Improving comprehension and cooperation through code structure

Andrea Capiluppi[1]
Department of Computing and Control Systems
Politecnico di Torino, Italy
Andrea.Capiluppi@polito.it

Abstract

Defining a relationship between a software system's architecture and the process' efforts is one of the most fascinating questions of software engineering. Apparently, when a system's architecture is complex, the process to improve and evolve it will be more difficult. We try to tackle this question from a different point of view: given an Open Source system, in all the phases of its evolution, we focus on both the aspects of software developers, and the obtained software product. More we observe one of the possible architectures of this system, based on the tree structure derived from source components.

First conclusions show that some patterns of tree evolution are recognizable: some branches may appear more promising than other, and are extensively evolved, while other remain in the same status for all the life cycle. More, when the tree structure reaches some status, the process of joining as a core developer seems to forestall.

1. Introduction

Open Source software usually comes in different flavors: in some cases, software systems are claimed to be Open source as long as they release their source code under an open license. In other cases, software systems not only give unrestricted access to the source code, but details on the development process are available too (in forms of how many developers are currently writing code, the CVS access logs, and so on).

Albeit the first case has an appeal in commercial environments, to gain momentum and possibly communities feedback, open process-based OSS systems are

In this paper we use metrics-based data derived from a pilot Open Source software (OSS in the following) system, in order to understand how its internal structure evolves.

We are interested basically to observe source code structure and its changes, to learn from a long-lived OSS system what types of structural changes are more frequently brought to the source code, and to seek for patterns in those changes. For accomplishing these goals, we observe levels of source folders: with this we mean any directory in the code repository which contains source files.

2. Related work

Empirical studies on software development gained momentum after pioneer work of Lehman and his collaborators on OS/360 system [Belady & Lehman, 1976]. Lehman and Belady 's first work [Belady and Lehman, 1976] on software evolution observed 20 releases of the proprietary OS/360 operating systems. From that work, and subsequent studies of other proprietary commercial software [Lehman, 1980], [Lehman and Belady, 1985] [Lehman et al, 1987], [Lehman et al, 1988], the SPE software classification, a set of laws of *E*-type software evolution, now eight, were formulated and refined. Their findings in the seventies and eighties were revisited and supplemented in the recent FEAST projects [Lehman et al., 1998].

OSS systems have an edge over commercial ones when it comes to availability of data: many studies were done since initial efforts on the Apache web-server and Mozilla browser [Mockus et al.].

New studies are both based on individual projects [German], [Koch], [Aoki et al], [Barahona et al.], [Stamelos et al.], [Godfrey & Tu], and on several systems [Capiluppi 2003]

Interesting new approaches to study the evolution phenomenon are given through both quantitative [Antoniades et al], and qualitative [Ramil & Smith] studies on software evolution.

3. Rationale

When investigating code structure of various OSS systems, one may be faced with different patterns of modifications: if we consider code structure from the perspective of code tree (one example is depicted in Figure 1), it's possible to visualize basic components (source files, source folders) as composing a tree, the root of the tree being represented by the parent folder. When analyzing software evolution in a tree perspective, one distinguishes two dimensions:

1. *vertical growth,* that is, creating a sub-branch in an existing branch (upper part of Figure 1)
2. *horizontal growth*, that is, adding a new branch over an existing, common, branch (lower part of

1A. Capiluppi is currently visiting the Department of Maths and Computing in Open University, Milton Keynes, UK

Figure 1)

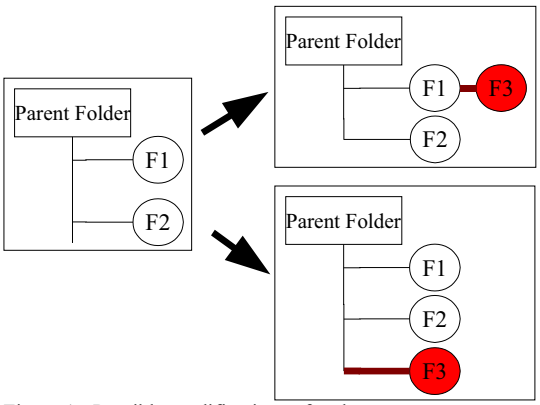


Figure 1 - Possible modifications of code structure

Our study tries to address the evolution of a particular OSS system, based on its structure and developers evolution. We are likely to investigate if there's any link between what happens from a process' perspective (how any developers join in), and the perspective of structural changes.

