh.ch/seal/people/panichella/DissertationPanichella.pdf.

[2] A. Begel and N. Nagappan, "Global software development: Who does it?" in *Global Software Engineering, 2008. ICGSE 2008. IEEE International Conference on*, Aug 2008, pp. 195–199.

[3] S. Panichella, G. Canfora, M. Di Penta, and R. Oliveto, "How the evolution of emerging collaborations relates to code changes: an empirical study." in *Proceedings of the 36th International Conference on Program Comprehension*, Hyderabad, India, 2014, pp. 177–188.

[4] S. Panichella, G. Bavota, M. Di Penta, G. Canfora, and G. Antoniol, "How developers' collaborations identified from different sources tell us about code changes." in *ICSME 2014. IEEE International Conference on Software Maintenance and Evolution*, 2014.

[5] G. Bavota, G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella, "The evolution of project inter-dependencies in a software ecosystem: the case of apache," in *Software Maintenance, 2013. ICSM 2013. IEEE International Conference on*, 2013, pp. 80–89.

[6] ——, "How the apache community upgrades dependencies: an evolutionary study," *Empirical Software Engineering*, pp. 1–43, 2014.

[7] I. Steinmacher, I. Wiese, A. Chaves, and M. Gerosa, "Why do newcomers abandon open source software projects?" in *Cooperative and Human Aspects of Software Engineering (CHASE), 2013 6th International Workshop on*, 2013, pp. 25–32.

[8] I. Steinmacher, I. S. Wiese, T. Conte, M. A. Gerosa, and D. Redmiles, "The hard life of open source software project newcomers," in *Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE 2014. ACM, 2014, pp. 72–78.

[9] R. Kraut, M. Burke, and J. Riedl, "Dealing with newcomers," 2010.

[10] M. Zhou and A. Mockus, "Does the initial environment impact the future of developers?" in *Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011*. ACM, 2011, pp. 271–280.

[11] A. Hars and S. Ou, "Working for free? motivations for participating in open-source projects," *Int. J. Electron. Commerce*, vol. 6, no. 3, pp. 25–39, 2002.

[12] S. Krishnamurthy, "On the intrinsic and extrinsic motivation of free/libre/open source (floss) developers," *Knowledge, Technology & Policy*, vol. 18, no. 4, pp. 17–39, 2006.

[13] B. Dagenais, H. Ossher, R. K. E. Bellamy, M. P. Robillard, and J. de Vries, "Moving into a new software project landscape," in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010*. ACM, 2010, pp. 275–284.

[14] I. Steinmacher and M. A. Gerosa, "How to support newcomers onboarding to open source software projects," in *Proceedings of Open Source Software: Mobile Open Source Technologies - 10th IFIP WG 2.13 International Conference on Open Source Systems, OSS 2014*, 2014, pp. 199–201.

[15] *The Role of Mentoring and Project Characteristics for Onboarding in Open Source Software Projects*. ACM, 2014.

[16] A. Labuschagne and R. Holmes, "Do onboarding programs work?" in *Proceedings of the 10th Working Conference on Mining Software Repositories, MSR '15*. IEEE / ACM, 2015.

[17] G. Bavota, S. Panichella, N. Tsantalis, M. Di Penta, R. Oliveto, and G. Canfora, "Recommending refactorings based on team co-maintenance patterns," in *Proceedings of the 29th international conference on Automated Software Engineering (ASE 2014)*, 2014.

[18] A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, and S. Panichella, "Using ir methods for labeling source code artifacts: Is it worthwhile?" in *IEEE 20th International Conference on Program Comprehension (ICPC'12)*, 2012, pp. 193–202.

[19] A. Lucia, M. Penta, R. Oliveto, A. Panichella, and S. Panichella, "Labeling source code with information retrieval methods: an empirical study," *Empirical Software Engineering*, pp. 1–38, 2013.

[20] G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella, "Who is going to mentor newcomers in open source projects?" in *Proceedings of the 20th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, 2012, p. 44.

[21] G. Canfora, M. Di Penta, S. Giannantonio, R. Oliveto, and S. Panichella, "YODA: Young and newcomer developer assistant," in *Proceedings of the 35th International Conference on Software Engineering*. IEEE CS Press, 2013.

[22] C. Vassallo, S. Panichella, G. Canfora, and M. Di Penta, "CODES: mining source code descriptions from developers discussions." in *Proceedings of the 36th International Conference on Program Comprehension*, 2014, pp. 106–109.

[23] S. Panichella, J. Aponte, M. D. Penta, A. Marcus, and G. Canfora, "Mining source code descriptions from developer communications," in *IEEE 20th International Conference on Program Comprehension, ICPC*, 2012, pp. 63–72.

[24] A. D. Sorbo, S. Panichella, C. Visaggio, M. Di Penta, G. Canfora, and H. Gall, "Development emails content analyzer: Intention mining in developer discussions," in *Proceedings of the 30th international conference on Automated Software Engineering (ASE 2015)*, 2015.

[25] S. Panichella, A. Di Sorbo, E. Guzman, C. Visaggio, G. Canfora, and H. Gall, "How can i improve my app? classifying user reviews for software maintenance and evolution," in *ICSME 2015. IEEE International Conference on Software Maintenance and Evolution*, 2015.

[26] S. Panichella, V. Arnaoudova, M. D. Penta, and G. Antoniol, "Would static analysis tools help developers with code reviews?" in *22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015*, 2015, pp. 161–170.