Software Architectures and Open Source Software – Where can Research Leverage the Most?

Budi Arief, Cristina Gacek, and Tony Lawrie
Centre for Software Reliability
Department of Computing Science
University of Newcastle
Newcastle upon Tyne NE1 7RU
United Kingdom
{L.B.Arief, Cristina.Gacek, A.T.Lawrie}@ncl.ac.uk

1 INTRODUCTION

Software architectures have been playing a central role in software engineering research for some years now. They are considered of pivotal importance in the success of complex software systems development. However, with the emergence of Open Source Software (OSS) development, a new opportunity for studying architectural issues arises. In this paper, we introduce accepted notions of software architectures (Section 2), discuss some of the known issues in OSS (Section 3), resulting in a set of aspects we consider to be relevant for future research (Section 4).

2 SOFTWARE ARCHITECTURES

Software Architecture can be defined as the structure(s) of a system, which comprise software components, the externally visible properties of those components and the relationships among them [1]. It typically acts as a bridge between software requirements and implementation. The architectural design of a software system can represent the most vital artefact for a software project, as it directly impacts upon the important management and technical processes of production and integration [2]. Hence, a sound software architecture is desirable in order to build a solid software system.

Some of the reasons why software architectures are believed to be important are that they: facilitate the communication among stakeholders, represent the manifestation of the earliest design decisions, and constitute a relatively small and understandable model of how a system is structured [1]. Garlan further elaborates six aspects of software development within which software architecture can play an important role: facilitating understanding by using high-level abstractions, supporting reuse at multiple levels of granularity, providing a partial blueprint for development by indicating the major components and dependencies among them, exposing the dimensions among which a system is expected to evolve, providing analysis opportunities at early stages of development, and for basic management support [3].

In order to fulfil their expected roles, software architectures should be modularised. This modularisation plays a triple role:

- It facilitates understanding by using high-level abstractions and reducing the complexity of the task at hand,
- It highlights areas where work can occur in a concurrent and distributed fashion, and
- It can also be used to determine the organizational structure that should be in place for developing the system being considered[1].

Based on its intrinsic characteristics, software architecture design becomes essential while developing a large complex software system. It should be the responsibility of a main architect (group) responsible for keeping *the vision* of the overall system [4]. Additionally, in order to support system evolution, while avoiding architecture erosion and drift [5], the software architecture must also be evolved accordingly, at times requiring some major restructuring.

The issues addressed in this section have been recognised within proprietary software development. In the next section, we explore how software architectures relate to OSS development.

[1] Conversely, it is important to note here that organizational structures have also been known to influence the creation of software architectures, by having the latter reflect the areas of expertise and availability of people in the former.

3 SOFTWARE ARCHITECTURES IN OPEN SOURCE SOFTWARE VS. PROPRIETARY SOFTWARE

With the emergence of Open Source Software (OSS) development [6, 7] as an alternative approach in building software systems, it is interesting to investigate whether software architecture still plays as prominent a role in the OSS development as in the traditional or proprietary software development.

At its root, the popular OSS definition makes the distinction between proprietary software development and OSS as being a centralised vs. decentralised software development argument, where the process of carefully controlling the construction of software is replaced by a rapid evolutionary process of voluntary submissions from all over the world [7]. Although a rapid, decentralised, and participative approach is not unique to OSS development, it does pose fundamental architectural considerations for the development of software systems.

However, OSS also presents a different attitude towards software development. By giving away the source code, OSS lets anyone inspect and modify the code as they please. There is also no explicit planning or project management in the open source approach, which puts a lot of strain on the architecture of the system.

Unlike most traditional software development, the original interest and vision in OSS projects usually emanates from the initiating projects owners [7]. Such individuals often assume complete authority [8]. Even in "shared-leadership" situations, such as the Apache web server, investigations have established that the core-developers still exercise the major influence over the design and direction of OSS development [9]. Consequently, in contrast to traditional software approaches, OSS project managers seem to possess greater power to determine the architectural direction of the software product. In this respect, even in the (supposedly) decentralised OSS process, the traditional architect role still appears to be a prerequisite for preserving the conceptual integrity of software [4]. However there are views expressed that OSS leaders may abuse this power to protect their own position by concealing the software architecture [10]. In doing so, they risk removing the blueprint that is vital for detailed understanding. Nevertheless, even in the absence of an explicit architectural blueprint, it may still be possible that the OSS development process can overcome the traditional software development barrier (c.f. Section 2) by narrowing the conceptual gap between requirements and implementation. The reasons being:

- Many of the users of OSS software are also contributing developers [11],
- Creating programs for oneself has long been considered less demanding than developing software for others [12],
- The rapid releases and early feedback allow a greater level of incremental development in the OSS process [7, 13].