4. Definition of attributes

4.1. Source code

In this study we measured source code size in three different forms (LOCs, SLOCs and Kbs of code).

4.2. Code Structure

4.2.1. Code components

We deal with source code size in three different forms:

1. *source files*, as to say, all files that match some well-known source code mapping (e.g., *"*.c"*)
2. *source folders*, as to say, macro-components containing at least one source file.
3. *source levels*, as to say each level in the code structure where topologically folders may be placed (folders level)

In some sense, these components form together a code structure which may be interpreted as an architectural view of the system. Based on an open source tool for displaying graphs [Graphviz], we show the evolution of source tree, when intended as composed of files, folders and levels.

In Figure 5, Figure 6 and Figure 7, three diverse stages of evolution in the ARLA system are displayed. The first of these, namely Figure 5, we extracted the tree structure by parsing source components, and automatically feeded the GraphViz tool, in order to display the graph of the first release of this project. Each ellipse represents a source folder. Each connection defines a relationship parent-child through components: labels on connectors represent the amount of files contained in the lowermost folder (the connector arla-0.0 -> appl displays 6, because the 'appl' folder contains 6 files). In this graph, we see that the tree structure is quite defined, further evolution doesn't revolutionary the basic structure (Figure 6 and Figure 7). Since this system involves different platforms in the first instance (linux, bsd, solaris, sun etc.), we suspect that evolution may appear basically on porting new architectures, as well as pushing new functionalities. Figure 6 shows the tree structure when observing the break point observed in Figure 2. A new subtree is added to the basic tree (the red semicircle), as well as new ports for new architectures are added (red rectangles). In Figure 7 we describe the last available release at the moment: a major enhancement is depicted in leftmost red rectangle, while minor enhancements are visible in other parts of the graph. That could mean an increased difficulty in growing the system, as well as a stagnation in both feedback and new functionalities requests (as authors don't grow, in Figure 4)

4.3. Developers

What we are interested in, is OSS software development when it's started by a single person and/or a core of developers, with others getting involved later on. Some studies have published distributions of new developers as the system evolves, even though no common statistical distributions have been found yet [Barahona et al, 2002].

In order to maintain a similar notation to [Mockus, 2002], we should distinguish among developers those who contribute with regularity from those who simply gave code once. We therefore posed as *authors* those developers who submitted code more than once, and in more than a single module/file. We call as *contributors* those developers who contributed code only once, or more than once to the same module/file. Since ARLA ChangeLog provides a very regular and parseable example of keeping tracks of change, we could distinguish which change was done by what developer in which module/file.

5. Pilot Project: The ARLA System

The ARLA project made available its first public release in February 1998, and its most recent release is labeled 0.35.12 (February 2003). 35 major releases were developed, 62 total releases are made available through their web sites, which then included 27 minor releases. ARLA's main purpose is to achieve similar functionality as the IBM AFS file system. It is likely that ARLA has currently achieved even more functionalities than AFS. Its application domain is distributed file systems management, a domain in which a lot of knowledge is available and openly shared. In this respect, this system is similar to flagships OSS successes (Linux, Apache, etc.). In

ARLA's evolution there have been two basic ways of enhancing and evolving the system: adding common features for the system (e.g. supporting of specific network protocols), and adding ports to different architectures.

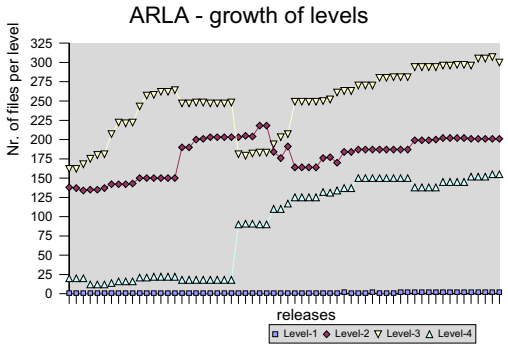


Figure 2 - ARLA levels structure

Observing its folders makeup, as measured by the number of files per folder level (Figure 2), we observe that the majority of the files have been located at Levels 2 and 3. Level 4 experimented a sudden midlife increase at around release 25. Several new folders were added on Level-4, other moved from other parts of the system, which also significantly affect Level-3.

