Empirical research into open source evolution – should we adjust the agenda to what seems possible?

Conference or Workshop Item

Empirical research into open source evolution – should we adjust the agenda to what seems possible?

Submitted to MEFLOSS 2008, Beijing, China, 3-Oct-2008

Juan Fernández-Ramil
Computing Dept., The Open University,
Walton Hall, Milton Keynes MK7 6AA, U.K. j.f.ramil@open.ac.uk
and
Institut d'Informatique, Université de Mons-Hainaut,
6 Avenue du Champ de Mars, Mons, B-7000, Belgium, j.f.ramil@umh.ac.be

Introduction

The quantitative study of software evolution seem to have kicked off in the early 1970's with the studies of Manny Lehman and Les Belady of "large program growth dynamics" in the IBM 360-370 operating system. The phenomenon studied was subsequently called *software evolution*. Their work led to the now eight Lehman's laws of software evolution and other insights such as the SPE program classification [1]. During its first 30 or so years, the topic of empirical studies of software evolution was pursued almost exclusively by Lehman and other few investigators (e.g. [2]) and it was limited to the analysis of metrics data from proprietary systems. Open source has opened up this topic to almost any interested researchers with Internet access, knowledge of the data extraction tools and sufficient computing power, not just for those with contacts in industry. This is because open source offers unlimited access to code and other to other artifacts and researchers are free to disclosure what they find (this not, by the way, the same as to say that open source researchers should be free from ethical considerations). Open source has triggered an increasing interest in empirical studies of software evolution, as reflected in the growing number of empirical studies of open source evolution being published. The MEFLOSS workshop gives also testimony to this interest. With more than 10 years involvement in software evolution research, I ask myself whether the actual possibilities of conducting thorough and solid scientific work in this area will match our current expectations (see, for example, the impressive list of, to me, difficult issues present in MEFLOSS'08 call for papers). Without any doubt, there are big opportunities (e.g. data availability) for research into this field. There are, however, big challenges, some of which are, to my view, not sufficiently addressed. I would go further and argue the following: research in this area is still in its infancy. The most serious challenges need to be systematically tackled, if we are going to achieve one day an understanding of the evolution of software similar to that achieved today in traditional engineering disciplines like mechanical or electrical engineering.

Some context

In order to provide the context of the present position paper, let's consider the following definitions or assumptions. An empirical study consist of investigating a falsifiable hypothesis(es) in the real world by measurement, analysis of data, exploring different explanations and interpretation of the results. The hypothesis(es) need to be relevant to the physical world. Software is not 'physical' until it is installed on a computer. However, software is 'information' that relates to the realm of physical entities (in strict sense, software reflects abstractions and it models aspects of physical reality). Software, seen as a meta-physical object, can be investigated through axiomatic, rather than naturalistic scientific methods, and this is what researchers in formal methods do. However, a naturalistic study requires measurements or real world collected data. The measurements or more generally, observations, may come either from a controlled experiment (unfortunately, almost never the case for empirical studies of software evolution) or from observations of phenomena that we did not intent or cannot control. We exclude all the hypotheses that can be tested by analytical means. This is to say, that hypotheses such as "Is true that 2+2 is 4?" that can be proven mathematically are not in the domain of empirical studies. Simulation-based studies are possible as part of empirical studies, even thought the empirical validation of simulation models is not likely to be easy and can raise many questions.

We conduct our empirical studies in the context of software engineering and our interest is (or should be) how to better understand the process of evolving software as to achieve greater levels of stakeholder satisfaction at lower risks and costs. As professionals, we want to abide always by the ethical principles we profess, including the ethical principles laid out by institutions like the BCS or the IEEE to which we may belong and by any other affiliations of us. This should include the highest respect for the developers, users and any others involved in the studies, including their well-being during and after our study.

The above provides the wider context of the studies I am referring to. In practice, generating and expressing hypothesis that are 'well-formed' for a scientific investigation might not be easy, as almost any empirical researcher has realised. As someone must have said: "to ask the right question is the first step towards truth". Just to briefly give an example, one hypothesis often studied in empirical studies of software evolution has been whether software's functionality (i.e. its size) grows over releases or not. This hypothesis relates to Lehman's sixth law of 'continuing growth'. A subsidiary question is whether the growth seems to follow a particular trend such as linear (or growth proportional to time or releases), sub-linear (positive growth but at a declining rate) or super-linear (at an increasing rate). A practical example of how to investigate such hypothesis is given in [3]. Having set out the context, let's us briefly consider the opportunities and the challenges in this type of research.