Finally, initial OSS releases may be lacking in code refinement and contain many residual faults [7]. Nevertheless, it has been recognised that for OSS projects to be successfully initiated, evolved, and maintained, the architecture must be modularised to promote code comprehension and concurrent collaboration [14].

An indication of the importance of architectural coherence in OSS was provided by Eric Raymond's interpretation of why the controversial release of Netscape's Mozilla source-code did not fulfil initial expectations [7:p 77]:

> *"…going 'open' will not necessarily save an existing project that suffers from ill-defined goals or spaghetti code, or any of the software engineering's other chronic ills."*

It should also be noted that most OSS endeavours are undertaken in fairly stable domains. This characteristic can help explain why major architectural restructuring is hardly ever witnessed.

4 FUTURE DIRECTIONS

In recognition of both traditional and OSS architectural issues (see Sections 2 and 3), a number of points can be tentatively stated for future discussion and research:

1. Analyse how the OSS organisational structures affect their software architectures.
2. Investigate how OSS approach may harmonise the two extremes of the centralised and decentralised software development.
3. Compare how the gap between requirements and implementation is handled within OSS vs. proprietary software development.
4. Explore the role of software architectures in OSS development.
5. On the issue of software architecture decay:
   a) Assess whether architectural decay happens in OSS. If so, how quickly does it occur, and how is it dealt with?
   b) Try to get some insight on architectural drift and erosion by studying OSS projects that failed.
6. Leveraging the benefits offered by interdisciplinary research in order to determine:
   a) which OSS characteristics are suitable for adoption within other software development processes,
   b) what implications the OSS characteristics may have, for example on "time-to-market",
   c) the communication patterns in OSS, both in terms of communication mode and quality of content.

ACKNOWLEDGEMENT

This paper has been funded by the UK EPSRC project on Dependable Interdisciplinary Research Collaboration (DIRC – http://www.dirc.org.uk/). We would like to thank Denis Besnard, Michael Jackson, and Cliff Jones for various fruitful discussions contributing towards this paper.

REFERENCES

[1] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*: Addison-Wesley, 1998.

[2] W. Royce, *Software Project Management: A Unified Framework*: Addison Wesley, 1998.

[3] D. Garlan, *Software Architecture: a Roadmap*, in *The Future of Software Engineering*, A. Finkelstein, Ed.: ACM Press, pp. 93-101, 2000.

[4] F. P. Brooks, *The Mythical Man Month: Essays on Software Engineering*: Addison-Wesley, 1995.

[5] D. E. Perry and A. L. Wolf, "Foundations for the Study of Software Architecture," *ACM Software Engineering Notes*, 4, vol. 17, pp. 40-52, October 1992.

[6] The Open Source Initiative online at *http://www.opensource.org*.

[7] E. S. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*: O'Reilly & Associates, 1999.

[8] M. Maclachlan, "Panelists Describe Open Source Dictatorships," in *TechWeb News*, 12 August 1999, online at *http://www.techweb.com/*.

[9] A. Mockus, R. T. Fielding, and J. Herbsleb, "A Case Study of Open Source Software Development: The Apache Server," presented at 22nd International Conference on Software Engineering, Limerick, Ireland, pp. 263-272, 2000.

[10] N. Bezroukov, "A Second Look at the Cathedral and the Bazaar," in *First Monday*, 9 December 1999, online at *http://www.firstmonday.dk/issues/issue4_12/bezroukov/*.

[11] K. Johnson, "Towards a Descriptive Process for Open-Source Software Development," Submission for the 2nd Workshop on software Engineering over the Internet, online at *http://www.cpsc.ucalgary.ca/~johnsonk/SENG/SENG691/towards.htm*.

[12] G. M. Weinberg, *The Psychology of Computer Programming*: Dorset House, 1971.

[13] R. C. Pavlicek, *Embracing Insanity: Open Source Software Development*: SAMS Publishing, 2000.

[14] T. Bollinger, R. Nelson, K. M. Self, and S. J. Turnbull, "Open-Source Methods: Peering Through the Clutter," *IEEE Software*, July/August, pp. 8-11, 1999.