Software Process Simulation based on Mining Software Repositories

Verena Honsel, Daniel Honsel, and Jens Grabowski
Institute of Computer Science
University of Göttingen
Goldschmidtstrasse 7, 37077 Göttingen, Germany
{verena.honsel, daniel.honsel, grabowski}@cs.uni-goettingen.de

***Abstract*—Software development processes and their quality depend on many evolutionary factors, such as project size and collaboration of the development team, and development strategies. For project managers it would be valuable to test the interplay of different possible future scenarios in advance, before making decisions about the further process. Our aim is to build such a tool based on agent-based simulation. In this paper, we present our simulation model and our approach in mining software repositories as input for it. The simulation model focuses on system growth, bugs lifetime and developer activity. The detection of patterns related to these topics allows us to estimate adequate simulation parameters close to reality. We evaluate this approach by comparing empirical and simulated data.**

Keywords - ***Agent-Based Simulation; Software Evolution; Mining Software Repositories.***

I. Introduction

The mining of open source software repositories has become a widespread research area in the last years, e.g., Mining Software Repositories (MSR) [1]. The insights gained from repositories of open source projects, i.e., Version Control Systems (VCS), Bug Tracking Systems (BTS) and Mailing Lists, are manifold. Analyzing these is a great help for many software evolution tasks and questions.

For modeling the evolution process of software projects we are concerned with developers and artifacts, e.g., modules, files and classes. These entities have different properties evolving over the time. In the area of software evolution there are two research directions according to Lehman [2]. The first deals with the *how* and the second with the *what* and *why*. Our intention is to understand the what and why and to utilize this knowledge to simulate software evolution.

When examining software evolution we first have to define our way of posing the problem to diagnose the right influencing factors suitable for our investigations. To this aim, we are mainly interested in three factors of software evolution and their triggers: project growth trends, bug appearance, and development strategies. Identified evolution parameters playing a role for this processes are, e.g., the number of artifacts, software changes, bug-fix information, and commit reports.

Lots of work has been done in tracking the evolution of changes [3], system size and complexity growth [4] [5], defect population and handling [6], and developer collaboration [7]. Nevertheless, none of these works put the gained results in context of a simulation for predicting risky trends and evaluating different alternatives. We retrieve history information from VCS and BTS as input for our simulation. From this starting point we select and preprocess the data according to our purpose and finally analyze the data to reveal patterns.

The contributions of this paper include:

- Adaption of data mining techniques for the estimation of simulation parameters.
- The usage of our initial case study of a long-living open source software project, which is K3b [8], for the simulation including a comparison of empirical and simulated data.

We present our work as following: In Section II we explain our underlying agent-based simulation model. Subsequently, we introduce the applied facts extraction and processing methods, as well as the data analysis and how to map these results back to the simulation model in Section III. Finally, we give a conclusion and outlook of our work in Section IV.

II. Background

We based out work on the approach of Smith and Ramil [9] as starting point for our research and we use Repast Symphony [10], an open source agent-based modeling and simulation platform, for employing it. The agent-based simulation runs round based and in each round desired behaviors of all agents are executed. An agent is an individual having dynamic interactions with other agents and its environment that influence its behavior [11]. The environment exposes the living space of the agents. In the approach of Smith and Ramil [9] the developers as active agents live on a environmental grid and work on modules as passive agents. The modules have as attributes fitness and complexity conditioning each other. To the best of our knowledge this is the only resource close to our research topic.

The work of Smith and Ramil [9] can reproduce different aspects of software evolution, e.g., the number of complex modules, number of file touches, and some patterns for system growth with different parameter sets. This makes the simulation very complex and intricate. To reduce the number of parameters, we are working with specialized models. The model presented in this paper covers only the aspects system growth and lifespan of bugs.

978-1-4799-4274-9/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDMW.2014.35

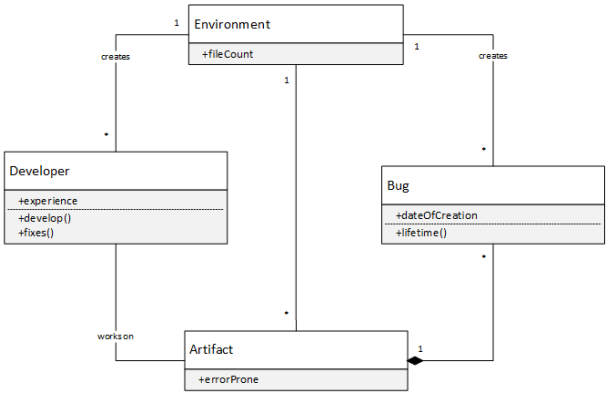

Fig. 1. Simulation model for system growth and the lifespan of bugs.