The opportunities

On the practical side, the free access to open source data offers 'the' big opportunity. This together with the increasingly advanced computational resources and tools for sharing, storage and analysing of raw and extracted data is opening up new possibilities for researchers and explaining the current up-trend in publications and research events in this field. Meanwhile, as part of the wider world, software is ever more pervasive and increasing aspects of human life depend on software that is appropriately evolved. As the real world rate of change seems not to decrease, the same applies to software. Hence, there is a big opportunity for empirical researchers of software evolution both to study an ever increasing number of software systems and their processes of change. There is also a growing demand for knowledge and understanding that should help open source communities to be more efficient, effective and professional.

The researchers, particularly those that relatively recently have started to look into this area, are building communities (e.g. ERCIM Working Group on Software Evolution, Mining Software Repositories, now MEFLOSS) which put together an impressive array of skill, talents and resources to tackle the research questions. So far, so good!

The challenges

Now, let's consider the 'not so good' news: in my view, there are a number of important obstacles for progress in empirical studies of software evolution. I focus here on the obstacles that are above and beyond those that apply to software engineering in general [4] and to software engineering empirical research in particular [5]. The challenges below are additional to those already discussed in [6].

E-type software[1] is always part of wider systems. Software, itself, as a model of the real world is a mirror of such a real world. The question arises is whether the patterns we observed in software evolution are a property of software or are simply a reflection of the dynamics of the domains in which the software is 'embedded'.

Despite the apparent large amount of raw data, there are many basic important attributes, such as effort, that are difficult to measure in an accurate way. We can still count the number of people involved in a particular project, but it is difficult to get more precise than that. Hence, some of our basic questions related to productivity may not be properly answered. Hence, it is difficult to compare, even in a very basic way, efficiency of methods and tools. There are also a myriad of other important attributes (e.g. developers' thinking processes, motivation, experience and background, informal interactions) that seem impossible to measure unless one conducts specific surveys or organise specific data gathering activities. There are also problems of missing data and inconsistencies or gaps between related, but independently updated artefacts or repositories (e.g. typical problem is to try to match defect data to code evolution data).

We do not seem to have a set of competing theories that can be empirically assessed and that we could use to encapsulate the gained knowledge and understanding. The laws of software evolution are perhaps the closest we have got to such a theory. But to a large extent they are rooted in the needs and concerns of the 1970s. Moreover, there isn't, to my knowledge an agreed way of empirically evaluating the laws of software evolution, making it even more difficult to compare any two studies. This also gives the impression that, as a community, we are constantly building on moving sand.

Software technology is changing at rapid speed. When we thought we had a limited understanding of monolithic bespoken software systems built in procedural languages by teams with the same organisation (e.g. the laws of software evolution) , we had the irruption of the object-orientation, the Internet, component-based development and similar "disrupting" technologies. As empirical researchers we need to recognise that our understanding seems to be lagging behind and challenged by new "developments".

In search for insights, researchers tend to look into other disciplines (e.g. Biology) in order to provide concepts or explanations for software evolution. For example, Lehman considers software evoluiton as the the fruit-fly of artificial, i.e. human-made, systems, whose study may provide insights into the evolution of other artificial systems [1]. One such borrowing is the idea of considering open source as an *eco-system* of interacting communities. It seems to me that the term *eco-system* [6]. With hindsight, it seems that the term does not do justice to the fact that such ecosystem is formed by persons, capable of abstract thinking, and self-consciousness. Whatever is going on in an open source community, is of a different level than the exchanges of food and energy that occur in a natural biological ecosystem. As in software development, in empirical research we struggle to find the right concepts but perhaps we need a better concept. This leads me to the last, but never least, is the role and impact human component in the software evolution process. We know that software evolution is done by people and for people. People are self-conscious and capable of knowing, they exercise self-determination and continually learn, react, adapt to and modify the environment. The most important part of any software process is the people and yet we understand so little in terms of what motivates or how to best support them. Sadly, the laws of software evolution do not seem to say anything about the human component conducting the evolution of the software. The

[1] By default, empirical researchers look at E-type software, that is programs that solve a problem or address an application within a real world domain. This software belongs to the real world physical world, in contrast to software like S-type programs, which are seen as pure mathematical entities.

