

# Qualitative Analysis and Simulation of Open Source Software Evolution

## *Extended Abstract*

J.F. Ramil N. Smith A. Capiluppi<sup>1</sup>  
Computing Dept.  
The Open University  
Walton Hall, Milton Keynes, U.K.  
{j.f.ramil,n.smith,a.capiluppi}@open.ac.uk

### **Background and motivation**

In previous work [Ramil & Smith 2002; Smith & Ramil 2003] we applied qualitative simulation methods to model the evolutionary trends of a set of five commercial software systems. In that study, we concluded that qualitative simulation and qualitative abstraction were useful tools for breaching the gap between empirical generalisations in natural language, such as Lehman's laws of *E*-type software evolution [Lehman 1974; Lehman & Belady 1985], and empirical data reflecting the evolutionary behaviour of software [Ramil & Smith 2002; Smith & Ramil 2003]. In this paper, we apply similar techniques to study the evolution of a number of open source (OS) software systems. This study investigates a number of research questions:

- What are the differences and similarities between the long-term evolutionary characteristics of OS and proprietary software? What is the support for Lehman's laws of software evolution in OS domains?
- What are the difference and similarities between the OS process simulation models and those reflecting proprietary environments?
- What are the implications of the tendency to increasing complexity [Lehman 1974] and, in general, the software ageing phenomenon [Parnas 1994] in OS and how is this issue tackled at the process level? Is software continually refactored or is it subject to massive re-structuring at particular moments of its lifetime?
- What lessons can be applied to software process improvement in general, from the study of OS processes?

The full paper aims at exploring how qualitative reasoning techniques, already justified and presented in [Ramil & Smith 2002; Smith & Ramil 2003] can be useful in tackling the above questions.

### **Results**

We have extracted metric data on the growth over releases and the number of files touched for seven different OS systems [Capiluppi 2003]. We have analyzed the trends for these systems using our previously-developed abstraction techniques [Ramil & Smith 2002], as presented in table 1 below. We have compared this results to our previous work which focused on proprietary software [Ramil & Smith 2002]. We have found that OS and proprietary exhibit very similar behaviours, but there are two systems in the OS set which exhibit differences, such as the presence in one of the OS systems of a significant decline in size (LCRZO) and one system with a very singular stagnated growth pattern (Gchemical). Moreover, four other OS systems (Arla, Ganymede, Gwydion and Sip) display growth stagnation periods. Both stagnation and decline were absent from the qualitative trends for the proprietary systems studied so far. The qualitative simulation models we have studied in [Ramil & Smith 2002] did display stagnation periods, not as a single segment as in LCRZO system, but as a follow-on to a period of excessive growth. For the rest of the OS systems, the outputs of the qualitative simulation models do replicate the observed behaviours. This

---

<sup>1</sup> Currently a visiting researcher at the OU. Also affiliated to Politecnico of Torino, Italy.

provides further support to the view of the software evolution phenomenon which is encapsulated in the models, and in particular to the Lehman's laws from which they were derived.

System	Full Pattern	Simplified Pattern
<i>Arla</i>	? ( ) ( / ) / ( \ ) ? ) ( - / ) / ? ) - / ?	( ) ( ) ( ) - ) - /
<i>Ganymede</i>	? ( - /	( - /
<i>Ghemical</i>	- ?	-
<i>Gist</i>	? / ( ) / ?	( )
<i>Gwydion</i>	? / ) / ) - ?	) -
<i>Lcrzo</i>	/ - \ ) / / ( / ) / ( ) \ ) / /	/ - ) ( ) ( ) ( ) \ ) / /
<i>Sip</i>	? / ) / ( - ?	) ( -

Table 1 – Qualitative trends extracted from growth data for several OS systems

In any process simulation attempt, it is important to determine the boundary between the system to be modelled and the rest of the world. Given that some OS systems are almost literally evolved *by the world*, process modelling would appear to be an almost impossible task. However, rather surprisingly, in spite of this and other differences between OS and proprietary software evolution which are discussed in the full paper, initial results suggest that the simulation models we have studied and developed so far [Ramil & Smith 2002] will not require fundamental changes in their structure. On the contrary, OS systems appear to display some of the behaviours predicted by the models and not found in the proprietary systems previously studied. The qualitative models we have studied so far represent the process at a very high level of abstraction and reflect only a few attributes: functional size, effort applied and productivity. One of our previous modelling assumptions, the application of constant effort for evolution, may have to be changed, just simply because OS systems can receive input from an almost unlimited pool of contributors. In this work we are addressing how to measure the level of effort applied by using *files touched* as a surrogate for effort.