III. The Approach

We present our approach by first explaining our simulation model and then describing several aspects of the data analysis.

A. Agent-Based Simulation Model

A major difference between our model and the one from [9] is that our model is not grid based. Instead all instantiated entities live in one environment and dependencies between them can be represented as networks. Possible dependencies are file coupling, contributor networks (Figure 5), and the change coupling. This allows us to develop more sophisticated models.

Our model consists of three entities representing agents and additionally the environment as depicted in Figure 1. The only active agent is the developer. When the simulation starts, the environment instantiates a constant number of developers. We assume that each developer can spend the same effort. *Develop* means that software artifacts are created, updated, or deleted. Thus, the system growth depends on the developers work. We assume that the larger the system, the more complex it is. Since the complexity is a limiting factor in productivity [9], the growth is limited by the size of the software under simulation.

Bugs are created by the environment. Each round, a certain number of bugs is generated and assigned to randomly chosen artifacts. If a artifact contains a bug, developers can fix it with a certain probability.

In the following we describe how the simulation parameters can be determined through software mining.

B. Data Retrieval and Analysis

For the facts extraction we used the tool *cvsanaly* [12], which is based on the version control log entries. With the help of *cvsanaly* we can store the necessary history information in a database. We extended this database with information about bugs from *Bugzilla*. This builds the platform for our analysis.

By exploring relationships and habits in the evolution of open source software projects we gained valuable insights. We analyzed K3b, a rather big project with over ten years of software development and over 6000 commits, to which 126 developers contributed in this space of time.

TABLE I
Fixing time of bugs in K3b in days.

Severity	Max	Mean	Std.Dev.
minor	1866	266,6	442,8
normal	2223	300,3	368,2
major	492	239	265,3
crash	1124	174,3	263,5
wishlist	1857	214	319,8
grave	1047	478,7	493,1

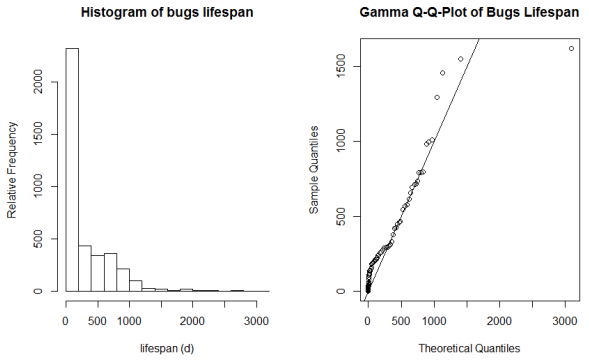

Fig. 2. Bugs lifespan distribution.

1) System Growth Trends: First, we analyzed system growth trends. Godfrey et al. [13] state that the growth in number of source modules usually is sub-linear with a decrease at the end of the development. We observed similar results for K3b and used this to build a statistical model for the growth based on creations, updates and deletions by developers. This is shown in Section III-C.

Empirical Studies showed that also super-linear trends are possible [14], but, with the assumption of a constant number of developers it is not possible to simulate this.

2) Bugs Lifetime: Second, we investigated the lifetime of bugs and their density. We declare a bugs lifetime as the time between the date it was reported and the date is was last modified. From this we gained the statistics displayed in Table I. For each kind we list severity, the maximum, mean and standard deviation. We observe that more critical bugs tend to be fixed sooner.

The distribution of all bugs is displayed in Figure 2. From the histogram, we assume that the lifespan of bugs follow a gamma distribution and compared it by a quantile-quantile-plot. In the simulation model we use this as lifetime attribute for a bug spread by the environment.

3) Developer activity: Third, we are interested in the how and why of the developer activity and collaboration evolution. For this, we created developer-files-networks (without file dependencies) for each year of development extracted from commit log files and visualized the results with Gephi [15]. The evolution of the graph is depicted in Figure 5. These graphs show that there is one main contributor who is the

TABLE II
DEVELOPER-FILE NETWORKS OF K3B.

Year	No. of Edges	No. of Nodes	Modularity
2001	345	312	0,35
2002	1155	712	0,42
2003	1580	1072	0,36
2004	1512	1259	0,26
2005	1613	1356	0,27
2006	2311	2298	0,02
2007	2884	2123	0,26
2008	1477	914	0,38
2009	1337	870	0,36
2010	1347	982	0,30
2011	328	286	0,41
2012	69	71	0,06

project creator. In the graphic he poses always the red node on the right side of the network. Over the years he gained more and more influence peaked in 2006. Afterwards the work gets more balanced and another (blue) developer arises and finally inherits the central status in the project as its maintainer. A similar observation can be found in [16].