laws seem to suggest that contributors are 'trapped' in the labyrinth of process feedback loops, accumulating change requests, increasing complexity, out-of-date documentation and 'slow' communication paths. The concept of eco-system seems to embody some similarities. But our experience is that people make a difference: would Linux have been possible without Linus Torvalds? Our attempts to model the evolution characteristics of software using mathematical models of growth (e.g. [3]) may be seen as forgetting that people are the most important component. Future laws and theories of software evolution need to give people their deserved first-class status. How any empirical laws 'discovered' in systems that involve human are compatible with our experience of freedom and self-determination seems to be an important open question [7]. I believed that all this requires a profound shift in the way of thinking about our research aims and questions. Should we, for example, try to study more carefully how human creativity, ingenuity and endeavour, and appropriate management of human resources or of a community, is behind the software evolution success stories? This may relate not only to the detailed technical problem-solving aspects, but also to find out how fairness and good-will can be promoted though an appropriate set of rules, governing a particular open source community. This seems to require qualitative [8] than quantitative methods of research.

The way forward

As scientists we share faith in reason and in its derivatives, such as the scientific method, that they will lead us to the truth in our particular domain of enquiry. The 19th and 20th century progress in the traditional engineering disciplines, propelled by the scientific method, have been impressive (e.g. mechanical and electrical engineering). The scientific method has not been so successful in other domains (e.g. social science, weather prediction, climate change) and one has to recognise its limitations. Some of the limitation are not in the method, but in the data and the nature of the entities of study. This position paper has mentioned some of the obstacles that are present when empirically studying open source evolution. It argues that we are in the early days of anything closer to a scientifically-backed engineering discipline of software evolution, including open source evolution, but humble also were the beginnings of the other engineering disciplines. Perhaps we cannot even imagine the shape and form that successful software engineering theories will have in the next decades. In my view, we need to dignify, so to speak, the human role in driving and implementing software evolution. This, itself, may call for a deeper understanding of the relationships between people skills, tool support, the wider social and community support and when and where the scientific method can provide us with answers. Furthermore, the software evolution phenomena itself may need to be abstracted and aggregated in different ways, perhaps not yet imagined yet, as for researchers to be able to find out and explain any regularities and patterns. The data analysis tools and modelling formalism may still need to be invented or 'borrowed' from other disciplines.

Reason will find the way to truth as light shines through the dark. The large number of obstacles should actually encourage us because they indicate that the room for further research and improvement over previous research is enormous. For the time being, however, we should keep our expectations modest and close to what can be achieved. What is achievable may not seem to explain a lot, but if is tackled well and systematically may provide solid foundations for the future. Let's keep our research scope small, so that others may become giants when sitting upon our shoulders.

Note

Due to other commitment in Beijing on the same day, the authors cannot be physically present at the MEFLOSS workshop. In case of selection for presentation, A. Capiluppi has kindly offered to present this paper.

Acknowledgements

The author gratefully acknowledges the Belgian FNRS for funding of his research at UMH during which this position paper was written.

References

[1] Lehman M.M, Belady L.A. (eds), Software Evolution-Processes of Software Change, Academic Press, London, 1985. Available from ftp://ftp.umh.ac.be/pub/ftp_infofs/1985/ProgramEvolution.pdf

[2] Kitchenham B., System Evolution Dynamics of VME/B, ICL Technical Journal, May, 1982, pp. 42-57

[3] Mens T., Fernández-Ramil J., Degrandsart S., The Evolution of Eclipse, Proc. ICSM 2008, Sept 28-Oct 4, https://fichiers.umh.ac.be/download.php?id=uRVTI0Jc1Ja7Ipuk#icsm2008manuscript#RP-25_Mens_T.pdf

[4] Brooks F.P., No Silver Bullet – Essence and Accident in Software Engineering, Computer 20(4), April 1987, pp. 10-19.

[5] Kitchenham B.A., et al, Preliminary Guidelines for Empirical Research in Software Engineering, IEEE Transactions on Software Engineering, 28(8), 2002, pp. 721-734

[6] Fernández-Ramil J., Lozano A., Wermelinger M. and Capiluppi A., Empirical Studies of Open Source Evolution, ch. 11 in Mens T. and Demeyer S. (eds), Software Evolution, Springer, 2008, pp. 263-288.

[7] Dorato M., The Software of the Universe- An Introduction to the History and Philosophy of Laws of Nature, Ashgate, Aldershot, England, 2006, 158 pp.

[8] Seaman C.B., Qualitative Methods in Empirical Studies of Software Engineering. IEEE Transactions on Software Engineering, 25(4), 1999, pp 557-572.