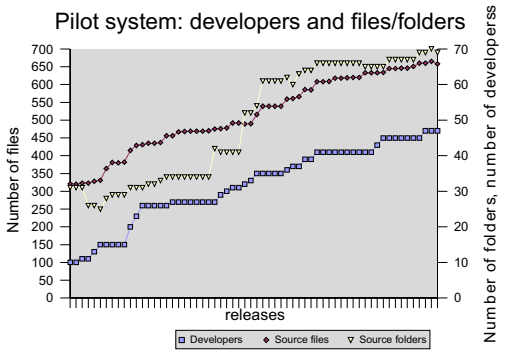
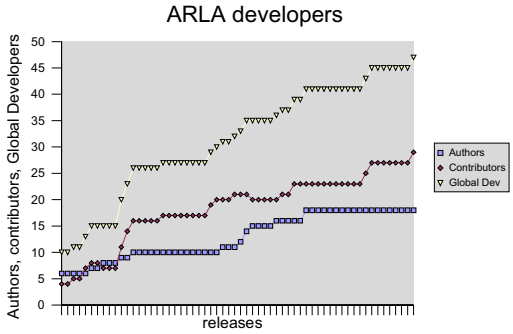


Figure 3 - Developers, Files and Folders growth

In Figure 3, we observe the makeup of evolution of ARLA as depicted by the total number of source files, and touched files over releases. The growth trend can be interpreted as two segments of decaying growth rate with a midlife growth regeneration point. The trend presents similarities with those observed in commercial systems [Ramil & Smith 2002]. The plot of files per level as in Figure 2 is particularly useful for identifying this regeneration points and Figure 2 suggest that the mid-life regeneration in growth rate was facilitated by a restructuring of the software

In Figure 3 it's displayed the trend of both the structural components (files and folders) growth, as well as the developers growth. We observe that the pattern of growth for developers resembles the folders growth, while files grow much faster. When investigating the dualism authors-contributors, we see that contributors grow faster than authors, i.e. it's easier to become a contributor than author (Figure 4).



6. Conclusions

We have deeply analyzed an OSS system from the point of view of code structure. Several subsequent releases were parsed in the perspective of both source code (LOCs, SLOCs, etc.), and source components (source files and folders). Based on this second view, we derived an integrated, graphical view of the system's structure, and we drew graphs for some interesting points we observed in components growth.

Interestingly enough, the system had a very well defined structure at the very beginning of its evolution (first available release), and it was able to evolve that structure with new functionalities. Part of this evolution is due to easily porting the existing system over a new architecture, which resolves in a similar patter over the tree structure (rightmost parts of Figure 6 and Figure 7). Part of the evolution, though, is due to new functionalities, which are either resolved in new subtrees (semicircle in Figure 6), or left as potential seeds over which connect new components (the leftmost rectangle in Figure 7 is derived from a longly lived folder, while other end-like folders are similar to their initial status).

7. References

[Aoki et al. 2002] Aoki, A., K. Hayashi, K. Kishida, K. Nakakoji, Y. Nishinaka, B. Reeves, A. Takashima, and Y. Yamamoto, 2002, "A Case Study of the Evolution of Jun: An Object-Oriented Open-Source".

[Antoniades 2003] Antoniades P., Samoladas I., Stamelos I., Bleris G.L. "Dynamical simulation models of the Open Source Development process". To appear in Free/Open Source Software Development, ed. Stefan Koch, Idea Group, Inc.

[Barahona et al 2001] Jesús M. González-Barahona, Miguel A. Ortuño Pérez, Pedro de las Heras Quirós, José

Centeno González, Vicente Matellán Olivera" Counting potatoes: The size of Debian 2.2", http://people.debian.org/~jgb/debian-counting/counting-potatoes-0.2/

[Basili et al. 1996] Basili, V. R. et al (1996), "Understanding and Predicting the Process of Software Maintenance Releases". In Proceedings of the18th International Conference on Software Engineering, Berlin, March 25-29, IEEE CS Press, Los Alamitos, CA.

[Belady et al. 1976] Belady L.A, Lehman M.M, "A Model of Large Program Development", IBM Systems J., vol. 15, no. 1, pp. 225-252, 1976.

[Capiluppi 2003] Capiluppi A., "Studying the evolution of OSS projects", on the Proceedings of the 7th International Conference on Software Maintenance, September 2003

[Capiluppi et al. 2002] Capiluppi A., Lago P., Morisio M., 2002, "Characteristics of Open Source Projects", on the Proceedings of the 7th European Conference on Software Maintenance and Reengineering, March 2003

[Chapin et al 2001] Chapin N., Hale J.E., Khan K.M., Ramil J.F. & Tan W.G. (2001), "Types of Software Evolution and Software Maintenance", Journal of Software Maintenance and Evolution: Res. and Practice, 13(1), January-February: 1 – 30

[German 2002] German D., 2002, "Using software trails to rebuild the evolution of software", 2002 ELISA Workshop, Amsterdam, The Netherlands

[Godfrey et al 2000] Godfrey, M., and Q. Tu, 2000, "Evolution in Open Source Software: A Case Study". In Proceedings of 2000 International Conference on Software Maintenance.