The results can also be retraced in Table II. There each year is listed with is corresponding details about the developer-file network. In there the modularity factor in this case gives us a hint on how balanced the work in this year was, because a low factor as in 2006 and 2012, means that the network cannot be modularized in clusters. There the work depends to much on a certain developer. The most balanced network in 2002 results the best modularity factor.

C. Case Study

We used the data retrieved by mining the open source project K3b to parameterize the simulation model. Since the mined projects lifespan took 4044 days, the simulation runs 4044 rounds whereby one round represents one day. In this simulation we are using the average developer with average behavior. This means, that each developer spend the same effort. For the number of files, that will be created, updated, or deleted due to a developers commit, we assume that the number of file changes follow a geometric distribution. We used the version of the geometric distribution were the number of failures before the first success is counted, i.e., zero is allowed. Therefore we have $P(action) = (1-p)^{k-1}p$. From developers point of view a success means that there is nothing to do and zero files are updated. The likelihood of file changes decreases with the increasing system growth. When starting the simulation the likelihood of file changes is also be initialized with mined parameters. From our data mining we inferred values $p_c = max(\frac{\#currentfiles^2}{\#totalfiles}, \frac{1}{1.4})$ for file creation, $p_d = max(\frac{\#currentfiles^2}{\#totalfiles}, \frac{1}{1.2})$ for file deletion, and $p_u = \frac{1}{5}$ for updates as suitable to describe the growth of K3b. This values together with the assumed distribution decides for each step which kind of change takes place. The here presented approach does not include networks with file dependencies, which is part of the future work (Section IV).

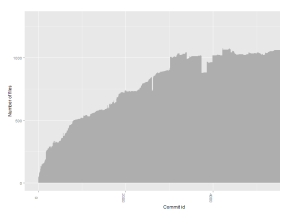

Fig. 3. Empirical system growth of K3b.

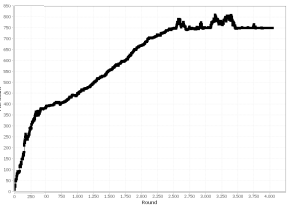

Fig. 4. Simulated system growth.

In Figure 3 the empirical file growth is shown and in Figure 4 the file growth produced by our simulation is pictured. Comparing these one can see that our simulation is able to reproduce the actual file growth of K3b.

IV. CONCLUSION AND FUTURE WORK

In this paper, we presented our agent-based simulation model for software evolution and an approach to gain suitable knowledge to estimate suitable input parameters. Thereby we determined how to model system growth, defect occurrence and activity of developers and suit parts of it to the simulation model. By comparing empirical and simulated data we were able to confirm these observations. For future investigations we plan to find out, if there are any developer strategies traceable and what impact they have on the other parameters such as the evolution of defects. Since the networks mentioned in Section III-A are currently under development, it is also one of the next steps to improve them.

The case study of K3b revealed some interesting facts as described in Section III-C, but other projects may give other conclusions. We currently apply our approach to other projects to reach a better comparability, find common triggers of software and developer behavior with the aim to finally enrich our simulation.

In addition and to guarantee the significance of the approach in Section III-B3, Tymchuk et al. [17] suggested to ask developers about their collaboration because developer networks obtained from VCS may not reflect the actual situation. For further investigation and integration into the simulation we are interested to find out the triggers for this development and possible relations to other observed behavior. We also plan to distinguish between core developers and others, which means to have different kinds of agents.

Furthermore, we are working on a realistic simulation of the lifespan of bugs according to the mined data of K3b combined with developer fixing strategies and on adapting our simulation model described in Section III-A tailored to that purpose. Because of using specialized models for different research questions we can adapt or extend it for different scenarios.

ACKNOWLEDGMENT

The authors would like to thank the SWZ Clausthal-Göttingen, which partially supported our work, as well as

