PhD offer 2016-2017
Advisor : Pr. Xavier Blanc - LaBRI

Title : Evolution of web applications

Short presentation:

Preamble

The success of Internet has provoked a mess in software maintenance. All applications that are designed to run on the Internet must not only fulfill all user requirements, which are more and more frequent and heterogeneous, but also must support the Internet evolutions. Further, they have to evolve at the Internet speed for not being deprecated and then unused.
The main problem is that internal factors that drive the maintenance of traditional applications, such as code quality for instance, becomes unpromising in front of external factors that broadly consider the evolution of Internet.
This proposal addresses Internet speed software evolution. It aims to define factors and facilities that will drive the maintenance of applications designed to run on the Internet. For that purpose, it uses source code analysis coupled with statistical measures of large set of applications with the objective to exhibit evolution trends, and to leverage on them.

General description

All Internet Applications must evolve at the Internet speed. They have to follow the rhythm of both users' requirements and technical evolutions of the Internet. Moreover, if an Internet application does not evolve, it quickly becomes deprecated and then unused.

While traditional applications are considered to be of a high quality when they are stable, an Internet application that does not change is doomed!

An Internet application must evolve at the Internet speed. It must always fulfill the new users' requirements and it has to be up to date regarding Internet evolutions.

Tradition approaches that address Software Evolution cannot deal with the Internet speed [BL+03]. Indeed evolutions that Internet application must handle are always due to external factors impossible to control neither in their rhythm nor in their importance. With traditional approaches, evolutions are first analyzed to measure their feasibility, while with Internet application, the speed of evolutions makes such analysis impossible. Besides quite often it is the market pressure or even the technical pressure of Internet that impose them.

Companies such as CDiscount have then to handle a large number of evolutions without being able to control them. This is probably one of the reasons why there are too many software failures and why maintenance cost still explodes [FBF+07].

This proposal addresses the maintenance of Internet applications (Internet Speed Software Evolution). The proposal is to tame the evolution of Internet applications by identifying evolution as soon as possible and by providing metrics for measuring their significance and their impact.

This proposal is based on a statistical analysis of large sets of Internet applications. The idea is to identify evolution trends that can be exploited to overcome Internet application

maintenance problems. To identify evolution trends, we aim to observe, classify and measure the evolutions that have been already realized by existing Internet applications. Hence when an Internet application will have to evolve, it will be possible check if there is an existing identified trends that match its specific context. If such a trends has been identified, then it would be possible to anticipate the effects of the evolution.

This kind of approach is already used by the mining repository community. For instance it has been shown to provide interesting results for fault identification [NBZ06]. Further, we have shown that trends of software library migration can be identified and qualified [Te 12].

This proposal breaks with classical approaches that deal with software evolution since they are all centered on a single application instead of considering a large set of them. Classical approaches are looking for internal metrics with the intent, for example, to identify fault prone components. These approaches do not consider external factors such as the evolution of Internet. On the contrary this approach targets external factors and aims to provide a global model that represents trends of a large number of applications.

Concretely this proposal aims to provide a reactive approach that gives solutions to evolutions that Internet applications should realize. The goal is to define indicators to tame internet evolution. These indicators are computed thanks to a statistical analysis of Internet applications.

This proposal will answer to the following questions:

- How to identify changes in Internet technologies?
    - By observing the modifications that are done on a large set of Internet applications, we can exhibit trends and hence identify Internet technologies that require changes in the applications. Moreover we can also identify Internet technologies that should not be adapted as they are only buzz.
- How to measure and reduce the cost of internet evolution?
    - By identifying similar Internet applications (either regarding their functionalities or by looking at their architecture) it is possible to plan the evolution cost of an application. We can also think about adapting the evolutions that have been already done to existing applications in order to reduce the maintenance cost.