[Graphviz] Graphviz - open source graph drawing software http://www.research.att.com/sw/tools/graphviz/

[Kemerer et al. 1999] Kemerer, C.F., and S. Slaughter. "An Empirical Approach to Studying Software Evolution". In IEEE Transactions on Software Engineering, 1999. 25(4): 493-509.

[Koch 2000] Koch S., Schneider G., "Results from Software Engineering Research into Open Source Development Projects Using Public Data", in zum Tätigkeitsfeld Informationsverarbeitung und Informationswirtschaft, Hans R. Hansen und Wolfgang H. Janko (Hrsg.), Nr. 22, Wirtschaftsuniversität Wien, 2000.

[Lehman 1969] Lehman M.M. (1969); The Programming Process, IBM Res. Rep. RC 2722, Dec. 1969: 46 pp. Also as Chapter 3 in [Lehman & Belady 1985]

[Lehman 1974] Lehman M.M. (1974); Programs, Cities, Students, Limits to Growth?, Inaugural Lecture, in Imperial College of Science and Technology Inaugural Lecture Series, v. 9, 1970, 1974: 211 229. Also in Programming Methodology, Gries D (ed.), Springer Verlag, 1978: 42 62. Reprinted as Chapter 7 in [Lehman & Belady 1985]

[Lehman et al., 1985] Lehman M. M., Belady L. A., 1985, "Program Evolution: Processes of Software Change". Academic Press.

[Lehman et al., 1997] Lehman M.M., J.F. Ramil, P.D. Wernick, D.E. Perry, and W.M. Turski, "Metrics and Laws of Software Evolution The Nineties View," Proc. Fourth Int'l Software Metrics Symp., Metrics '97, Albuquerque, N.M., 1997.

[Lehman, 1980] Lehman M.M, "Programs, Life Cycles, and Laws of Software Evolution", Proc. Special Issue Software Eng., IEEE, vol. 68, no. 9, pp. 1,060±1,076, 1980.

[Mockus et al. 2002] Mockus, A., Fielding, R.T., Herbsleb, J.D., 2002, "Two Case Studies of Open Source Development: Apache and Mozilla". In ACM Transactions on Software Engineering and Methodology Vol.11, No. 3, 2002, 309-346.

[Nakakoji et al. 2002] Nakakoji K., Yamamoto Y., Nishinaka Y., Kishida K.,Ye Y., 2002, "Evolution Patterns of Open-Source Software Systems and Communities". In Proceedings of International Workshop on Principles of Software Evolution (IWPSE 2002).

[Nikora & Munson 2003] Nikora A.P. & Munson J.C., "Understanding the Nature of Software Evolution", Proc. ICSM 2003, 22 – 26 Sept., Amsterdam, pp. 83 - 93

[Ramil & Smith 2003] Ramil J.F. and Smith N., "Qualitative Simulation of Models of Software Evolution", Journal of Software Process: Improvement and Practice, vol. 7, 2002, pp. 95 – 112

[Shankland 2000] Shankland S. , 2000, "Linux kernel release falls behind schedule", http://news.com.com/2100-1001-240061.html?legacy=cnet&tag=st.ne.1002.thed.1003-200-1808165

[Stamelos 2002] Stamelos, I., Angelis, L., Oikonomou, A., Bleris, G.L., "Code Quality Analysis in Open-Source Software Development", Information Systems Journal, 2nd Special Issue on OS Software, 12(1), January 2002, pp. 43-60.