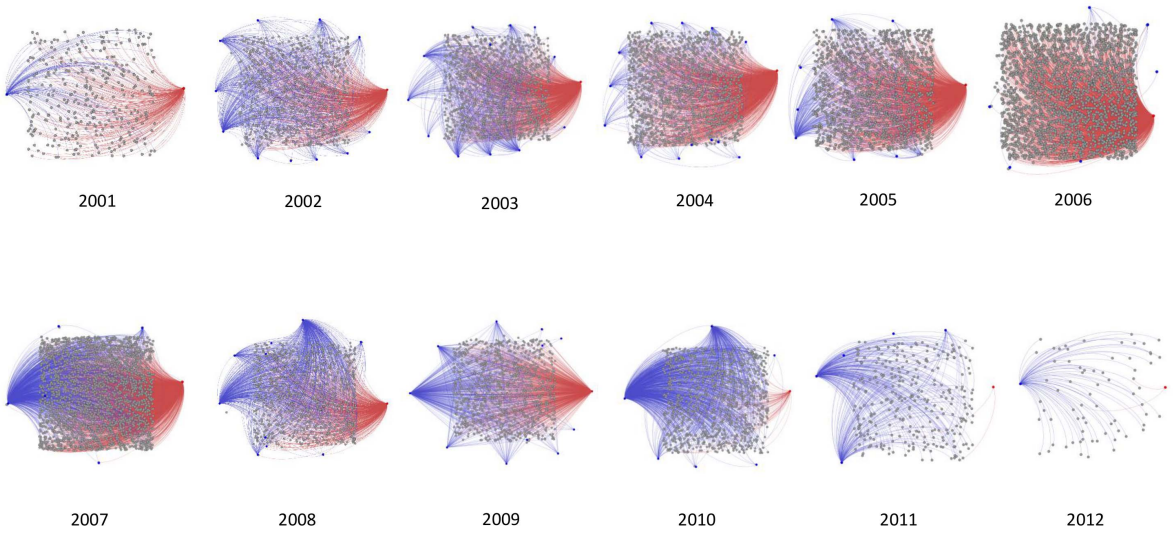

Fig. 5. Yearly developer-file networks of K3b.

Stephan Waack and Marlon Welter for fruitful discussions.

REFERENCES

[1] M. Di Penta, "Mining software repositories (msr)," online, 2014. [Online]. Available: http://2015.msrconf.org/

[2] M. Lehman and J. Ramil, "Towards a theory of software evolution - and its practical impact (working paper)," in *Invited Talk, Proceedings Intl. Symposium on Principles of Softw. Evolution, ISPSE 2000, 1-2 Nov*. Press, 2000, pp. 2–11.

[3] R. Robbes and M. Lanza, "A change-based approach to software evolution." *Electr. Notes Theor. Comput. Sci.*, vol. 166, pp. 93–109, 2007.

[4] G. Xie, J. Chen, and I. Neamtiu, "Towards a better understanding of software evolution: An empirical study on open source software." in *ICSM*. IEEE, 2009, pp. 51–60.

[5] M. W. Godfrey and Q. Tu, "Evolution in open source software: A case study," in *Proc. Int'l Conf. Software Maintenance (ICSM)*. Los Alamitos, California: IEEE Computer Society Press, 2000, pp. 131–142.

[6] M. D'Ambros and M. Lanza, "Software bugs and evolution: A visual approach to uncover their relationship." IEEE Press, 2006, pp. 227–236.

[7] Q. Hong, S. Kim, S. C. Cheung, and C. Bird, "Understanding a developer social network and its evolution." in *ICSM*. IEEE, 2011, pp. 323–332.

[8] S. Trueg, "K3b-the cd/dvd kreator for linux," online, 2011. [Online]. Available: http://www.k3b.org/

[9] N. Smith and J. F. Ramil, "Agent-based simulation of open source evolution," in *Software Process Improvement and Practice*, 2006, pp. 423–434.

[10] M. J. North, N. T. Collier, J. Ozik, E. R. Tatara, C. M. Macal, M. Bragen, and P. Sydelko, "Complex adaptive systems modeling with repast simphony," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, pp. 1–26, 2013. [Online]. Available: http://dx.doi.org/10.1186/2194-3206-1-3

[11] C. M. Macal and M. J. North, "Tutorial on agent-based modelling and simulation," *Journal of Simulation*, vol. 4, no. 3, pp. 151–162, 2010.

[12] C. G. Campos, "Cvsanaly," 2014. [Online]. Available: http://metricsgrimoire.github.io/CVSAnalY/

[13] M. Godfrey and Q. Tu, "Growth, evolution and structural change in open source software," September 2001.

[14] G. Robles, J. Amor, J. Gonzalez-Barahona, and I. Herraiz, "Evolution and growth in large libre software projects," *Author Index*, pp. 165–174, 2005.

[15] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," 2009. [Online]. Available: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154

[16] M. S. Zanetti, I. Scholtes, C. J. Tessone, and F. Schweitzer, "The rise and fall of a central contributor: Dynamics of social organization and performance in the gentoo community," *CoRR*, vol. abs/1302.7191, 2013.

[17] Y. Tymchuk, A. Mocci, and M. Lanza, "Collaboration in open-source projects: Myth or reality?" in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: ACM, 2014, pp. 304–307. [Online]. Available: http://doi.acm.org/10.1145/2597073.2597093