Scientific dimensions

Temporal model for evolution: A temporal model for evolution is absolutely mandatory to compare the evolution of several Internet applications. An evolution has not the same impact on a young application than on an older one. However, the concept of time is not so defined in software evolution. Clock time is used to measure development cost but who can say that an application is young if it has been started few months ago? What if a huge development has been done on it? If we now look at source code repository, the order of commits can also be considered to be a temporal model[Con00]. The order of commits is interesting as it shows the developers vivacity, the order between different revisions but also all the parallel branches of development. The problem is that this model depends on the behavior of the developers and hence does not really express the age of an application. We can also consider the CHURN, a measure that sums all modifications done to software artifacts[NB05]. However this measure has a bias as it depends on programming languages. Finally, no one of these temporal models consider the dynamic of teams of developers neither the age of external dependencies. They are therefore not adequate to measure the age of an Internet application. I therefore aim to provide a temporal model for evolution with

the intent to measure the age of an Internet application. This model should take into account the development effort but also the frequency and the importance of the modifications done to the application. Moreover, it should support intra and inter temporal dependencies between applications (update synchronization, branches, etc.).

Syntaxical level of abstraction: The evolution of an Internet application can be observed syntactically at different levels of abstraction. At the lower level, an application can be considered to be composed of a finite set of software artifacts (source code, documentation, bug reports, etc.). As any software artifact has a syntactical representation, an evolution can be represented as a syntactical rewriting [CCP07]. However, the difficulty of such a level lies in the complexity and the heterogeneity of the software artifacts. Moreover, the well-known problem of the origin analysis, which aims to track syntactical elements during successive evolutions, has to be tackle to perform impact analysis [FW+07]. At higher levels of abstraction, we can observe evolution at the programming language level, at the programming style level or even at the architecture level. Further, some research approaches aim to express evolution at higher levels by abstracting lower levels [ND 04]. The higher level, the better the observation and the analysis. I therefore aim to support higher levels of abstraction. Currently Pr. Xavier Blanc, the supervisor of this proposal, already support the lower level of abstraction (https://code.google.com/p/harmony/) for source code artifact (mainly Java). He also designed an approach that addresses the problem of origin analysis [ASE14]. Further, Cédric Teyton, a PhD student he advised, addresses the evolution of libraries for Java.

Statistical analysis: This proposal uses statistical analysis on large sets of Internet application. Unfortunately, statistical methods have been poorly used in Software Evolution as there are many issues to face [BW+00]. The first issue consists of disposing of a large set of entities to observe. Hopefully this is less and less an issue as there are many applications available on the Internet and as more and more industrials open their doors and let academics performing their studies. The second issue lies in building samples. To build a sample without any bias either the universe of entity has to be fully available or the distribution law has to be known. No one of these two conditions holds for software evolution. To overcome this major issue, sampling techniques based on a reject probability can be used. I already make some tries but without any success (to many entities were rejected). The third and last issue consists in choosing the adequate statistical methods to compute analyzes. Even if there are some approaches that use statistical methods in software evolution, no one of them uses a temporal model. As a consequence, this proposal aims to propose a sampling method for Internet applications accompanied with a set of statistical methods to perform analyzes on their evolution.

Réferences

[ASE-14] J-R Falleri, F Morandat, X Blanc, M Martinez, M Monperrus: Fine-grained and accurate source code differencing. ASE 2014, pp313-324

[BL+ 01] Richard. Baskerville, Linda. Levine, Jan. Pries-Heje, Balasubramaniam. Ramesh, Sandra. Slaughter. 2001. "How Internet Software Companies Negotiate Quality" IEEE Computer, Vol 34, Issue 5, May 2001.

[BL+ 03] Richard. Baskerville, Linda. Levine, Jan. Pries-Heje, Sandra. Slaughter. 2003. "Is Internet-Speed Software Development Different ? " IEEE Software, november-december 2003, pp70-77

[Bo 83] Barry. W. Boehm, "Seven Basic Principles of Software Engineering" Journal of Systems and Software (JSS), vol. 3, no. 1, Mar. 1983, pp.3–24.