[XSCC] A tool for extraction source lines of code, http://members.tripod.com/vgoenka/unixscripts/xscc.html

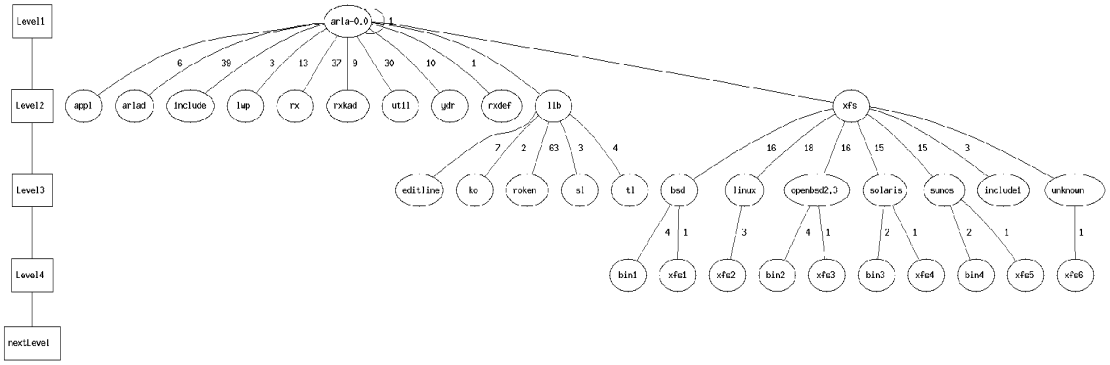


Figure 5 - Arla structure: first stage

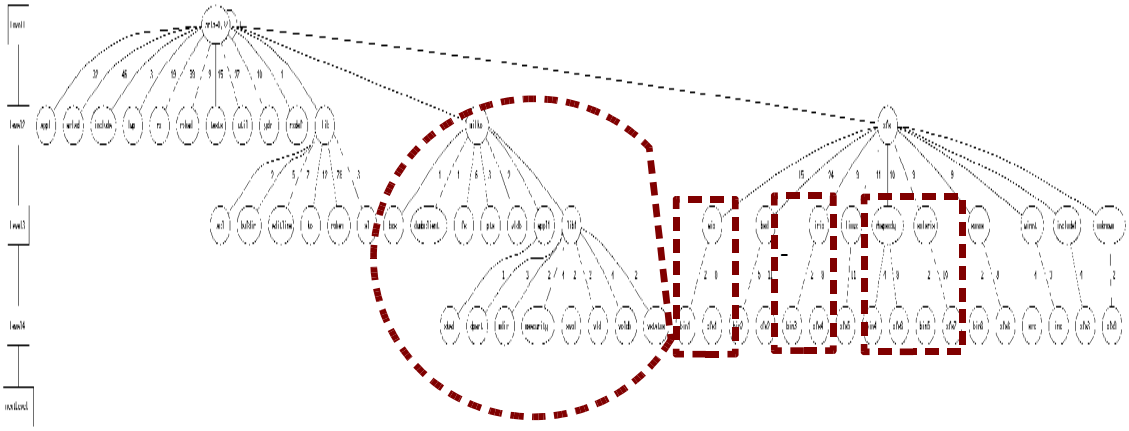


Figure 6 - Arla structure: intermediate stage (in red new branches and folders are displayed)

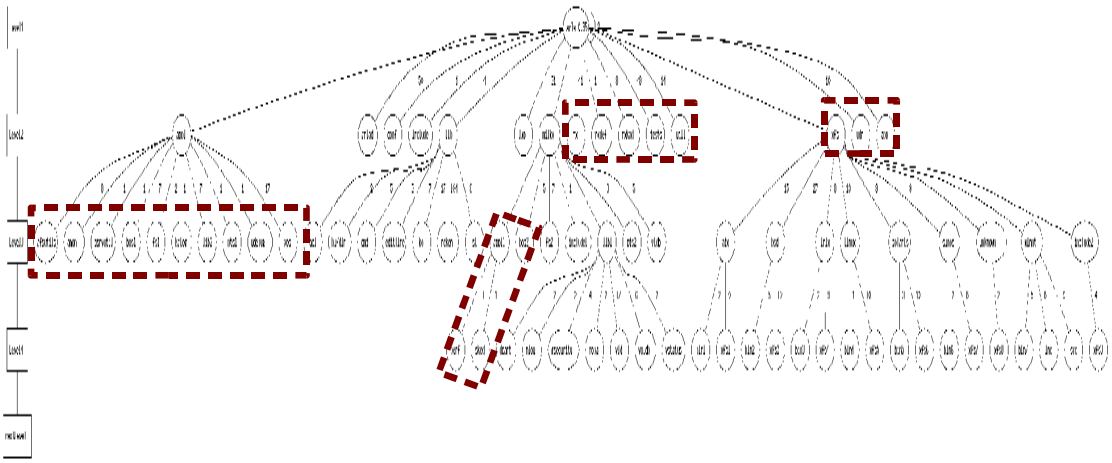


Figure 7 - ARLA structure: final stage