### Further Work

In other to address the set of research questions indicated at the beginning of this extended abstract, we need to test the simulation models which consider software ageing effects and the possible impact of refactoring and restructuring work. In order to so we need some measure of both the complexity of the software and the amount of work applied to control such complexity. Classifying evolution activity into types is not a trivial issue. In particular, we would like to identify the portions of the code which have been subject to restructuring and refactoring [Fowler 1999]. At the total system level, we can use structural analysis techniques [Capiluppi *et al* 2004] and study the patterns of the evolution of the architecture, in search for clean-up and restructuring events. At the module level, we believe that we can make use of techniques for the identification of modules which would benefit from refactoring. In principle one would wish to examine the code directly [Fowler 1999] but given the size of some of the systems, this is not feasible, so one will have to use indirect approaches, such as those which are metrics-based<sup>2</sup>.

### Final Remarks

Given the availability of raw data in the form of code repositories, for example, for the study of OS evolution, the authors are confident that they will be able to pursue the questions

<sup>2</sup> We are considering how Nikura and Munson's software measurement approach [Nikora & Munson 2003] could be use retrospectively, that is comparing a later release to a previous release, for the purpose of determine which modules underwent complexity control work.

indicated in the introductory section to an extent which has not been possible in the study of proprietary systems. The authors wish to participate in the ProSim 2004 workshop in order to reporting on the status of their investigation into the evolution of OS systems, its impact on previous findings in this topic and also discuss with the process simulation community how several measurement, analysis and simulation issues – described in the full paper – mentioned above can be successfully addressed. One of the long term goals of this line of study is to better understand the OS evolution phenomenon and, in particular, to determine what are the main characteristics of the evolution dynamics of OS software, in its various modalities, how they differ from those observed in conventional proprietary software and how the findings can, in general, contribute towards extending and refining the body of knowledge about the software evolution phenomenon.

## References

- [Capiluppi 2003] Capiluppi A., *Models for the Evolution of OS Projects*, Proc. ICSM, Amsterdam, 22 – 26 Sept. 2003
- [Capiluppi *et al* 2004] Capiluppi A., Morisio M. & Ramil J.F., *Code Architecture Metrics as a Means to Comprehend Software Systems: A Case Study*, submitted for publication
- [Fowler 1999] Fowler M., *Refactoring: Improving the Design of Existing Code*, Addison-Wesley, 1999
- [German 2004] German D., *Using Software Trails to Rebuild the Evolution of Software*, Journal of Software Maintenance and Evolution, to appear
- [Godfrey & Tu 2000] Godfrey M.W. & Tu Q., *Evolution in Open Source Software: A Case Study*, Proc. ICSM 2000, 11-14 Oct., San Jose, CA: 131 – 142
- [Lehman 1974] Lehman M.M., *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
- [Lehman & Belady 1985] Lehman M.M. & Belady L.A. (eds.), *Program Evolution – Processes of Software Change*, Academic Press, London, 1985
- [Nikora & Munson 2003] Nikora P. & Munson J.C., “Understanding the Nature of Software Evolution”, Proc. ICSM 2003, Amsterdam, 22 – 26 Sept.
- [Parnas 1994] Parnas D.L., *Software Aging*, Proc. ICSE 16, May 16-21, 1994, Sorrento, Italy: 279 – 287
- [Ramil & Smith 2002] Ramil J.F. & Smith N., *Qualitative Simulation of Models of Software Evolution*, Journal of Software Process: Improvement and Practice, vol. 7, 2002, pp. 95 – 112
- [Smith & Ramil 2003] Smith N & Ramil J.F., *Qualitative Abstraction and Simulation in the Study of Software Evolution*, ProSim 2003, Portland, Oregon, May 3-4, 2003