[BW+00] L. C. Briand, J. Wüst, J. W. Daly, D. V. Porter. Exploring the relationships between design measures and software quality in object-oriented systems. Journal of Systems and Software 51(3): 245-273 (2000)

[BWD+ 00] Lionel Briand, Jurgen Wust, John Daly, Victor Porter, "Exploring the relationship between design measures and software quality in object-oriented systems", Journal of System and Software, vol 51, num 3, pp 245-273, may 2000.

[CCP 07] G. Canfora, L. Cerulo, and M.D. Penta, Identifying Changed Source Code Lines from Version Repositories, Proc. Int'l Workshop Mining Software Repositories, p. 14, May 2007.

[Con 00] R. Conradi, B. Westfechtel. Version Models for Software Configuration Management. ACM Comput. Surv. 30(2): 232-282 (1998)

[Cu 00] Hillman. Curtis "Flash Web Design", March 2000, ISBN 0735708967

[FBF+ 07] Steven Fraser, Frederick P. Brooks Jr., Martin Fowler, Ricardo Lopez, Aki Namioka, Linda M. Northrop, David Lorge Parnas, and Dave A. Thomas. "no silver bullet" reloaded: retrospective on "essence and accidents of software engineering". In Companion to the 22nd Annual ACM SIGPLAN Conference on Object- Oriented Programming, Systems, Languages, and Applications, OOPSLA 2007, October 21-25, 2007, Montreal, Quebec, Canada, pages 1026-1030, 2007

[Fi 00] Roy. T. Fielding, 2000. "Architectural Styles and the Design of Network-based Software Architectures," Doctoral dissertation, University of California, Irvine.

[FW+07] B. Fluri, M. Würsch, M. Pinzger, H. Gall. Change Distilling: Tree Differencing for Fine-Grained Source Code Change Extraction. IEEE Trans. Software Eng. 33(11): 725-743 (2007)

[Lu 08] Mircea F. Lungi, 2008. "Towards reverse engineering software ecosystems" International Conference on Software Maintenance, 2008, pp.428-431.

[Ma 01] Alan MacCormack, 2001. "Product-Development Practices That Work : How Internet Companies Build Software" MIT Sloan Management Review, pp75-84

[MS 05] David G. Messerschmitt & Clemens Szyperski, 2005. "Software Ecosystem: Understanding an Indispensable Technology and Industry," MIT Press Books, The MIT Press, edition 1, volume 1, number 0262633310.

[NB05] N. Nagappan, T. Ball. Use of relative code churn measures to predict system defect density. 27th International Conference on Software Engineering (ICSE 2005), 15-21 May 2005, St. Louis, Missouri, USA, pp284-292

[NBZ 06] N. Nagappan, T. Ball, A. Zeller. Mining metrics to predict component failures. 28th International Conference on Software Engineering (ICSE 2006). pp452-461

[ND 04] O. Nierstrasz, S. Demeyer. Object-Oriented Reengineering Patterns. 26th International Conference on Software Engineering (ICSE 2004), 23-28 May 2004, Edinburgh, United Kingdom. pp734-735

[Te 12] C. Teyton, J-R Falleri, X. Blanc Mining Library Migration Graphs. 19th Working Conference on Reverse Engineering, WCRE 2012, Kingston, ON, Canada, October 15-18, 2012. pp289-298

[Ti 08] S. Tilley, "Ten years of web site evolution." In: 10th IEEE International Symposium on Web Site Evolution. IEEE Computer Society, 2008

[WLR 11] R. Wettel, M. Lanza, R. Robbed, "Software systems as cities: a controlled experiment." Proceedings of the 33rd International Conference on Software Engineering, ICSE 2011, Waikiki, Honolulu , HI, USA, May21-28, 2011, pp551-560